Project Phase B

Βαρσάμης Χαράλαμπος csd3744
Απόστολος Μαυρογιαννάκης csd3799

July project Submission

September project Submission

Introduction

Κομματια υλοποιησης:

Αρχικα ειχαμε να διορθωσουμε το κομματι του Indexing της πρωτης φασης.Ειχαμε θεματα στο merge,δεν ειχαμε υλοποιησει τα A9,10 ,δεν ειχαμε υπολογισει τα βαρη στο Document's file και ειχαμε θεματα κατα την τοποθετηση και ταξινομηση των δεδομενων κατα την διαδικασια του Json Read.

Ο Αποστολος Μαυρογιαννακης ανελαβε να διορθωσει το merge(A6) και να αποθηκευσει τα βαρη στο document's file.

Ο Βαρσαμης Χαραλαμπος ανελαβε να διορθωσει και να υλοποιησει τα A8,9,10.

Εκτος αυτων ειχαμε και καποια μικρο προβληματα κατα την επεξεργασια των Json reads και το προγραμμα μας καθυστερουσε πολυ οποτε και οι δυο μας ασχοληθηκαμε για να βρουμε και να λυσουμε αυτα τα προβληματα.

Οσον αφορα το κομματι της Β φασης:

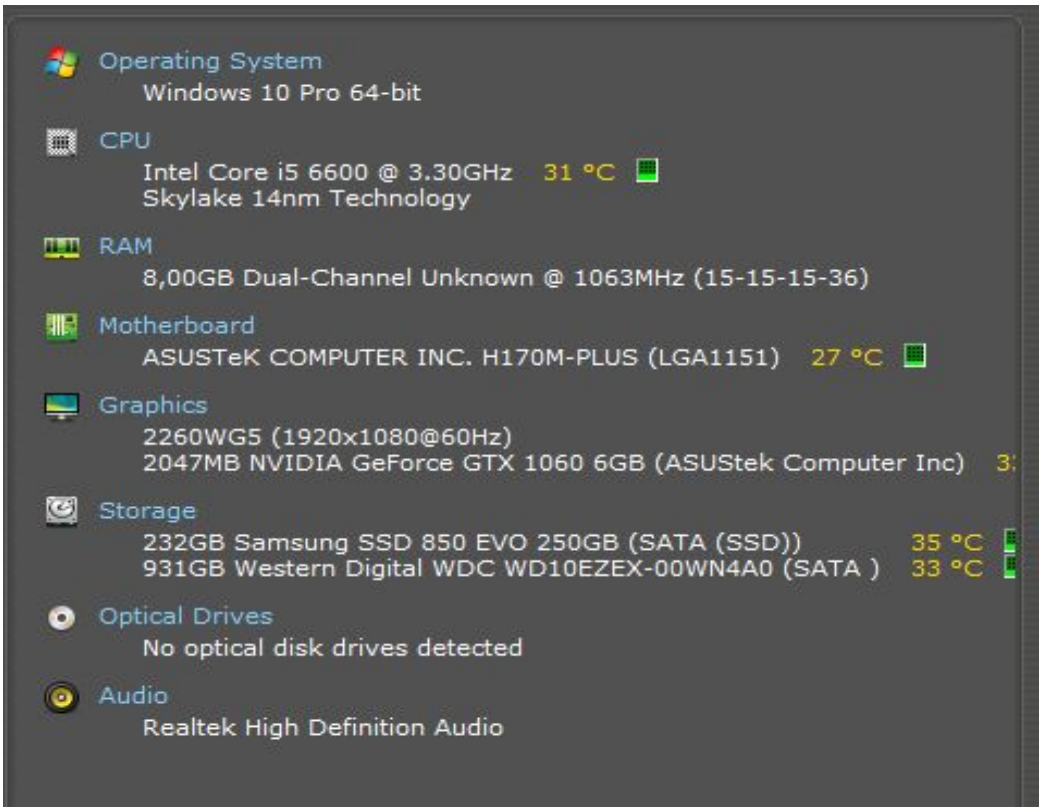
Ο Αποστολος Μαυριογιαννακης ανελαβε να υλοποιησει τα B5,B6.

Ο Βαρσαμης Χαραλαμπος ανελαβε να υλοποιησει τα B2,B3,B4.

