ΣΥΣΤΗΜΑΤΑ ΑΝΑΚΤΗΣΗΣ ΠΛΗΡΟΦΟΡΙΩΝ

ΑΚΑΔΗΜΑΙΚΟ ΕΤΟΣ: 2019

ΙΑΚΩΒΟΣ ΕΥΔΑΙΜΩΝ 3130059

ΦΑΣΗ 1^{Η}

ΔΙΑΔΙΚΑΣΤΙΚΑ ΓΙΑ ΤΗΝ ΕΚΤΕΛΕΣΗ ΤΟΥ ΠΡΟΓΡΑΜΜΑΤΟΣ ΤΗΣ ΦΑΣΗΣ 1

Για τους σκοπούς αυτής της φάσης της εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Python και το IDE της PyCharm. Επίσης χρησιμοποιήθηκε η ElasticSearch-6.6.2. Για την διαχείριση του ElasticSearch χρησιμοποιήθηκε Python Client.

Για την εκτέλεση του script της Python είναι απαραίτητη η εγκατάσταση 2 packages, το package xmltodict και το package της elasticsearch. Για την εγκατάσταση του package xmltodict πρέπει στο command prompt(cmd) να πληκτρολογηθεί η εντολή pip install xmltodict. Ενώ για το package της elasticsearch πρέπει πάλι στο command prompt να πληκτρολογηθεί η εντολή pip install elasticsearch. Εγκαταστήστε πρώτα τα δύο πακέτα και μετά ανοίξτε το με οποιοδήποτε IDE. Επίσης, σε ορισμένους IDE χρειάζεται να οριστεί ένας Interpreter, στην περίπτωση αυτή προτείνω την χρήση του Interpreter που είναι εγκαταστημένος στον υπολογιστή σας ώστε να έχει και τα νέα packages που εγκαταστήθηκαν. Το επισημάνω αυτό καθώς υπάρχει η πιθανότητα να σας έχει σαν επιλογή την χρήση του Interpreter που χρησιμοποιήθηκε για το project και που ήταν εγκαταστημένος στον δικό μου υπολογιστή.

ΥΛΟΠΟΙΗΣΗ ΦΑΣΗΣ 1 ΚΑΙ ΕΠΕΞΗΓΗΣΗ ΚΩΔΙΚΑ

Αρχικά μέσα στο παραδοτέο project υπάρχει ένας άλλος φάκελος που λέγεται Parsed files όπου μέσα σε αυτόν τον φάκελο αντιγράφουμε τα xml αρχεία που αποτελούν την συλλογή μας. Πριν εκτελέσουμε τον κώδικα του project πρέπει πρώτα να ξεκινήσουμε το elasticsearch εκτελώντας το elasticsearch.bat που βρίσκεται μέσα στο φάκελο bin του

