Συστήματα ανάκτησης πληροφοριών

Προγραμματιστική εργασία, φάση 3

Δημήτρης Μπάστας, 3130139
Φίλιππος Δουραχαλής, 3170045

Η εκπόνηση της εργασίας πραγματοποιήθηκε στο περιβάλλον Intellij IDEA.

Οι βιβλιοθήκες που θα χρειαστούμε είναι οι ίδιες που χρησιμοποιήθηκαν στην πρώτη φάση του project. Το φόρτωμά τους γίνεται με τον ίδιο τρόπο που έχει περιγραφεί στις προηγούμενες αναφορές, δηλαδή μέσα από τις ρυθμίσεις του Intellij επιλέγουμε

file > *Project Structure* > *Libraries* > *Add* > *Java*

και στη συνέχεια επιλέγουμε τα κατάλληλα αρχεία jar.

Όπως και προηγουμένως, τροποποιούμε κατάλληλα τις σταθερές TREC_EVAL_PATH και QUERIES_PATH για να δείχνουν στον φάκελο του trec_eval και στο αρχείο qrels.text της συλλογής CACM.

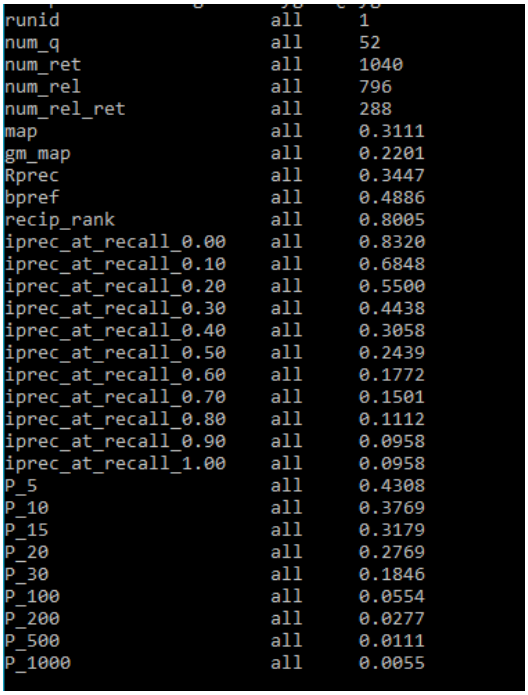
BM25Similarity

Για το συγκεκριμένο σκέλος, αρκεί να αλλάξουμε το similarity από ClassicSimilarity σε BM25Similarity. Στη συνέχεια η εκτέλεση του κώδικα πραγματοποιείται όπως ακριβώς και στην πρώτη φάση της εργασίας.

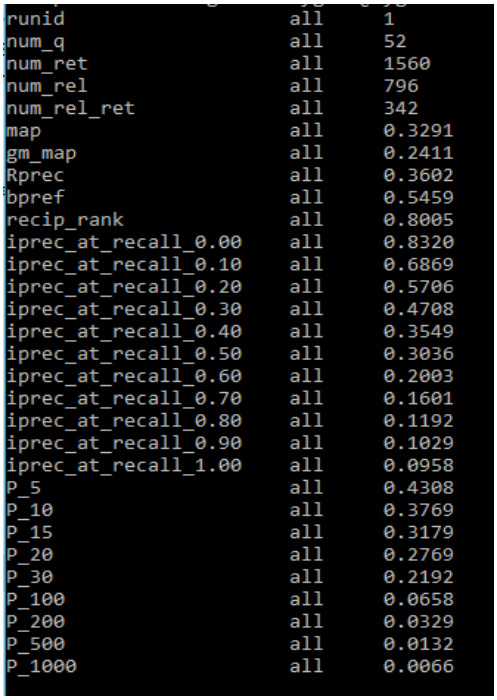
Το αποτέλεσμα του trec eval για τις τιμές k = 20, 30, 50 φαίνεται στον παρακάτω πίνακα. Τα αντίστοιχα πλήρη outputs βρίσκονται στα αρχεία map20.txt, map30.txt και map50.txt

k	20	30	50
MAP	0.3111	0.3291	0.3455
P_5	0.4308	0.4308	0.4308
P_10	0.3769	0.3769	0.3769
P_15	0.3179	0.3179	0.3179
P_20	0.2769	0.2769	0.2769