Δεν αντιμετωπισαμε καποιο ιδιαιτερο προβλημα περα απο τις βελτιστοποιησεις που δεν προλαβαμε να κανουμε σε μερικα κομματια οπως το να διαβαζουμε 4KB απο το Documents file για να αξιοποιησουμε την χωρικη και χρονικη τοπικοτητα.Αυτο δεν το προλαβαμε γιατι χασαμε μεγαλο χρονικο διαστημα προσπαθωντας να διορθωσουμε την Α φαση και στο τελος δεν καταφεραμε να ασχοληθουμε πολυ με την βελτιστοποιηση του μερος των Queries.Στο B2 για καποιο λογο εχουμε καποιο λαθος με τις τιμες του AveragePrecision το οποιο δεν βρηκα ακομα τι ειναι(δεν προλαβα γιατι πετουσα 15 και μεχρι τοτε ασχολιομουν με τα B3,B4).Στο B5 καταφεραμε να το τρεξουμε αλλα παρατηρησαμε οτι το threshold 10^{-4} ηταν πολυ μικρο και τελειωσε γρηγορα.Να επισημανω ομως οτι ο στοχος του ερωτηματος ηταν να βρουμε εναν καλο τροπο για να αποθηκευσουμε τα δεδομενα στην μνημη χωρις να καταρρευσει το προγραμμα λογω μνημης το οποιο το καταφερε ο Αποστολος Μαυρογιαννακης.

Απο ολα αυτα καταφεραμε να τρεξουμε στην Pangaia τα B1,B2,B5.Τα υπολοιπα λογω χρονου αναγκαστηκαμε να τα τρεξουμε στην μικρη συλλογη.

Αυτα που τρεξαμε στην μικρη συλλογη αξιολογηθηκαν απο συστημα το οποιο αποτελουνταν απο τα εξης εξαρτηματα:



Important parts of the project

Phase A:

Starts from CreateIndex.index()

Indexer:Method index(String path)

Εδω γινεται η διαδικασια του indexing η οποια περιεχει τα Json reads ολων των φακελων την αποθηκευση τους σε μια δομη HashMap ωστε κατα την διαρκεια του indexing να εχουμε read/write σε O(1) (ιδανικο) και πριν το dump να τα μεταφερουμε σε ενα TreeMap ωστε να γινονται sorted κατα λεξικογραφικη σειρα.

Για το καθε entry ακολουθει μια συγκεκριμενη προετοιμασια η οποια γινεται στην μεθοδο Actions(int articleId,S2TextualEntry entry) ή οποια ετοιμαζει το entry για το documents file και μετα απο αυτο το τελικο σωμα αφαιρει τους αριθμους και καποιους ειδικους χαρακτηρες που δεν προσφερουν καποιο ιδιαιτερο νοημα στο κειμενο(.replaceAll).Επειτα ακολουθει η πιο βαθια επεξαργασια η οποια ειναι η αφαιρεση των stopwords και stemming τα οποια μας τα δωσατε εσεις οποτε δεν θα αναφερθω στην διαδικασια αφαιρεσης/επεξαργασιας τους.Μολις τελειωσουν αυτες οι διαδικασιες αποθηκευουμε στην δομη DocumentStructure η οποια ειναι υπευθυνη να κραταει για το καθε αρθρο στο αναλογο πεδιο τα αναλογα δεδομενα.

Σημειωση:Σε αυτην την συναρτηση καλουμε το ApplyBoth(both stemming and stopword removal).Εδω αντιμετωπισαμε ενα προβλημα που μας επιβραδυνε το προγραμμα κατα 4 λεπτα(Το JsonRead Που ανεφερα και η ταξινομηση/τοποθετηση των δεδομενων).Το προβλημα ηταν οτι ψαχναμε το Occurrence σε ολη την λιστα για την καθε λεξη και διαβαζαμε ολα τα πιθανα document ID’s για να το βρουμε το οποιο σημαινει οτι ειχαμε n^2 χρονικη πολυπλοκοτητα και μας αργουσε κατα πολυ.Στο τελος ειδαμε οτι εφοσον μπαινουν sorted μπορουμε να ξεκιναμε απο το τελος οποτε θα ειναι O(1) lookup search.Οποτε απο τα 4 λεπτα πεσαμε στα 30 δευτερολεπτα και ειχε ως αποτελεσμα ο χρονος μας να βελτιωθει σημαντικα.