elasticSearch. Στο script phase1.py γίνονται όλες οι διαδικασίες από την επεξεργασία των xml αρχείων της συλλογής και την μετατροπή τους σε json αρχεία μέχρι και την σύνδεση με την elasticsearch, την δημιουργία του index, την φόρτωση της συλλογής στο index, την επεξεργασία των ερωτημάτων, την εκτέλεση των αποτελεσμάτων και την δημιουργία του αρχείου με τα αποτελέσματα των ερωτημάτων μας που θα κατατεθεί στο trec_eval. Στη μέθοδο process_xml() του script αρχικά διαβάζουμε τα xml αρχεία της συλλογής που τα έχουμε αντιγράψει στο φάκελο Parsed files του project μας, καθώς τα διαβάζουμε ένα-ένα το κάθε xml αρχείο δημιουργούμε το DOM tree του και έπειτα δημιουργούμε το tag με όνομα text και το εισάγουμε στο DOM tree. Αφού το εισάγουμε μετά το tag acronym τότε, παίρνοντας πρώτα το περιεχόμενο του tag title και έπειτα το περιεχόμενο του tag objective, καθαρίζουμε τα δύο περιεχόμενα αυτών των δύο tags από οποιονδήποτε special character(χρήση πακέτου re για την αφαίρεση κάποιων δοθέντων regular expressions) γεγονός το οποίο θα μας βοηθήσει για την σωστή φόρτωση των αρχείων στον index αργότερα και για την σωστή εκτέλεση των queries ώστε να ανακτήσουμε τα σχετικά κείμενα με όσο το δυνατόν καλύτερα αποτελέσματα. Αφού, κάνουμε τις προαναφερθέντες ενέργειες μετά ενώνουμε τα επεξεργασμένα περιεχόμενα των tags title και objective σε ένα με πρώτα τα περιεχόμενα του title και τα εισάγουμε ως περιεχόμενο για το νέο tag text που δημιουργήσαμε προηγουμένως. Μετά από αυτές τις ενέργειες διαγράφουμε τα tags title και objective καθώς και το περιεχόμενο τους από το DOM tree και μετατρέπουμε το DOM tree σε String. Αφού μετατρέψουμε το DOM tree σε String αυτό το string το μετατρέπουμε σε ένα collection τύπου dictionary μέσω του πακέτου της python xmltodict, όπου με την σειρά του μετατρέπουμε το συγκεκριμένο dictionary σε json string μέσω της μεθόδου dumps() του πακέτου json της python. Το παραγόμενο json string το αποθηκεύουμε σε μία λίστα. Αφού επεξεργαστούμε όλα τα xml αρχεία της συλλογής και τα κάνουμε json string και τα τοποθετήσουμε στην λίστα μας κατά τον τρόπο που περιγράφηκε παραπάνω, έπειτα γυρνάμε αυτή τη λίστα με τα αποθηκευμένα json strings στην main μέθοδο μας ώστε αργότερα να τα φορτώσουμε στον index. Μετά το πέρας της μεθόδου process_xml(), δημιουργούμε την σύνδεση με την elasticsearch τοπικά στην port 9200(αφού πρώτα έχει ξεκινήσει σωστά η elasticsearch μέσω της εκτέλεσης του αρχείου elasticsearch.bat). Αφού κάνουμε την σύνδεση με την elasticsearch και κάνοντάς την ping ώστε να είμαστε σίγουροι ότι μας "ακούει" και είναι διαθέσιμη, τότε καλούμε την μέθοδο index _creation η οποία ουσιαστικά όπως λέει και το όνομά της δημιουργεί τον index με τα κατάλληλα settings. Για την δημιουργία του index χρησιμοποιούμε τον analyzer English, ο οποίος είναι built-in analyzer που προσφέρεται από την elasticsearch. Ο English analyzer προσφέρει:

- tokenize text into individual words: π.χ. *The quick brown foxes → [The, quick, brown, foxes]*
- Lowercase tokens: π.χ. *The → the*
- Remove common stopwords: π.χ. [*The, quick, brown, foxes] → [quick, brown, foxes]*
- Stem tokens to their root form: π.χ. *foxes → fox*

Αποτελώντας έτσι έναν κατάλληλο analyzer για τις απαιτήσεις της εργασίας μας. Για να τον χρησιμοποιήσουμε ορίζουμε ως default στα settings του index τον English analyzer. Με αυτό τον τρόπο χρησιμοποιείται ο English analyzer και κατά την φόρτωση των αρχείων της συλλογής στον index και κατά την αναζήτηση για την ανάλυση των ερωτημάτων. Χρησιμοποιούμε τον ίδιο analyzer και κατά τον index-time και κατά τον search-time καθώς αυτή η πρακτική αποτελεί μια καλή τακτική ώστε να έχουμε καλύτερα αποτελέσματα κατά την αναζήτηση.

```
settings = {
    "settings": {
        "index": {
            "analysis": {
                "analyzer": {
                    "default": {
                        "type": "english"
                    }
                }
            }
        }
    }
}
```

