Πανεπιστήμιο Κρήτης
Τμήμα Επιστήμης Υπολογιστών
ΗΥ463 Συστήματα Ανάκτησης Πληροφοριών
Εξάμηνο: Άνοιξη 2017

Γραπτή Αναφορά Έργου

Στοιχεία Φοιτητών

Μέλος	1ο
Ονοματεπώνυμο	Αναστασάς Αναστάσιος
ΑΜ	3166
Email	csd3166@csd.uoc.gr

Μέλος	2ο
Ονοματεπώνυμο	Γιακουμής Γιώργος
ΑΜ	3157
Email	csd3157@csd.uoc.gr

Πίνακας Περιεχομένων

1 Εισαγωγή

Στην 2η φάση του project υλοποιήσαμε κάποια μέτρα τα οποία θα μας βοηθήσουν να αξιολογήσουμε τον indexer και τον searcher που υλοποιήσαμε στην 1η φάση. Επίσης διορθώσαμε κάποια λάθη της 1ης φάσης και κάναμε κάποιες βελτιώσεις που μείωσαν κατα πολύ τους χρόνους εκτέλεσης τόσο του indexer όσο και του searcher.

2 < Υλοποίηση >

2.1 < Αναλυτική Περιγραφή >

Αρχικά για κάθε ένα από τα topics που έχουμε, διαβάζουμε από το qrels τα έγγραφα για τα οποία έχουμε κάποια πληροφορία για το κατά πόσο σχετικά είναι με το topic. Στην συνέχεια διαβάζουμε τα result που πήραμε για το συγκεκριμένο topic και με βάσει αυτά τα δεδομένα υπολογίζουμε τις μετρικές bpref [7], Avep'[7] και NDCG' [4]. Έπειτα γράφουμε τα δεδομένα αυτά στο αρχείο. Υπάρχουν topics όπου οι μετρικές είναι 0, ο λόγος είναι ότι κανένα από τα 1000 αποτελέσματα που βρήκαμε για αυτό το topic δεν ήταν σχετικό. Αυτό συμβαίνει μάλλον επειδή κάναμε indexing ένα υποσύνολο τις συλλογής (5gb από moodle) και όχι ολόκληρη, έτσι τα σχετικά με το topic έγγραφα απο το qrels.txt δεν βρίσκονταν καν μέσα στη συλλογή που ευρετηριάσαμε.

2.2 < Τρόπος εκτέλεσης >

Πριν εκτελέσουμε το jar πρέπει να εξασφαλίσουμε την ύπαρξη δύο αρχείων στο ίδιο path όπου βρίσκεται και το jar. Αυτά τα αρχεία είναι:

- qrels.txt: το οποίο περιέχει την σχετικότητα τον topics με κάποια έγγραφα.
- results.txt: τα αποτελέσματα του indexer για τα topics

Στην συνέχεια τρέχουμε το πρόγραμμα με την εντολη:

- java –jar ./Biomedical_IRS_Evaluation.jar

Τα αποτελέσματα αποθηκεύονται στο path που βρίσκεται το .jar με όνομα eval_results.txt

2.3 < Βελτιώσεις >

2.3.1 < Indexer >

Αρχικά πρέπει να αναφέρουμε ότι στην 1η φάση είχαμε κάνει εκ παραδρομής ένα λάθος το οποίο μας καθυστερούσε το σύστημα και μας αύξανε την μνήμη άσκοπα. Αντί να κάνουμε set σε ένα νέο value για κάποιο index κάναμε add, η οποία όχι απλά δεν άλλαζε το value στο συγκεκριμένο index αλλά έβαζε ένα νέο entry σε εκείνο το index και έκανε shift όλα τα επόμενα.

Επίσης ενώσαμε κάποια κομμάτια κώδικα που έκαναν το ίδιο iterate ώστε να μειώσουμε και άλλο τον χρόνο indexing. Με αυτές τις αλλαγές καταφέραμε και τρέξαμε την συλλογή των 5 GB που υπήρχε στο moodle σε χρόνο περίπου 30-35 λεπτά.