Συνεχιζουμε και το κανουμε αυτο μεχρι να φτασουμε στα Partial_Doc_Size εγγραφα οπου ειναι το συνολικο πληθος εγγραφων του καθε partial.Οταν φτανουμε σε αυτο το νουμερο αυτο που κανουμε ειναι να κανουμε dump στον δισκο τα δεδομενα μας

και να δημιουργουμε ενα partial με id το νουμερο του +100 γιατι μας δημιουργηθηκε ενα προβλημα (το 0,1 γινονται “1””0” και εχουν ιδιο id με το “10”οποτε κρασσαρει).Φυσικα μετα καθαριζουμε τις δομες μας και το ξανατρεχουμε απο την αρχη για το επομενο partial.

*Σημειωση: Στο νεο dump(γτ το διορθωσαμε), αποθηκευουμε ολα τα bytes σε ενα ByteOutputStream το οποιο δεν ειναι fixed size,επομενως μας δινει την δυνατοτητα να κανουμε συνεχως append(λυνει το προβλημα των μεταβλητων πεδιων-titles,authors,authors id…-) και στο τελος γραφουμε στον δισκο τα bytes που περιεχει το ByteArrayOutputStream.Επισης κατα την διαρκεια του dump αποθηκευουμε σε ενα HashMap τα WeightOffsets ωστε να μπορεσουμε μετα οταν ερθει η ωρα να υπολογισουμε τα βαρη να παμε κατευθειαν σε αυτο το offset και να τα αναγραψουμε απο 0 σε καποιον αριθμο.Επισης σημαντικο ειναι να αναφερουμε πως κανουμε dump τα KeyEncoding_Offsets το οποιο ειναι (τα offsets για τα βαρη του καθε εγγραφου) στην μνημη ωστε αν τα χρειαστουμε να τα κανουμε load,πραγμα που οντως γινεται γιατι χρειαζομαστε τα offsets για το pagerank οποτε φορτωνοντας στην μνημη τα offsets απο τα βαρη μπορουμε να βρουμε και τα offsets για το pagerank και ετσι γλυτωνουμε πολυ χρονο στο ερωτημα B5. *

Αυτο συνεχιζει μεχρι να τελειωσουν τα εγγραφα που πρεπει να διαβαστουν και αφου διαβαστουν και δημιουργηθουν και τα καταλληλα partials τοτε ξεκιναει η διαδικασια του merge.

Το merge χρειαστηκε να το διορθωσουμε απο την προηγουμενη φαση καθως αργουσε πολυ αλλα λειτουργουσε.Καναμε ανα δυο τα merge(ολα μαζι στο τελος αλλα σε δυαδες).Χασαμε πολυ χρονο στην διορθωση του merge αλλα καταφεραμε να φτιαξουμε μια ικανοποιητικη Merge μεθοδο.Αυτο που μας καθυστερησε ιδιαιτερα ηταν το οτι προσπαθουσαμε να εκμεταλλευτουμε την διαθεσιμη μνημη του προγραμματος για την στιγμη που θα κανουμε dump το Postings(με την μεθοδο `Runtime.getRuntime().freememory + df*16 <1 GB`) αλλα δεν λειτουργουσε σωστα και κρασσαρε το merge καθε φορα.Οποτε στο τελος αναγκαστηκαμε να κανουμε df dump το οποιο εκανε το merge μας πιο αργο.

Οταν τελειωσει το Merge ειναι η στιγμη που εχουμε ολοκληρωση το indexing και μας μενει να υπολογισουμε τα βαρη.

Αυτο που κανουμε εδω ειναι να διαβαζουμε μια λεξη και να κραταμε σε ενα HashMap το score tf-idf για την καθε λεξη στο καθε doc.Φυσικα καθε φορα που διαβαζουμε νεα λεξη αυτο που κανουμε ειναι να ψαχνουμε το εγγραφο μεσα στο map και να αθροιζουμε στο τωρινο το νεο και να το ξανα αποθηκευουμε.Μολις τελειωσει αυτη η διαδικασια για ολο το vocabulary τοτε γραφουμε μεσα στους φακελους χρησιμοποιωντας τα weight offsets τα βαρη(νορμα) του καθε εγγραφου.