Δεν χρειάζεται να ορίσουμε mapping και επίσης δεν χρειάζεται να κάνουμε οποιαδήποτε άλλη παραμετροποίηση στον index καθώς χρησιμοποιούμε το default weighting scheme(BM25). Οπότε αφού έχουμε ορίσει σωστά τα settings μας δημιουργούμε τον index μας με όνομα collection και type project. Αφού δημιουργήσουμε τον index σκοπός μας έπειτα είναι να φορτώσουμε την συλλογή μας η οποία είναι αποθηκευμένη σαν json strings σε μία λίστα. Περνάμε αυτή την λίστα καθώς και το όνομα του index όπως και το στιγμιότυπο του elasticsearch μέσα σε μία μέθοδο με το όνομα bulk_load() και καλούμε αυτή την μέθοδο. Αυτή η μέθοδος παίρνει ένα-ένα τα json strings από την λίστα που είναι αποθηκευμένα και τα αντιγράφει σε μία μεταβλητή με όνομα “a” η οποία εκτός από το κάθε json string έχει και κάποια ακόμα πεδία που ουσιαστικά

βοηθάνε την elasticsearch να φορτώσει σωστά τα json strings στον index που θέλουμε. Αφού, ουσιαστικά ορίσουμε το περιεχόμενο της μεταβλητής "a" όπως πρέπει μετά αποθηκεύουμε αυτή την μεταβλητή σε μία λίστα και αφού διαβάσουμε όλα τα json strings και ορίσουμε την μεταβλητή a για κάθε json string και αποθηκεύσουμε κάθε μεταβλητή a στην λίστα μας τότε φορτώνουμε την λίστα αυτή στον index μας μέσω του Bulk API το οποίο μας προσδίδει την δυνατότητα να φορτώσουμε γρήγορα την συλλογή στον index. Μετά την φόρτωση της συλλογής στον index καλείται η μέθοδος search() η οποία λαμβάνει σαν παραμέτρους το στιγμιότυπο της elasticsearch και το όνομα του index. Η μέθοδος όταν κληθεί με την σειρά της καλεί την μέθοδο query_processing() η οποία διαβάζει από το αρχείο testingQueries.txt,που βρίσκεται στον φάκελο Queries του project, τα ερωτήματα ένα-ένα, τα επεξεργάζεται αφαιρώντας από αυτά το πρόθεμα Q01,Q02…Q10 καθώς επίσης αφαιρούνται και τα special characters από κάθε ερώτημα. Το κάθε ερώτημα διαφοροποιείται από τα υπόλοιπα καθώς το καθένα βρίσκεται σε ξεχωριστή σειρά του αρχείου. Αφού γίνει αυτή η επεξεργασία το κάθε ερώτημα αποθηκεύεται σε μία λίστα και αφού διαβαστούν, επεξεργαστούν και αποθηκευτούν όλα τα ερωτήματα στην λίστα, αυτή η λίστα επιστρέφεται στην καλούσα μέθοδο search(). Παίρνοντας την λίστα που επιστράφηκε από την μέθοδο query_processing(), η μέθοδος search() με την σειρά της διαβάζει ένα-ένα τα ερωτήματα της λίστα και αναζητεί στον index και συγκεκριμένα στο πεδίο text του index το κάθε ερώτημα που διάβασε. Κατά την αναζήτηση επιστρέφονται τα 20 πρώτα κείμενα που είναι πιο σχετικά με το κάθε ερώτημα που τέθηκε στον index. Αυτά τα 20 κείμενα που ανακτώνται σε κάθε αναζήτηση γράφονται με μια συγκεκριμένη μορφολογία σε ένα αρχείο txt με το όνομα myResults, το οποίο αρχείο αυτό αποθηκεύεται στον φάκελο Results of queries του project. Αφού διαβάσουμε κάθε ερώτημα της λίστας και κάνουμε τις παραπάνω ενέργειες που περιεγράφηκαν τότε το πρόγραμμα τερματίζει.