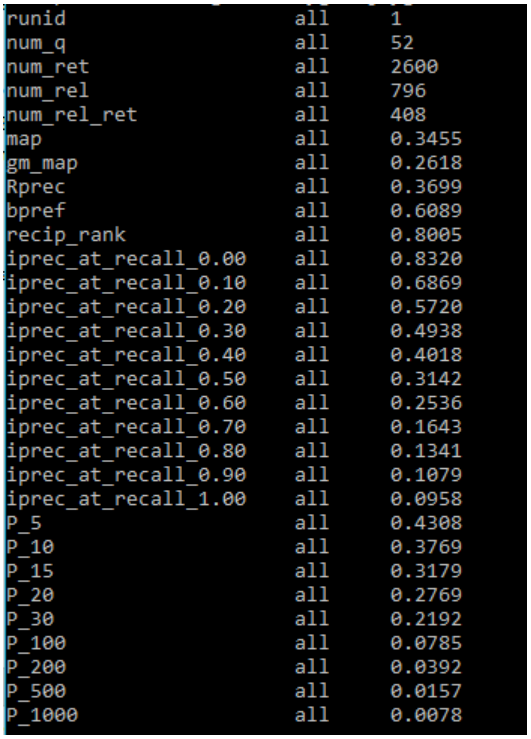
Όπως παρατηρούμε, τα αποτελέσματα είναι ελαφρώς αλλά σταθερά ανεβασμένα σε σχέση με τη ClassicSimilarity σε όλα τα map για k επιστροφές. Στις επόμενες εικόνες φαίνεται το output του trec_eval σε σειρά (για k=20, 30 και 50):



```
runid                   all     1
num_q                   all     52
num_ret                 all     1040
num_rel                 all     796
num_rel_ret             all     288
map                     all     0.3111
gm_map                  all     0.2201
Rprec                   all     0.3447
bpref                   all     0.4886
recip_rank              all     0.8005
iprec_at_recall_0.00    all     0.8320
iprec_at_recall_0.10    all     0.6848
iprec_at_recall_0.20    all     0.5500
iprec_at_recall_0.30    all     0.4438
iprec_at_recall_0.40    all     0.3058
iprec_at_recall_0.50    all     0.2439
iprec_at_recall_0.60    all     0.1772
iprec_at_recall_0.70    all     0.1501
iprec_at_recall_0.80    all     0.1112
iprec_at_recall_0.90    all     0.0958
iprec_at_recall_1.00    all     0.0958
P_5                     all     0.4308
P_10                    all     0.3769
P_15                    all     0.3179
P_20                    all     0.2769
P_30                    all     0.1846
P_100                   all     0.0554
P_200                   all     0.0277
P_500                   all     0.0111
P_1000                  all     0.0055
```



```
runid                   all     1
num_q                   all     52
num_ret                 all     1560
num_rel                 all     796
num_rel_ret             all     342
map                     all     0.3291
gm_map                  all     0.2411
Rprec                   all     0.3602
bpref                   all     0.5459
recip_rank              all     0.8005
iprec_at_recall_0.00    all     0.8320
iprec_at_recall_0.10    all     0.6869
iprec_at_recall_0.20    all     0.5706
iprec_at_recall_0.30    all     0.4708
iprec_at_recall_0.40    all     0.3549
iprec_at_recall_0.50    all     0.3036
iprec_at_recall_0.60    all     0.2003
iprec_at_recall_0.70    all     0.1601
iprec_at_recall_0.80    all     0.1192
iprec_at_recall_0.90    all     0.1029
iprec_at_recall_1.00    all     0.0958
P_5                     all     0.4308
P_10                    all     0.3769
P_15                    all     0.3179
P_20                    all     0.2769
P_30                    all     0.2192
P_100                   all     0.0658
P_200                   all     0.0329
P_500                   all     0.0132
P_1000                  all     0.0066
```



```
runid                   all     1
num_q                   all     52
num_ret                 all     2600
num_rel                 all     796
num_rel_ret             all     408
map                     all     0.3455
gm_map                  all     0.2618
Rprec                   all     0.3699
bpref                   all     0.6089
recip_rank              all     0.8005
iprec_at_recall_0.00    all     0.8320
iprec_at_recall_0.10    all     0.6869
iprec_at_recall_0.20    all     0.5720
iprec_at_recall_0.30    all     0.4938
iprec_at_recall_0.40    all     0.4018
iprec_at_recall_0.50    all     0.3142
iprec_at_recall_0.60    all     0.2536
iprec_at_recall_0.70    all     0.1643
iprec_at_recall_0.80    all     0.1341
iprec_at_recall_0.90    all     0.1079
iprec_at_recall_1.00    all     0.0958
P_5                     all     0.4308
P_10                    all     0.3769
P_15                    all     0.3179
P_20                    all     0.2769
P_30                    all     0.2192
P_100                   all     0.0785
P_200                   all     0.0392
P_500                   all     0.0157
P_1000                  all     0.0078
```

LMJelinekMercerSimilarity

Στο συγκεκριμένο κομμάτι της εργασίας αλλάξαμε ξανά την κλάση της ομοιότητας που χρησιμοποιεί η Lucene δίνοντας την LMJelinekMercerSimilarity. Προκειμένου να χρησιμοποιηθεί η συγκεκριμένη κλάση πρέπει να δώσουμε στον κατασκευαστή της μια τιμή για το λ μεταξύ 0 και 1, που προσδιορίζει το βάρος που δίνεται στις πιθανότητες ένας όρος να βρίσκεται στη συλλογή ή σε ένα κείμενο αντίστοιχα. Για να βρούμε την τιμή του λ που δίνει καλύτερα αποτελέσματα ελέγχουμε διαδοχικά τιμές και παρατηρούμε το output του trec_eval για καθεμία από αυτές. Η στρατηγική που ακολουθήσαμε είναι να ξεκινήσουμε με ίσα βάρη και για τις δύο πιθανότητες (λ=0.5).