Παρατηρησαμε οτι υπαρχουν εγγραφα τα οποια ειναι αδεια οποτε το βαρος τους παραμενει 0.

*Σημειωση: Εδω εχουμε ενα θεμα,δεν ξερουμε γιατι αλλα οταν καλουμε την CalculateWeights μετα το merge τρωμε NullPointerException το οποιο μπορει να γινεται λογω καποιου buffer που ξεχναμε ανοιχτο,ομως δεν βρηκαμε τιποτα τετοιο και στο τελος δεν καταφεραμε να λυσουμε αυτο το προβλημα.Αυτο που μπορεσαμε να κανουμε ειναι να ξανατρεξουμε το προγραμμα μονο για το CalculateWeights, δηλαδη χωρις το Indexing, φορτωνοντας στην μνημη τα weight offsets και λειτουργησε με επιτυχια. *

Μετα απο ολα εχουμε και την δημιουργια του meta_index_info το οποιο ειναι αυτο που κραταει χρονους,το μεσο μεγεθος εγγραφου σε πληθος λεξεων,το path,το μεγεθος των partials,τα stemming/stopwords και την μεγιστη μνημη.

Search for PhaseA Search_FR for PhaseB FR->FileRead.

Για τα A8,9,10 εχουμε εναν σχετικα παρομοιο τροπο διαχειρισης των δεδομενων,το μονο που αλλαζει ειναι τα δεδομενα που χρειαζομαστε για να βγαλουμε το score στην καθε περιπτωση.Αυτο που κανουμε ειναι να παιρνουμε το query,το επεξεργαζομαστε με τον ιδιο τροπο που επεξεργαζομαστε το σωμα κατα το Indexing και επειτα το σπαμε σε ορους της μορφης QueryTerm και αναλογα με το ποσες φορες εμφανιζεται ενας ορος αλλαζει και το βαρος του(πχ αν η λεξη “dog” υπαρχει δυο φορες η βαρυτητα της αυξανεται).

Οσον αφορα τα μοντελα σε ολα αυτο που κανουμε ειναι να διαβαζουμε την καθε λεξη απο την λιστα με τα QueryTerms απο το vocabulary το οποιο το κανουμε load στην μνημη σε ενα hashmap για γρηγορα Lookup.Επειτα παιρνουμε το df και τον offset pointer και πηγαινουμε σε αυτο το offset στο postings και κανουμε Df*16 read.Επειτα για καθε entry στο Postings List που διαβασαμε υπολογιζουμε το score αυτου του εγγραφου(Το αποθηκευουμε σε μια δομη HashMap<DocumentID,HashMap<Term(String),Score>>) για αυτον τον ορο και φορτωνουμε στην μνημη την νορμα του εγγραφου.Το κανουμε αυτο για καθε QueryTerm μεχρι να εξαντληθουν ολοι.Οταν εξαντληθουν αυτο που κανουμε ειναι να εφαρμοζουμε για τον καθε ορο στο καθε εγγραφο το cosine similarity ή το αναλογο για το Okapi BM25.Τελος εχουμε μια λιστα με documents τα οποια πρεπει να τα ταξινομησουμε το οποιο το κανουμε με την χρηση των Collections.sort. Αν χρησιμοποιησουμε την συναρτηση που επιστρεφει τα topk αποτελεσματα τοτε αυτο που κανουμε ειναι πριν απ’ολα να ταξινομησουμε τους ορους στα QueryTerms με βαση το βαρος τους(βρισκουμε τα διπλα,δηλαδη “dog cat dog”->”dog”,2>”cat”,1 και επειτα βρισκουμε το df του ορου και αυτο που κανουμε ειναι να τα κανουμε sort με βαση το tf-idf του QueryTerm) και επειτα να ψαξουμε αυτους τους ορους και να