Η μέθοδος search():

```
def search(els, index_n):
    path = 'Results of queries'
    f_name = 'myResults.txt'
    fullname = os.path.join(path, f_name)
    fh = open(fullname, 'w+')
    queries = query_processing()
    count = 1
    for q in queries:
        query = {
            'from': 1, 'size': 20,
            'query': {
                'match': {
                    'project.text': {
                        'query': q
                    }
                }
            }
        }
        res = els.search(index=index_n, doc_type='project', body=query)
        for hit in res['hits']['hits']:
            if count < 10:
                line = 'Q0'+str(count)+" "+"Q0"+" "+hit['_source']['project']['rcn']+" "+"0"+" "+str(hit['_score'])+" "+"es"+"\n"
            else:
                line = 'Q10'+" "+"Q0"+" "+hit['_source']['project']['rcn']+" "+"0"+" "+str(hit['_score'])+" "+"es"+"\n"
            fh.write(line)
        count += 1
    fh.close()
```

Ενδεικτική μορφή του αρχείο myResults.txt :

```
Q01 Q0 193373 0 270.78262 es
Q01 Q0 205685 0 241.97241 es
Q01 Q0 193715 0 239.73412 es
Q01 Q0 193375 0 234.37888 es
Q01 Q0 206230 0 225.38728 es
Q01 Q0 210137 0 219.48593 es
Q01 Q0 193353 0 217.84995 es
Q01 Q0 211346 0 217.47359 es
Q01 Q0 193386 0 214.29146 es
Q01 Q0 206228 0 212.33821 es
Q01 Q0 202703 0 211.17493 es
```

ΕΡΓΑΛΕΙΟ TREC_EVAL

Στον φάκελο που βρίσκεται το εργαλείο trec_eval αντιγράφουμε το αρχείο myResults.txt που δημιουργήσαμε, του οποίου την δημιουργία την περιγράψαμε παραπάνω. Επίσης, στον φάκελο που βρίσκεται το εργαλείο trec_eval αντιγράφουμε και το αρχείο qrels.txt που μας δόθηκε και το

οποίο περιέχει τις σωστές απαντήσεις σε κάθε κείμενο. Έπειτα, τρέχοντας το command prompt με current directory το path του φακέλου που βρίσκεται το trec_eval, εκτελούμε την εντολή:

trec_eval.exe -q qrels.txt myResults.txt

```
trec_eval.exe -q qrels.txt myResults.txt
```

Αυτή η εντολή μας εμφανίζει τα ακόλουθα αποτελέσματα:

```
num_ret                 Q01     20
num_rel                 Q01     15
num_rel_ret             Q01     14
map                     Q01     0.6911
Rprec                   Q01     0.8000
bpref                   Q01     0.9333
recip_rank              Q01     0.5000
iprec_at_recall_0.00    Q01     0.8333
iprec_at_recall_0.10    Q01     0.8333
iprec_at_recall_0.20    Q01     0.8333
iprec_at_recall_0.30    Q01     0.8333
iprec_at_recall_0.40    Q01     0.8000
iprec_at_recall_0.50    Q01     0.8000
iprec_at_recall_0.60    Q01     0.8000
iprec_at_recall_0.70    Q01     0.8000
iprec_at_recall_0.80    Q01     0.8000
iprec_at_recall_0.90    Q01     0.7000
iprec_at_recall_1.00    Q01     0.0000
P_5                     Q01     0.8000
P_10                    Q01     0.8000
P_15                    Q01     0.8000
P_20                    Q01     0.7000
P_30                    Q01     0.4667
P_100                   Q01     0.1400
P_200                   Q01     0.0700
P_500                   Q01     0.0280
P_1000                  Q01     0.0140
num_ret                 Q02     20
num_rel                 Q02     11
num_rel_ret             Q02     9
map                     Q02     0.5494
Rprec                   Q02     0.5455
bpref                   Q02     0.8182
recip_rank              Q02     1.0000
iprec_at_recall_0.00    Q02     1.0000
iprec_at_recall_0.10    Q02     1.0000
iprec_at_recall_0.20    Q02     0.7500
iprec_at_recall_0.30    Q02     0.6667
iprec_at_recall_0.40    Q02     0.6250
iprec_at_recall_0.50    Q02     0.5455
iprec_at_recall_0.60    Q02     0.5385
iprec_at_recall_0.70    Q02     0.4737
```

```
iprec_at_recall_0.80    Q02     0.4737
iprec_at_recall_0.90    Q02     0.0000
iprec_at_recall_1.00    Q02     0.0000
P_5                     Q02     0.6000
P_10                    Q02     0.5000
P_15                    Q02     0.4667
P_20                    Q02     0.4500
P_30                    Q02     0.3000
P_100                   Q02     0.0900
P_200                   Q02     0.0450
P_500                   Q02     0.0180
P_1000                  Q02     0.0090
num_ret                 Q03     20
num_rel                 Q03     13
num_rel_ret             Q03     12
map                     Q03     0.5917
Rprec                   Q03     0.6154
bpref                   Q03     0.9231
recip_rank              Q03     1.0000
iprec_at_recall_0.00    Q03     1.0000
iprec_at_recall_0.10    Q03     1.0000
iprec_at_recall_0.20    Q03     0.6875
iprec_at_recall_0.30    Q03     0.6875
iprec_at_recall_0.40    Q03     0.6875
iprec_at_recall_0.50    Q03     0.6875
iprec_at_recall_0.60    Q03     0.6875
iprec_at_recall_0.70    Q03     0.6875
iprec_at_recall_0.80    Q03     0.6875
iprec_at_recall_0.90    Q03     0.6316
iprec_at_recall_1.00    Q03     0.0000
P_5                     Q03     0.4000
P_10                    Q03     0.5000
P_15                    Q03     0.6667
P_20                    Q03     0.6000
P_30                    Q03     0.4000
P_100                   Q03     0.1200
P_200                   Q03     0.0600
P_500                   Q03     0.0240
P_1000                  Q03     0.0120
num_ret                 Q04     20
num_rel                 Q04     13
num_rel_ret             Q04     10
map                     Q04     0.5209
Rprec                   Q04     0.5385
bpref                   Q04     0.7692
recip_rank              Q04     1.0000
```

