Συστήματα Ανάκτησης – Φάση 2

ΑΝΑΦΟΡΑ ΠΑΡΑΔΩΣΗΣ

(**Όνομα:** Αναστάσιος Παπαπαναγιώτου, **ΑΜ:** 3200143, **Email:** p3200143@aueb.gr)

(**Όνομα:** Φοίβος Παπαθανασίου, **ΑΜ:** 3200138, **Email:** p3200138@aueb.gr)

Βήματα 1, 2

Η μέθοδος **buildIndex()** της κλάσης Index δέχεται ως είσοδο τις τιμές των παραμέτρων (λ, b, k1) και διαλέγει ανάλογα ποιο similarity να χρησιμοποιηθεί. Μετά διαβάζει το αρχείο που περιέχει τα κείμενα (documents.txt) και μετατρέπει κάθε κείμενο σε ένα document με δύο fields – body και id, το id το χρειάζεται ώστε να μπορέσουμε να κατασκευάσουμε τα αρχεία για το trec_eval. Για κάθε συνδυασμό παραμέτρων δημιουργείται και ένα αρχείο με τα αποτελέσματα.

Βήμα 3

Η μέθοδος getDocsForQueries(...) της κλάσης Index επιστρέφει ένα ArrayList που περιέχει για κάθε query από ένα queries.txt με το δοθέντο format τα top k documents που έγιναν retrieve.

Πιο συγκεκριμένα, αρχικά κατασκευάζουμε ένα αντικείμενο της κλάσης IndexSearcher σεττάρωντας το similarity σε BM25Similarity ή LMJelinekMercerSimilarity, ανάλογα με τις παραμέτρους που έχουν δοθεί ως είσοδος, κατασκευάζουμε επίσης ένα αντικείμενο της κλάσης QueryParser μόνο για το field “body” και του περνάμε τον EnglishAnalyzer.

Στη συνέχεια κάνουμε extract τα queries από το queries.txt με χρήση της βοηθητικής συνάρτησης readFile της κλάσης Utils και για κάθε query κάνουμε retrieve τα top k documents τα οποία είναι αντικείμενα της κλάσης ScoreDoc.

Εκτός όμως από το score τους χρειαζόμαστε και το αντίστοιχο id τους προκειμένου να κατασκευάσουμε το κάθε αρχείο με τα αποτελέσματα. Για το λόγο αυτό στη συνέχεια για κάθε ένα από αυτά τα queries και για καθένα από τα hits τους παίρνουμε το αντίστοιχο Document instance (και όχι ScoreDoc instance), και με χρήση μίας βοηθητικής κλάσης **QueriedDoc** μαζεύουμε και το Document instance και το score ώστε να μπορέσουμε μετά να παράξουμε το αναγκαίο αρχείο.

Βήμα 4

Λαμβάνοντας το ArrayList από τη μέθοδο getDocsForQueries(...) της κλάσης Index, κατασκευάζουμε ένα αντικείμενο της κλάσης Evaluation και χρησιμοποιούμε τη μέθοδο storeResults(...) για τη κατασκευή του αρχείου bm25_i_k1.txt ή του lmj_λ.txt. Η τιμή του i είναι το index του πίνακα {0, 0.25, 0.5, 0.75, 1} για τις πιθανές τιμές του b, το k1 είναι οι πιθανές τιμές του k1, {0, 1.2, 1.5, 1.8, 2, 3}, πολλαπλασιασμένες με το 10. Ομοίως η τιμή του λ στο αρχείο είναι οι πιθανές τιμές του λ, {0, 0.1, 0.3, 0.5, 0.7, 0.9, 1}, πολλαπλασιασμένες με το 10.

Τα αρχεία αυτά επειδή είναι πολλά σε πλήθος δημιουργήσαμε ένα bash script για να γίνει πιο γρήγορα το οποίο εκτελεί το command του trec_eval για κάθε αρχείο και όλα τα αποτελέσματα του trec_eval για τα αρχεία του bm25 για το MAP τα αποθηκεύει σε ένα αρχείο (bm25_map.txt), για το Precision σε ένα άλλο αρχείο (bm25_prec.txt). Με τον ίδιο τρόπο δημιουργούνται τα αρχεία και για το LMJ.

Τοποθετούμε στο ίδιο directory όπου κάναμε extract το trec_eval.zip τα δύο αρχεία qrels.txt, όλα τα αρχεία για το bm25 και LMJ, και το bash script. Τρέχουμε το bash script και στη συνέχεια βλέπουμε τα αρχεία που δημιουργήθηκαν από αυτό.

Ανάλυση Παραμέτρων

Αρχικά, για b=0, και το MAP και το Precision (για όλες τις τιμές) είναι πολύ χαμηλότερα για μεγαλύτερες τιμές επομένως αποκλείουμε να είναι 0. Μετά παρατηρούμε ότι μικρότερες τιμές στο b (>0) έχουν καλύτερα αποτελέσματα για μικρές τιμές του Precision π.χ. το (P.5) όμως έχουν χειρότερα αποτελέσματα για μεγάλες τιμές του Precision. Επομένως αν θέλαμε ένα σύστημα να ανακτά λίγα documents, θα προτιμούσαμε μικρότερο b όπως b=1.2. Αν το σύστημα θέλαμε να ανακτά πολλά docs, τότε θα διαλέγαμε μια μεγαλύτερη τιμή για το b, όπως b=3.0. Γενικότερα όμως, υπάρχει μικρή απόκλιση (~0.02) μεταξύ των τιμών του b.