κανουμε εξοδο την στιγμη που εχει ολοκληρωθει το πληθος topk και οι υπολογισμοι του συγκεκριμενου df ωστε να μην υπαρχει μεγαλο σφαλμα μεταξυ των πραγματικων τιμων με αυτων που θελουν τα πρωτα topk αποτελεσματα.Αν μαζεψουμε τα topk αλλα υπαρχουν και αλλοι οροι οι οποιοι μπορουν να βαθμολογηθουν σωστο ειναι να αναφερουμε πως δεν προσμετρουνται γιατι 1)Μαζεψαμε το πληθος που θελαμε και 2) Ειναι πιθανο να μην προσφερουν κατι το ιδιαιτερο στο συνολο καθως ειναι χαμηλοτερα στην λιστα με τα sorted QueryTerms.Επισης μπορουμε να πουμε οτι ο στοχος αυτης της συναρτησης ειναι να δωσει αποτελεσματα πολυ πιο γρηγορα με οσο μικροτερο σφαλμα γινεται πραγμα που επιτυγχανεται αρα ειμαστε σχετικα καλυμμενοι.

PhaseB:

Το B1 ειναι η A φαση.

Το B2 ειναι οι υπολογισμοι των μετρικων στις οποιες οπως ανεφερα στην αρχη,εγινε καποιο λαθος το οποιο δεν προλαβα να βρω και το AP βγαινει σε ολα >1 και το average,min,max 1.0.Το nDCG φαινεται να δουλευει σωστα.

Αυτο που κανουμε ειναι να παιρνουμε ενα map το οποιο κραταει τα hashkeys και το relevance bit se μορφη boolean και αναλογα με την τιμη της μεταβλητης να κανουμε και τις απαραιτητες αλλαγες στις μεταβλητες και τους καταλληλους υπολογισμους/αθροισματα.Στο τελος αφου τελειωσουν ολα αυτα κραταμε σε μια λιστα τα APs/nDCGs και υπολογιζουμε το Mean και το Average score.

Στο B3 προσπαθησαμε να δοκιμασουμε κατι διαφορετικο αλλα για καποιον λογο ειναι πολυ αργο.Δεν προλαβαμε να δουμε ποιο ειναι το πραγματικο προβλημα και τι ειναι αυτο που φταιει αλλα η ιδεα μας μας φανηκε ενδιαφερουσα.Αυτο που καναμε ειναι να υπολογιζουμε το score του Αρχικου Query και να βρουμε για τον καθε ορο αλλα 2 Συνωνυμα(αν εχει).Καθως βρισκουμε τα συνωνυμα αυτο που κανουμε ειναι να ξαναφτιαχνουμε ενα Query για καθε συνωνυμο και επειτα να υπολογιζουμε το score για το αρχικο query για το Συνωνυμο Query και για το αλλο Συνωνυμο Query. Ο τροπος που εγιναν αυτα σιγουρα δεν ειναι ο βελτιστος και το γνωριζουμε,απλα δεν προλαβαμε λογω πολλων απροβλεπτων προβλημματων που αντιμετωπισαμε στα προηγουμενα.Αυτο που ειναι ενδιαφερον ειναι οτι επιδιωξαμε να χρησιμοποιησουμε και αντωνυμα στην βαθμολογηση,για καθε ορο του αρχικου query προσπαθησαμε να βρουμε και ενα αντωνυμο.Οταν βρηκαμε τα αντωνυμα αυτο που καναμε ειναι να υπολογισουμε το score για τα αντωνυμα και να τα κρατησουμε σε μια λιστα αρχικα και επειτα σε ενα HashMap(για γρηγορο Lookup) για να τα χρησιμοποιησουμε ωστε να προσθεσουμε αρνητικη βαρυτητα στα αρχικα queries.Δηλαδη ειχαμε μια λιστα με εγγραφα για το αρχικο,για το συνωνυμο1,για το συνωνυμο2 και για το αντωνυμο.Τα υπολογιζαμε ολα αυτα και μετα αφαιρουσαμε απο τα κοινα εγγραφα αρχικο ^ αντωνυμο,Συνωνυμο1 ^ αντωνυμο , Συνωνυμο2 ^ αντωνυμο τα score του αντωνυμου απο τα αλλα.Η ιδεα ηταν οτι αν εχουμε καποια εγγραφα και για καποιον λογο το αντωνυμο λαβει υψηλο score σε αυτα τα εγγραφα τοτε αυτο θα σημαινει οτι αυτο το εγγραφο ειναι πιθανοτατα αντιθετο με αυτο που ψαχνουμε εμεις(εφοσον ειναι αντωνυμο),επομενως πρεπει το αρχικο/συνωνυμο1,2 να βαθμολογηθει πιο χαμηλα.