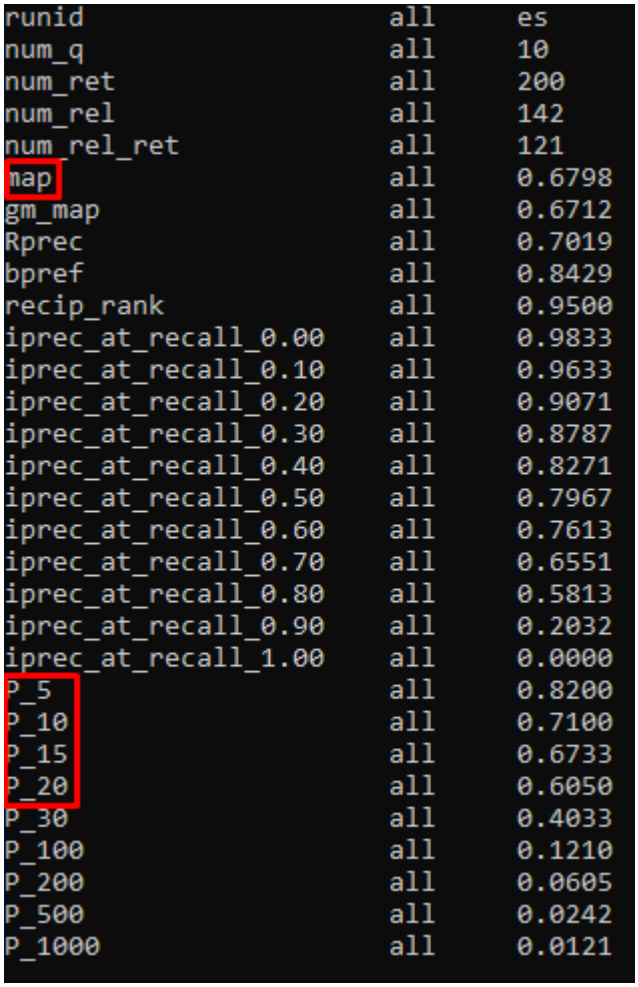
```
iprec_at_recall_0.00    Q04     1.0000
iprec_at_recall_0.10    Q04     0.8000
iprec_at_recall_0.20    Q04     0.8000
iprec_at_recall_0.30    Q04     0.8000
iprec_at_recall_0.40    Q04     0.6667
iprec_at_recall_0.50    Q04     0.6364
iprec_at_recall_0.60    Q04     0.5714
iprec_at_recall_0.70    Q04     0.5294
iprec_at_recall_0.80    Q04     0.0000
iprec_at_recall_0.90    Q04     0.0000
iprec_at_recall_1.00    Q04     0.0000
P_5                     Q04     0.8000
P_10                    Q04     0.6000
P_15                    Q04     0.5333
P_20                    Q04     0.5000
P_30                    Q04     0.3333
P_100                   Q04     0.1000
P_200                   Q04     0.0500
P_500                   Q04     0.0200
P_1000                  Q04     0.0100
num_ret                 Q05     20
num_rel                 Q05     15
num_rel_ret             Q05     14
map                     Q05     0.8361
Rprec                   Q05     0.8000
bpref                   Q05     0.9333
recip_rank              Q05     1.0000
iprec_at_recall_0.00    Q05     1.0000
iprec_at_recall_0.10    Q05     1.0000
iprec_at_recall_0.20    Q05     1.0000
iprec_at_recall_0.30    Q05     1.0000
iprec_at_recall_0.40    Q05     1.0000
iprec_at_recall_0.50    Q05     0.8889
iprec_at_recall_0.60    Q05     0.8571
iprec_at_recall_0.70    Q05     0.8571
iprec_at_recall_0.80    Q05     0.8571
iprec_at_recall_0.90    Q05     0.7000
iprec_at_recall_1.00    Q05     0.0000
P_5                     Q05     1.0000
P_10                    Q05     0.8000
P_15                    Q05     0.8000
P_20                    Q05     0.7000
P_30                    Q05     0.4667
P_100                   Q05     0.1400
P_200                   Q05     0.0700
P_500                   Q05     0.0280
```

```
P_1000                  Q05     0.0140
num_ret                 Q06     20
num_rel                 Q06     18
num_rel_ret             Q06     16
map                     Q06     0.8257
Rprec                   Q06     0.7778
bpref                   Q06     0.8889
recip_rank              Q06     1.0000
iprec_at_recall_0.00    Q06     1.0000
iprec_at_recall_0.10    Q06     1.0000
iprec_at_recall_0.20    Q06     1.0000
iprec_at_recall_0.30    Q06     1.0000
iprec_at_recall_0.40    Q06     1.0000
iprec_at_recall_0.50    Q06     0.9167
iprec_at_recall_0.60    Q06     0.9167
iprec_at_recall_0.70    Q06     0.8667
iprec_at_recall_0.80    Q06     0.8000
iprec_at_recall_0.90    Q06     0.0000
iprec_at_recall_1.00    Q06     0.0000
P_5                     Q06     1.0000
P_10                    Q06     0.9000
P_15                    Q06     0.8667
P_20                    Q06     0.8000
P_30                    Q06     0.5333
P_100                   Q06     0.1600
P_200                   Q06     0.0800
P_500                   Q06     0.0320
P_1000                  Q06     0.0160
num_ret                 Q07     20
num_rel                 Q07     15
num_rel_ret             Q07     12
map                     Q07     0.6696
Rprec                   Q07     0.7333
bpref                   Q07     0.8000
recip_rank              Q07     1.0000
iprec_at_recall_0.00    Q07     1.0000
iprec_at_recall_0.10    Q07     1.0000
iprec_at_recall_0.20    Q07     1.0000
iprec_at_recall_0.30    Q07     0.8000
iprec_at_recall_0.40    Q07     0.8000
iprec_at_recall_0.50    Q07     0.8000
iprec_at_recall_0.60    Q07     0.7692
iprec_at_recall_0.70    Q07     0.7500
iprec_at_recall_0.80    Q07     0.7500
iprec_at_recall_0.90    Q07     0.0000