2.3.2 < Searcher >

Παρατηρήσαμε ότι υπερβολικά πολύ χρόνο χρειάζεται για να ανοίξει ένα αρχείο nxml και να διαβάσει κάποια tags. Επομένως κατά τον υπολογισμό των αποτελεσμάτων των topics, αντί να ανοίγουμε πάλι τα nxml ώστε να διαβάσουμε το PMCID, το παίρνουμε από το file path δεδομένου ότι το PMCID είναι το

όνομα του εκάστοτε αρχείου. Επίσης αφού είδαμε ότι εμφανίζουμε κατά την αναζήτηση 1000 αποτελέσματα, ήταν άσκοπο να υπολογίζουμε τα snippet για όλα τα αποτελέσματα. Αυτές οι αλλαγές μείωσαν κατα πολύ τον χρόνο της αναζήτησης, για παράδειγμα το search των topics που στην προηγούμενη φάση έκανε 7 ώρες και 30 λεπτά, τώρα έκανε 83 δευτερόλεπτα.

3 Μετρήσεις

- *Νέοι χρόνοι από την φάση Α*

Indexing:

Collection: οι φάκελοι από την συλλογή που κατεβάσαμε από το moodle (μικρή)
00 … 30 (Ολόκληρη)

Collection analyzed σε: 1.944 seconds

Distinct words: 1.781.760

Store Vocabulary: 0.5 seconds
TFS & Norms calculated: 25 seconds
Store Posting: 28 seconds
Store Documents file: 0.3 seconds

Topics search:

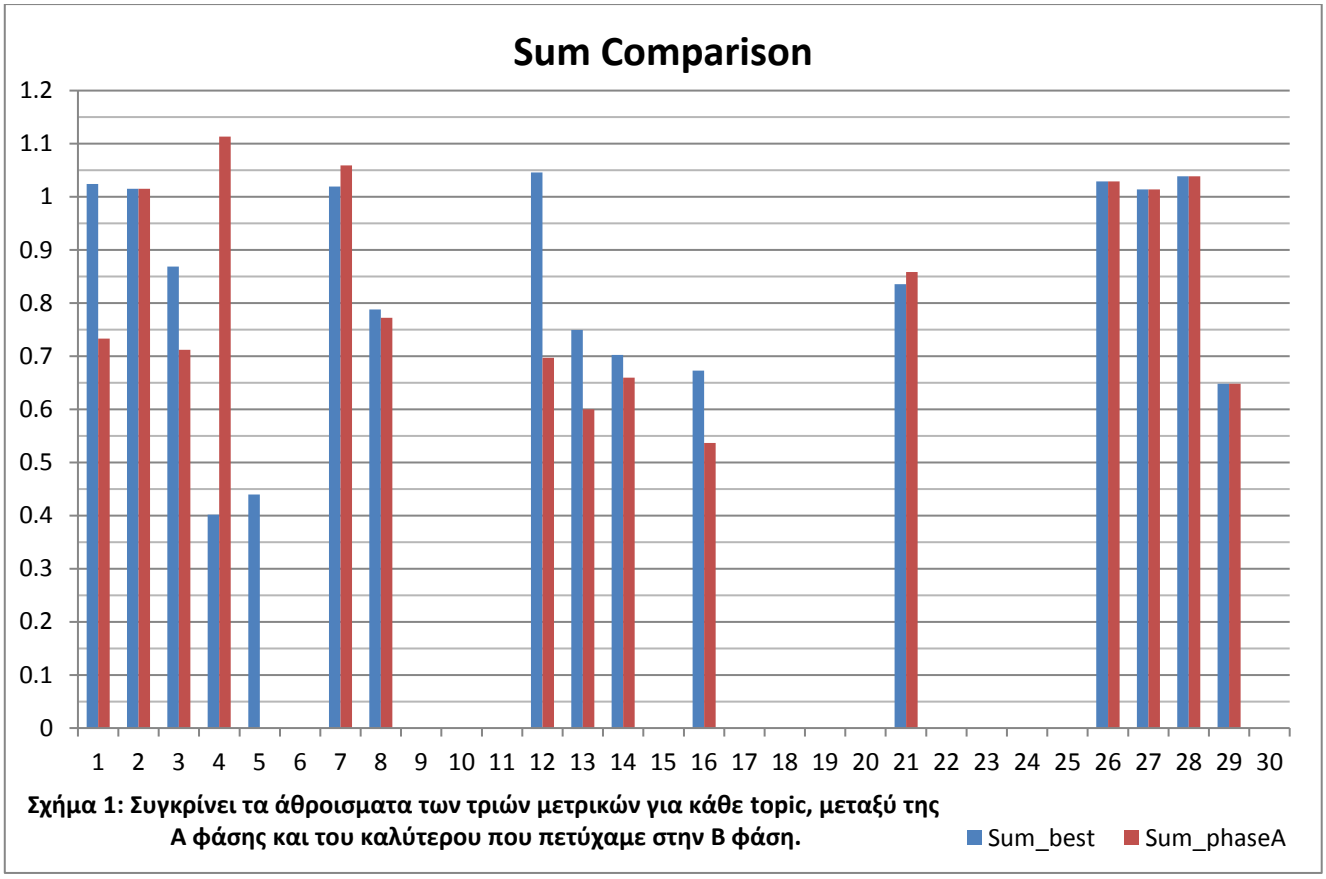
Summary:	290 seconds ≈ 5 λεπτά
Description:	409 seconds ≈ 7 λεπτά