Το Β4 λειτουργησε μονο με συνωνυμα και χωρις καποιο προβλημα.Ετρεξε την μικρη συλλογη μεσα σε 20 λεπτα.Η μονη παρατηρηση η οποια ειναι αξιας προσοχης ειναι οτι μεσα στα queries υπαρχαν οροι οι οποιοι μπορει να ηταν ονοματα ή ονομασιες για ιδιαιτερα πραγματα και παλι ομως εβρισκε με κατι να τα συσχετισει.

το Β5) Αρχικα τρεχω το documents file, -διοτι ειναι πιο γρηγορο απο το να διαβασω ολο το collection απο την αρχη (γλιτωνω το διαβασμα αρκετης αχρηστης πληροφοριας)- διαβαζω απο το documents file το hashkey καθε εγγραφου και δημιουργω το αντιστοιχο node με αυτο το id, μετα το κανω insert μεσα graph. Επειτα, τρεχω ξανα ολη την συλλογη , δημιουργωντας για καθε publication ενα GraphEntry απο το οποιο παιρνουμε τα citations , και δημιουργω τα καταλληλα edges απο την πληροφορια που πηρα απτο GraphEntry. Αρα μεχρι τωρα στην μνημη εχουμε δυο HashMap , το ενα που αποθηκευει τα Nodes ( ), και το αλλο που αποθηκευει τα Edges, τα οποια βρισκονται μεσα στο graph, χρησιμοποιω και ακομη ενα hashmap το οποιο αποθηκευει το PageRank καθε Node (Ονομα: PR). Για τον υπολογισμο του PageRank τωρα: κανω ενα iteration μεσα στο . hashMap , οπου σε καθε current Node που βρισκομαι, κανω iteration στα out nodes του, και σε καθε iteration αθροιζω στην προηγουμενη τιμη του PR του out node το PageRank του current node δια το πληθος του outDegree του, δηλαδη
newPR = PR.get(outNode) + (PR.get(currentNode) / graph.adjancentsList().size())
Για το threshold: Αποθηκευω προσωρινα στην μνημη το ID και το PageRank του πρωτου out Node πριν αλλαξω την τιμη του, αφου υπολογισω το καινουργιο PageRank αρχικα το αποθηκευω στο PR , και επειτα, αν η διαφορα μεταξυ της καινουργιας του τιμης και της παλιας του ειναι μικροτερη του pagerank threshold τοτε σταματαει να υπολογιζει.

το Β6) Για το Β6 πηρα τον κωδικα του OkapiBM και απλα αλλαξα τον υπολογισμο που γινεται στο score, πιο συγκεκριμενα, εκανα normalize τις τιμες των Okapi και PageRank διαιρωντας και τις δυο αυτες τιμες με την ριζα του αθροισματος των τετραγωνων των average τιμων τους, δηλαδη
new_pagerank_score = PageRank / (sqrt(pow(avgPR,2) + pow(avgOk,2)))
new_okapi_score = OkapiScore / (sqrt(pow(avgPR,2) + pow(avgOk,2)))
αρα new_final_score = new_pagerank_score * page_rank_weight + new_okapi_score*okapi_weight
Εν ολιγοις, αντικατεστησα το score που ειχαμε στο okapi με το new_final_score

Sizes:

B1

Vocabulary:420MB
Postings:53.1 GB
Documents:13.17 GB

B2

382KB
dump:String Query,String Model(vsm,okapi),String Document ID ,Double Score
dump2:Time for vsm evaluation,Time for okapi evaluation
min,max,average,mean AP,nDCG

B3

Αγνωστο

B4

426KB

B5 -

B6 19KB

Times:

B1 4 ωρες και 55 λεπτα
B2 vsm->176 minutes okapi->170 minutes
B3 not finished
B4 23.3 mins
B5 1 hour for threshold 10^-4
B6 9.5 minutes

September Project Submission

PhaseA:

Changes for Phase A:

Πηραμε την αποφαση να αλλαξουμε το index μας για να κανουμε λιγο πιο γρηγορο το QueryEvaluation και γενικα την προσβαση στο index.Η αλλαγη που καναμε ηταν στο Documents File και ειχε ως στοχο να αφαιρεσει τα μεταβλητα πεδια απο αυτην την θεση και να τα μεταφερει σε εναν αλλον φακελο αντικαθιστωντας τα απο το Documents File με εναν long pointer ο οποιος θα δειχνει σε αυτον τον φακελο.