iprec_at_recall_1.00    Q07     0.0000
```

```
P_5                     Q07     0.8000
P_10                    Q07     0.8000
P_15                    Q07     0.7333
P_20                    Q07     0.6000
P_30                    Q07     0.4000
P_100                   Q07     0.1200
P_200                   Q07     0.0600
P_500                   Q07     0.0240
P_1000                  Q07     0.0120
num_ret                 Q08     20
num_rel                 Q08     13
num_rel_ret             Q08     11
map                     Q08     0.6458
Rprec                   Q08     0.6923
bpref                   Q08     0.8462
recip_rank              Q08     1.0000
iprec_at_recall_0.00    Q08     1.0000
iprec_at_recall_0.10    Q08     1.0000
iprec_at_recall_0.20    Q08     1.0000
iprec_at_recall_0.30    Q08     1.0000
iprec_at_recall_0.40    Q08     0.6923
iprec_at_recall_0.50    Q08     0.6923
iprec_at_recall_0.60    Q08     0.6923
iprec_at_recall_0.70    Q08     0.6923
iprec_at_recall_0.80    Q08     0.5500
iprec_at_recall_0.90    Q08     0.0000
iprec_at_recall_1.00    Q08     0.0000
P_5                     Q08     0.8000
P_10                    Q08     0.6000
P_15                    Q08     0.6000
P_20                    Q08     0.5500
P_30                    Q08     0.3667
P_100                   Q08     0.1100
P_200                   Q08     0.0550
P_500                   Q08     0.0220
P_1000                  Q08     0.0110
num_ret                 Q09     20
num_rel                 Q09     20
num_rel_ret             Q09     17
map                     Q09     0.8165
Rprec                   Q09     0.8500
bpref                   Q09     0.8500
recip_rank              Q09     1.0000
iprec_at_recall_0.00    Q09     1.0000
```

```
iprec_at_recall_0.10    Q09     1.0000
iprec_at_recall_0.20    Q09     1.0000
iprec_at_recall_0.30    Q09     1.0000
iprec_at_recall_0.40    Q09     1.0000
iprec_at_recall_0.50    Q09     1.0000
iprec_at_recall_0.60    Q09     0.9231
iprec_at_recall_0.70    Q09     0.8947
iprec_at_recall_0.80    Q09     0.8947
iprec_at_recall_0.90    Q09     0.0000
iprec_at_recall_1.00    Q09     0.0000
P_5                     Q09     1.0000
P_10                    Q09     1.0000
P_15                    Q09     0.8667
P_20                    Q09     0.8500
P_30                    Q09     0.5667
P_100                   Q09     0.1700
P_200                   Q09     0.0850
P_500                   Q09     0.0340
P_1000                  Q09     0.0170
num_ret                 Q10     20
num_rel                 Q10     9
num_rel_ret             Q10     6
map                     Q10     0.6508
Rprec                   Q10     0.6667
bpref                   Q10     0.6667
recip_rank              Q10     1.0000
iprec_at_recall_0.00    Q10     1.0000
iprec_at_recall_0.10    Q10     1.0000
iprec_at_recall_0.20    Q10     1.0000
iprec_at_recall_0.30    Q10     1.0000
iprec_at_recall_0.40    Q10     1.0000
iprec_at_recall_0.50    Q10     1.0000
iprec_at_recall_0.60    Q10     0.8571
iprec_at_recall_0.70    Q10     0.0000
iprec_at_recall_0.80    Q10     0.0000
iprec_at_recall_0.90    Q10     0.0000
iprec_at_recall_1.00    Q10     0.0000
P_5                     Q10     1.0000
P_10                    Q10     0.6000
P_15                    Q10     0.4000
P_20                    Q10     0.3000
P_30                    Q10     0.2000
P_100                   Q10     0.0600
P_200                   Q10     0.0300
P_500                   Q10     0.0120
P_1000                  Q10     0.0060
```