- *Μετρήσεις για την φάση Β*

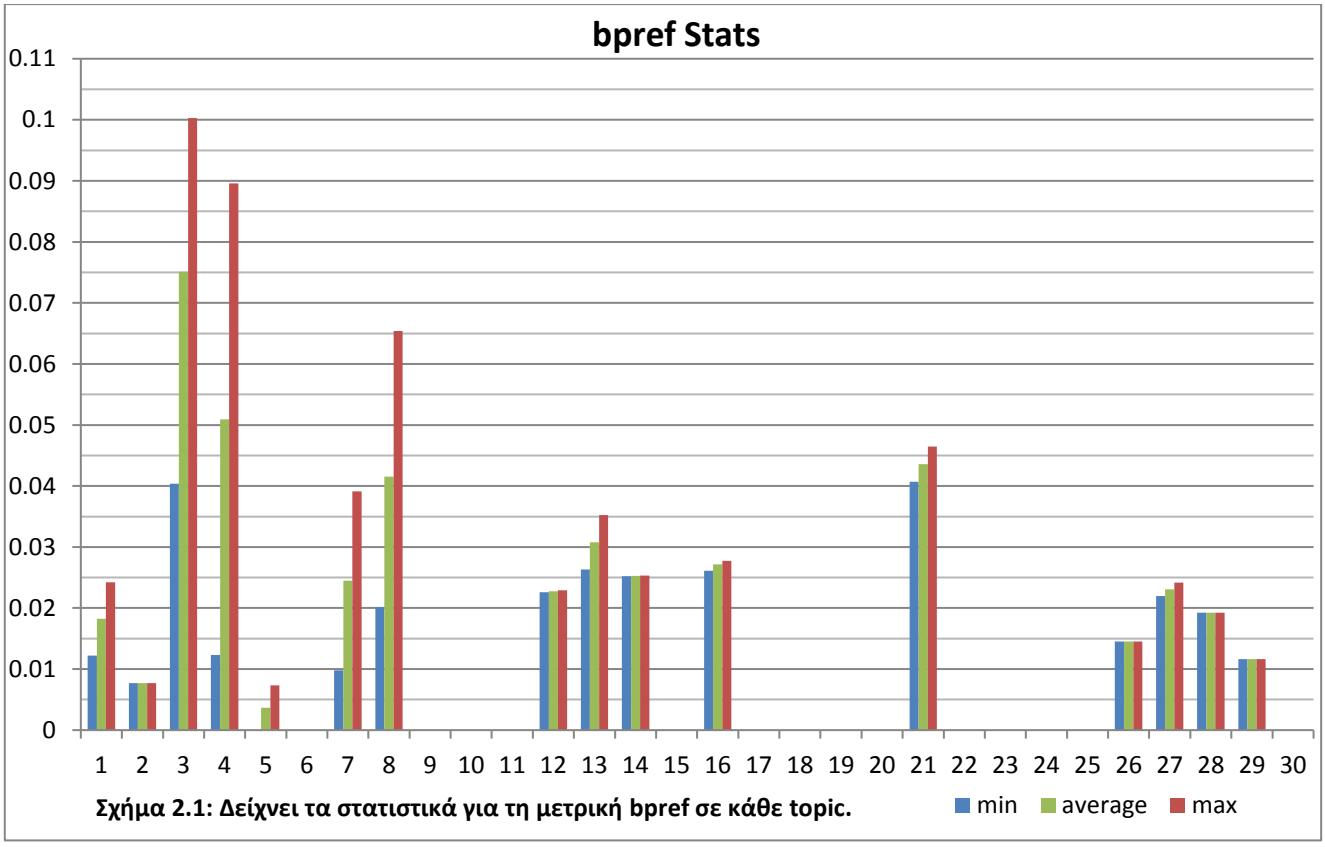
Στην φάση Α, όταν γινόταν η αναζήτηση ενός topic, βάζαμε στο τέλος του query τον τύπο του topic με σκοπό να βρεθούν αποτελέσματα που σχετίζονται με αυτό το topic. Όμως είδαμε ότι χωρίς αυτό, τα αποτελέσματα είναι καλύτερα οπότε το αφαιρέσαμε.