Στη συνέχεια εξετάζουμε τις τιμές 0.2 και 0.8 ώστε να δούμε σε ποιο υποδιάστημα παίρνουμε υψηλότερο ποσοστό συναφών κειμένων. Από εκεί μπορεί να δοκιμάσουμε και άλλες τιμές που ανήκουν στο διάστημα (π.χ. 0.4 και 0.7) για να εξετάσουμε πως μεταβάλλεται το map και έτσι να μπορέσουμε μετά να κάνουμε μια καλύτερη εκτίμηση για την τιμή του λ. Επαναλαμβάνουμε την παραπάνω διαδικασία για όλες τις τιμές του k που μας ενδιαφέρουν.

Παρακάτω παρουσιάζουμε τους συνοπτικούς συγκριτικούς πίνακες για κάθε λ και για κάθε τιμή του k για την οποία έτρεξε το πρόγραμμα.

(Σημείωση: Οι κενές γραμμές σε έναν πίνακα σημαίνει πως για το συγκεκριμένο k δεν δοκιμάστηκε η τιμή του λ που αναφέρεται καθώς το αποτέλεσμα της δεν θα είχε κάποια ουσιαστική επίδραση, δηλαδή την είχαμε αποκλείσει ήδη ως μη-βέλτιστη).

Λ=0.2

k	20	30	50
MAP	0.2581	0.2755	0.2911
P_5	0.4000	0.4000	0.4000
P_10	0.3250	0.3250	0.3250
P_15	0.2808	0.2808	0.2808
P_20	0.2442	0.2442	0.2442

Λ=0.4

k	20	30	50
MAP	0.2719		0.3048
P_5	0.4000		0.4000
P_10	0.3442		0.3442
P_15	0.2846		0.2846
P_20	0.2529		0.2529

Λ=0.5

k	20	30	50
MAP	0.2275	0.2961	0.3108
P_5	0.4038	0.4038	0.4038
P_10	0.3423	0.3423	0.3423
P_15	0.2872	0.2872	0.2872
P_20	0.2558	0.2558	0.2558

Λ=0.7

k	20	30	50
MAP	0.2856	0.3029	0.3192
P_5	0.4115	0.4115	0.4115
P_10	0.3423	0.3423	0.3423
P_15	0.2923	0.2923	0.2923
P_20	0.2577	0.2577	0.2577

Λ=0.75

k	20	30	50
MAP		0.3025	0.3195
P_5		0.4038	0.4038
P_10		0.3481	0.3481
P_15		0.2923	0.2923
P_20		0.2538	0.2538

Λ=0.8

k	20	30	50
MAP	0.2815	0.3002	0.3166
P_5	0.4038	0.4038	0.4038
P_10	0.3423	0.3423	0.3423
P_15	0.2974	0.2974	0.2974
P_20	0.2558	0.2558	0.2558

Παρατηρούμε πως δίνοντας περισσότερο βάρος στην πιθανότητα ο όρος να περιέχεται σε ολόκληρη την συλλογή κειμένων ($\lambda > 0.5$), τα αποτελέσματα είναι καλύτερα ασχέτως k. Για να

βρούμε όμως το ιδανικό λ αφού περιοριστούμε στο υποδιάστημα $(0.5, 1)$ ελέγξαμε διαδοχικά τις τιμές 0.8, 0.9 και 0.7. Όπως φαίνεται και ανωτέρω, τα αποτελέσματα ήταν πάντα καλύτερα για λ=0.7, ενώ στην περίπτωση του k=50 η υψηλότερη τιμή του map παρατηρήθηκε για λ=0.75. Όπως ήταν αναμενόμενο όλα τα P5-20 μένουν απαράλλακτα για κάθε λ μεταξύ των k.

Συνοψίζοντας όλα τα ανωτέρω, για k=20 και k=30 καλύτερα αποτελέσματα παρατηρήθηκαν για λ=0.7, ενώ για k=50 η ιδανική τιμή για το λ φάνηκε να ήταν η λ=0.75. Τα πλήρη outputs για τις συγκεκριμένες τιμές ως συνήθως βρίσκονται επίσης στα αρχεία map20_07.txt, map30_07.txt και map50_075.txt