Και το βασικότερο που μας ενδιαφέρει περισσότερο:



```
runid                   all     es
num_q                   all     10
num_ret                 all     200
num_rel                 all     142
num_rel_ret             all     121
map                     all     0.6798
gm_map                  all     0.6712
Rprec                   all     0.7019
bpref                   all     0.8429
recip_rank              all     0.9500
iprec_at_recall_0.00    all     0.9833
iprec_at_recall_0.10    all     0.9633
iprec_at_recall_0.20    all     0.9071
iprec_at_recall_0.30    all     0.8787
iprec_at_recall_0.40    all     0.8271
iprec_at_recall_0.50    all     0.7967
iprec_at_recall_0.60    all     0.7613
iprec_at_recall_0.70    all     0.6551
iprec_at_recall_0.80    all     0.5813
iprec_at_recall_0.90    all     0.2032
iprec_at_recall_1.00    all     0.0000
P_5                     all     0.8200
P_10                    all     0.7100
P_15                    all     0.6733
P_20                    all     0.6050
P_30                    all     0.4033
P_100                   all     0.1210
P_200                   all     0.0605
P_500                   all     0.0242
P_1000                  all     0.0121
```

Όπου έχουμε το συνολικό MAP και το avgPre@k,(μέση ακρίβεια στα k-πρώτα ανακτηθέντα κείμενα) με k=5, 10, 15, 20(P_5, P_10, P_15, P_20).