Επίσης προκειμένου να βελτιώσουμε τα αποτελέσματα της αναζήτησης δώσαμε, κατα το indexing, μεγαλύτερο βάρος στις λέξεις που βρίσκονται στο title, στα categories ή στο pmcid. Κάναμε κάποια πειράματα για να δούμε ποιό θα είναι αυτο το βάρος και το καλύτερο αποτέλεσμα φαίνεται στο σχήμα 1, ως άθροισμα και των τριών μετρικών.

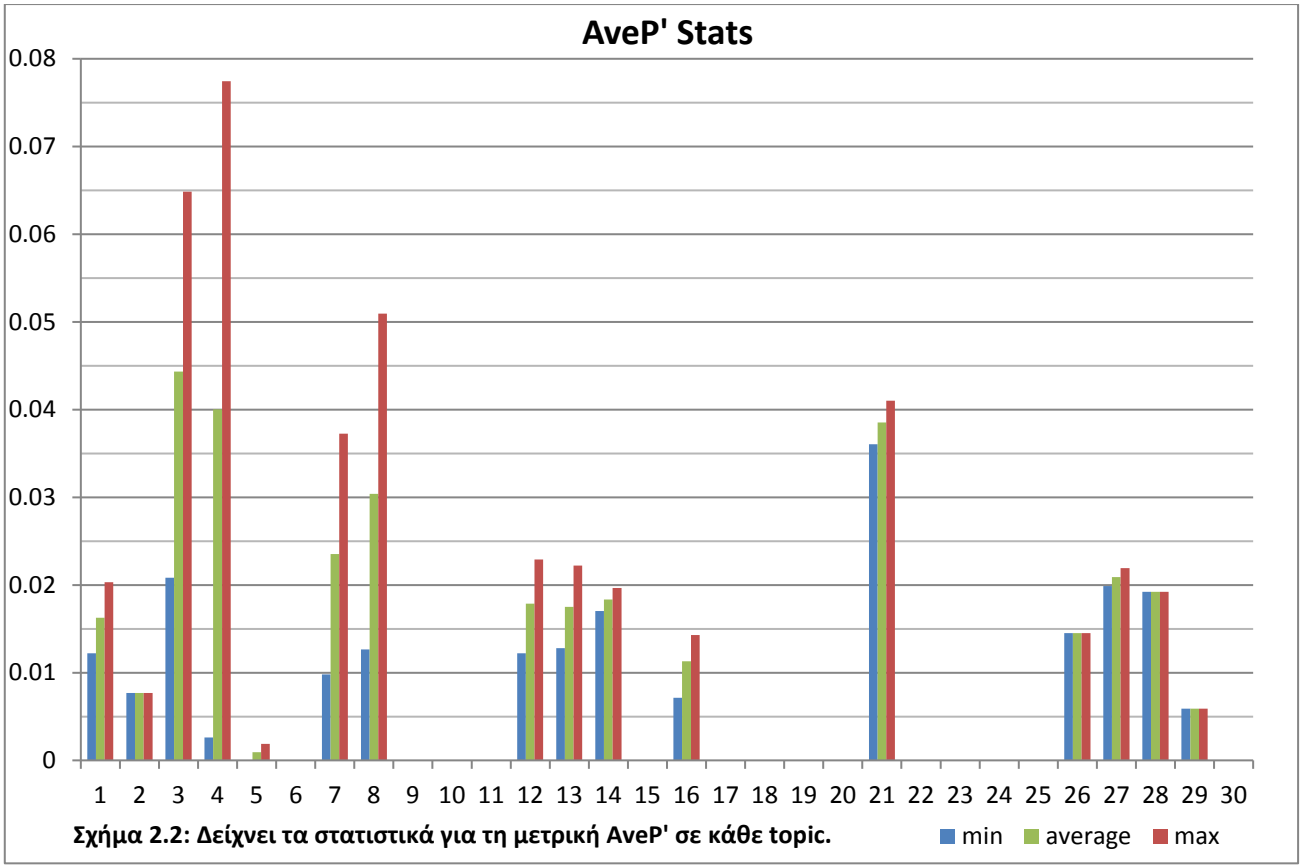
Στα σχήματα 2.1 – 2.3 φαίνεται για κάθε μετρική η μικρότερη και η μεγαλύτερη τιμή που καταφέραμε να βρούμε σε κάθε topic, όπως επίσης και το μέσο όρο απο όλα τα πειράματα που κάναμε προκειμένου να βελτιώσουμε το σύστημα.