Τα αποτελέσματα για το MAP είναι επίσης καλύτερα για μεγαλύτερες τιμές του b, όμως δεν υπάρχει μεγάλη απόκλιση και για κάποιες μικρότερες τιμές.

Καθώς το σύνολο των συνδυασμών είναι μεγάλο, δημιουργήσαμε ένα python script που διαβάζει τα αποτελέσματα από το trec_eval και κρατάει όλους τους συνδυασμούς που βρίσκονται στο X-th percentile. Συγκεκριμένα, για καθένα από τα αρχεία bm25_prec.txt, bm25_map.txt, lmj_map.txt, lmj_prec.txt διαβάζει τα αποτελέσματα και αποθηκεύει σε μια λίστα το αποτέλεσμα για όλα τα queries (all). Μετά για τα αρχεία bm25_prec και bm25_map (ομοίως και για το lmj) υπολογίζει το X-th percentile για το precision, το X-th percentile για το map και βρίσκει ποιοι συνδυασμοί βρίσκονται στο X-th percentile. Τέλος βρίσκει το intersection μεταξύ των συνδυασμών του map και του prec. Το X θέλουμε να είναι ο μεγαλύτερος αριθμός τέτοιος ώστε το intersection να είναι μη κενό σύνολο.

Βρήκαμε ότι για το LMJ υπάρχει μόνο μια τιμή για το 95-th percentile η οποία είναι λ=0.9 και για το BM25 υπάρχουν 3 συνδυασμοί για το 85-th percentile, οι οποίοι είναι: {b=0, k1=3}, {b=0.25, k1=3}, {b=0.5, k1=2}. Και ο πίνακας με τα αποτελέσματα για αυτές τις τιμές είναι:

- LMJ με λ=0.9:

Q	MAP	Precision@5	Precision@10	Precision@15	Precision@20
Q01	0.7294	0.8	0.9	0.8	0.65
Q02	0.2833	0.6	0.4	0.2667	0.2
Q03	0.6217	1.0	0.6	0.4667	0.4
Q04	0.0636	0.0	0.3	0.2	0.15
Q05	0.2583	0.2	0.3	0.2667	0.35
Q06	0.0636	0.0	0.2	0.2	0.2
Q07	0.1755	0.2	0.3	0.2	0.15
Q08	0.8449	1.0	1	0.7333	0.55
Q09	0.2173	0.6	0.3	0.2667	0.25
Q10	0.2540	0.4	0.4	0.2667	0.2
all	0.3512	0.48	0.47	0.3667	0.31

- BM25 με b=0, k1=3

Q	MAP	Precision@5	Precision@10	Precision@15	Precision@20
Q01	0.7779	0.8	0.8	0.8	0.7
Q02	0.3017	0.6	0.3	0.2667	0.25
Q03	0.6184	0.8	0.6	0.5333	0.4
Q04	0.0428	0.0	0.1	0.1333	0.15
Q05	0.2628	0.4	0.3	0.3333	0.35
Q06	0.1100	0.2	0.3	0.2	0.2
Q07	0.2470	0.2	0.2	0.2667	0.3
Q08	0.8915	1.0	1	0.8	0.6
Q09	0.2025	0.6	0.4	0.2667	0.25
Q10	0.2861	0.6	0.4	0.2667	0.2
all	0.3741	0.52	0.44	0.3867	0.34

- BM25 με b=0.25, k1=3

Q	MAP	Precision@5	Precision@10	Precision@15	Precision@20
Q01	0.7779	0.8	0.8	0.8	0.7
Q02	0.2970	0.6	0.3	0.2667	0.25
Q03	0.6177	0.8	0.6	0.5333	0.4
Q04	0.0458	0.0	0.1	0.2	0.15
Q05	0.2628	0.4	0.3	0.3333	0.35
Q06	0.1008	0.2	0.2	0.2	0.2
Q07	0.2507	0.2	0.2	0.2667	0.3
Q08	0.8892	1.0	1	0.8	0.6
Q09	0.1991	0.6	0.4	0.2667	0.25
Q10	0.2861	0.6	0.4	0.2667	0.2
all	0.3727	0.52	0.43	0.3933	0.34

- BM25 με b=0.5, k1=2

Q	MAP	Precision@5	Precision@10	Precision@15	Precision@20
Q01	0.7927	0.8	0.9	0.8	0.75
Q02	0.2875	0.6	0.3	0.2	0.2
Q03	0.6054	0.8	0.6	0.5333	0.4
Q04	0.0621	0.2	0.2	0.2	0.15
Q05	0.2503	0.4	0.3	0.2667	0.35
Q06	0.0821	0.2	0.1	0.2	0.15
Q07	0.2516	0.2	0.2	0.2	0.25
Q08	0.8837	1.0	1	0.8	0.6
Q09	0.2111	0.6	0.4	0.3333	0.25
Q10	0.2917	0.6	0.4	0.2667	0.2
all	0.3718	0.54	0.44	0.38	0.325

Παρατηρούμε η καλύτερη περίπτωση είναι η BM25 με b=0, k1=3 (αν και οι άλλοι συνδυασμοί του BM25 είναι σχεδόν ίδιοι). Σε όλα τα metrics τα πάει καλύτερα από τα αποτελέσματα της 1ης φάσης.

Πηγές: eclass, Lucene's documentation

Directory map: τα αρχεία analysis.py που έγινε η ανάλυση και το bash script βρίσκονται στο directory: "src/main/resources/output"