ΟΡΙΣΜΕΝΕΣ ΕΠΙΠΛΕΟΝ ΔΙΕΥΚΡΙΝΗΣΕΙΣ

Στον κώδικα που παραδίδεται υπάρχουν κάποιες επιπλέον μέθοδοι οι οποίες κάνουν ίδιες διαδικασίες απλά με μία μικρή διαφορά. Αυτές οι μέθοδοι είναι οι process_xml_create_json(), load_collection(), load_collection_from_files() και η bulk_load_from_files(). Η πρώτη κάνει τις ίδιες ενέργειες με την process_xml() εκτός από το γεγονός ότι αντί να γράφει τα json strings σε λίστα, γράφει το κάθε json string σε νέο ξεχωριστό αρχείο json το οποίο αποθηκεύει στον φάκελο Json files του project. Η δεύτερη, έχει παρόμοια λειτουργικότητα με την bulk_load() μόνο που αντί να χρησιμοποιεί το Bulk API διαβάζει ένα-ένα τα json strings από την λίστα και τα φορτώνει στον index με αποτέλεσμα να είναι πολύ πιο αργή από ότι με την χρήση Bulk API η φόρτωση των json strings στον index. Παρόλα αυτά η

μέθοδος αυτή που χρησιμοποιεί αυτό τον τρόπο φόρτωσης της συλλογής στον index είναι πιο σταθερή σε σχέση με όταν χρησιμοποιείται η bulk_load(), δηλαδή πάντα δίνει ελάχιστα καλύτερα αποτελέσματα και πάντα το trec_eval επιστρέφει ίδια ποσοστά. Ενώ με την χρήση του Bulk API κάθε φορά που τρέχουμε το project μπορεί στο τέλος όταν θα κάνουμε την αξιολόγηση με το trec_eval να μας επιστραφούν διαφορετικά ποσοστά χωρίς να έχουμε κάνει καμία τροποποίηση στον κώδικά μας. Βέβαια, λόγω της ταχύτητας που έχουμε με το Bulk API προτιμάμε την bulk_load() ας είναι λίγο ασταθής με ελάχιστα χειρότερα αποτελέσματα. Η Τρίτη μέθοδος έχει την ίδια λειτουργικότητα με την load_collection() με την διαφορά ότι δεν διαβάζουμε τα json strings από μία λίστα αλλά διαβάζουμε ένα-ένα τα json αρχεία που δημιουργήθηκαν από την μέθοδο process_xml_create_json() και καθένα που διαβάζουμε το φορτώνουμε στον index. Τέλος, η μέθοδος bulk_load_from_files() έχει την ίδια λειτουργικότητα με την μέθοδο bulk_load() με την μόνη διαφορά ότι δεν διαβάζουμε τα json strings από μία λίστα αλλά διαβάζουμε ένα-ένα τα json αρχεία που δημιουργήθηκαν από την μέθοδο process_xml_create_json().