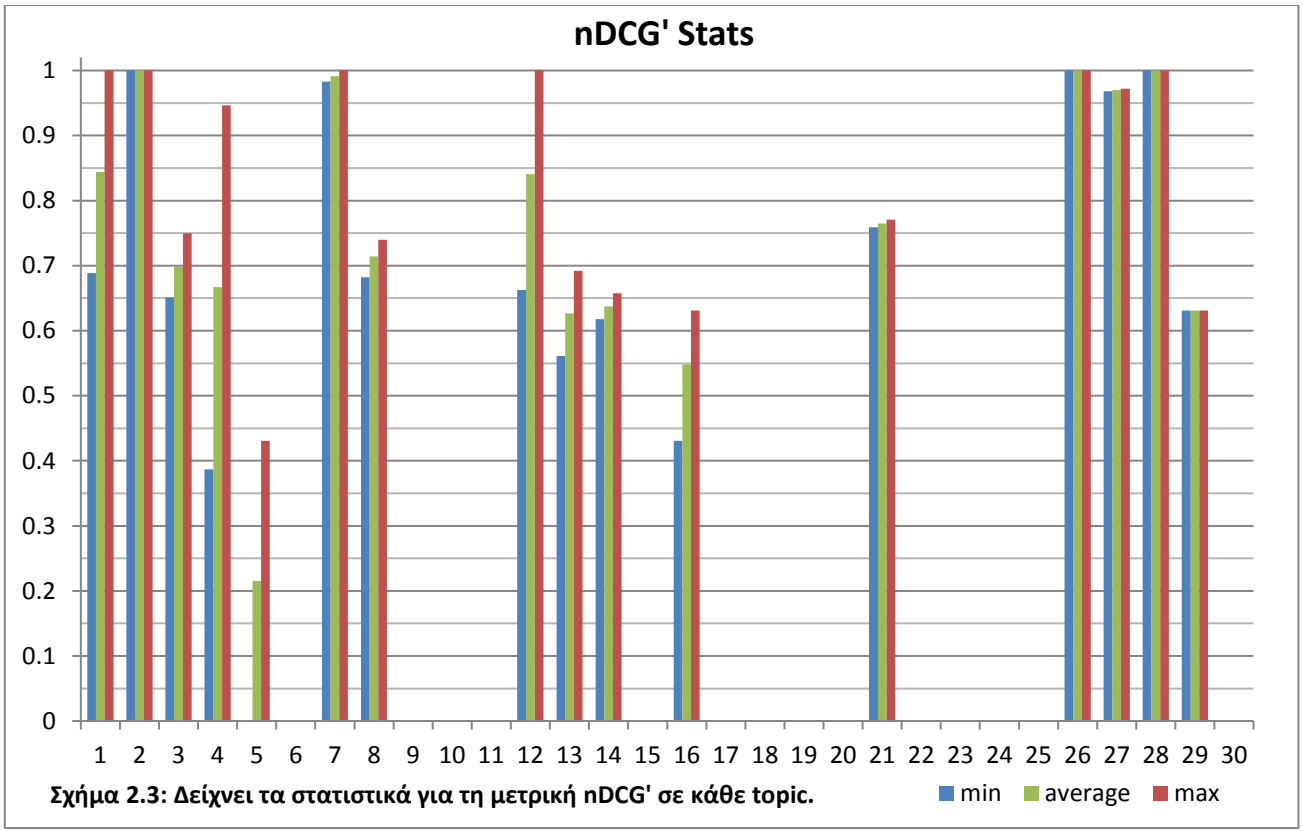
Σχήμα 1: Συγκρίνει τα άθροισματα των τριών μετρικών για κάθε topic, μεταξύ της Α φάσης και του καλύτερου που πετύχαμε στην Β φάση.



Σχήμα 2.1: Δείχνει τα στατιστικά για τη μετρική bpref σε κάθε topic.



Σχήμα 2.2: Δείχνει τα στατιστικά για τη μετρική AveP' σε κάθε topic.



Σχήμα 2.3: Δείχνει τα στατιστικά για τη μετρική nDCG' σε κάθε topic.

4 Επίλογος

Γενικά σε αυτήν τη φάση βελτιώσαμε αρκετά το σύστημα, κυρίως σε θέματα ταχύτητας αλλά και απόδοσης των αποτελεσμάτων. Μάλλον, λόγο του ότι καταφέραμε να κάνουμε indexing μόνο την μικρή συλλογή(5 gb), δεν είχαμε τα καλύτερα αποτελέσμα των topics. Τέλος το σύστημα αξιολόγησης που υλοποιήσαμε βοήθησε πολύ στην κατανόηση και των μετρικών, αλλά και της δυσκολίας του να βρεθούν σωστές μετρικές [7] αξιολόγησης.

5 Αναφορές

[1] http://stackoverflow.com/, 2017.
[2] https://www.tutorialspoint.com/, 2017.
[3] https://www.google.gr/, 2017.
[4] https://en.wikipedia.org/wiki/Discounted_cumulative_gain, 2017.
[5] Ricardo Baeza-Yates, Berthier Ribeiro-Neto, Modern Information Retrieval: The Concepts and Technology behind Search (2nd Edition).
[6] Buckley, C. and Voorhees, E. M.: Retrieval Evaluation with Incomplete Information, ACM SIGIR 2004 Proceedings, pp. 25-32, 2004.
[7] Sakai, T.: Alternatives to Bpref, SIGIR 2007 Proceedings, 2007.
[8] Yilmaz, E. and Aslam, J. A.: Estimating Average Precision with Incomplete and Imperfect Judgments, CIKM 2006 Proceedings, 2006.