Ο Απόστολος Μαυρογιαννάκης ανελαβε να αλλαξει την δομη του Documents File καθως και τον υπολοιπων συσχετιζομενων φακελων(Calculate Weights<-Norm)ωστε να μπορεσουμε να πετυχουμε αυτο που θελουμε.Τωρα ειναι της μορφης

String hashkey->40 Bytes
Long pointer->8 bytes
Double weight(norm)->8 bytes
Int length->4 bytes
Double Pagerank->8 Bytes

A total of 68 Bytes

Το περιεχομενο του Contents εχει εναν Long στην αρχη του καθε Entry ετσι ωστε να μπορουμε να διαβαζουμε το Publication με δυο Reads αντι για 2n(οπου n ειναι το πληθος των πεδιων και 2n το ReadBytes(ReadShort())).

Ο Βαρσάμης Χαράλαμπος για αυτην την αλλαγη χρειαστηκε να αλλαξει τον τροπο που γινεται το Read απο την μνημη για να παρει τα δεδομενα.Δηλαδη,ανελαβε να αλλαξει το Read των A8,A9,A10.

Η αλλη αλλαγη που καναμε και ανελαβε να την κανει ο Βαρσαμης Χαραλαμπος ειναι οτι δεδομενου οτι καθε Document Entry εχει 68 Bytes να αξιοποιησουμε οσο γινεται την χωρικη και χρονικη τοπικοτητα.Αυτη ειχε ως στοχο το να μετατρεψει τον Buffer που προηγουμενως διαβαζε ενα προς ενα τα Entry λογω του οτι ηταν μεταβλητου μεγεθους,τωρα να διαβαζει ενα πληθος των 60 Entries(4KB) και να ελεγχει αν το

επομενο document βρισκεται μεσα στον buffer πρωτου ξανα ξεκινησει ενα νεο seek and read στον Documents File.Ολο αυτο μας κερδιζει καποιον χρονο καθως χρειαζεται να κανουμε λιγοτερα reads απο τον φακελο που ειναι πιο αργος απο τον Buffer που ειναι φορτωμενος στην μνημη RAM.

Sizes for Phase A:

Vocabulary: 420 MB

Postings file: 53.1 GB

Documents File=Documents_File+Contents File=3.2+10.6=13.8 GB

Times for Phase A:

Total Time for Indexing process: 17407 seconds -> 4hours and 50 minutes

Reading Time 9031631713851 ns -> ~150 minutes

ApplyBoth Time(stem/stop): 6725496818728 ns -> ~112 minutes

Document Info dumping Time: 10122155230 ns -> 10.12 Seconds

Merge Time: 3792162475860 ns -> ~63 minutes

All Info Dumping Time: 1340031893769 ns -> ~22 minutes

PhaseB:

Changes for Phase B:

Στην Β φαση ο Βαρσαμης Χαραλαμπος εχει αναλαβει την υλοποιηση των Β2,Β3,Β4 και ο Αποστολος Μαυρογιαννακης εχει αναλαβει την υλοποιηση των Β5,Β6

B2:

Το Β2 ειχε ενα λαθος στις μετρικες και αυτο ηταν οτι κατα τον υπολογισμο του IDCG αλλαζε η σειρα των στοιχειων(επειδη κανουμε sort) επομενως και οι minimum τιμες επαιρναν το maximum value.

Για το Β2 αυτο που εχουμε κανει ειναι να τρεξουμε τα δυο μοντελα vsm και Okapi-bm25 και να κρατησουμε τα scores,τους χρονους και καποια αλλα δεδομενα.Σε αυτο το ερωτημα δεν θα αναφερθουμε σε αποτελεσματα απο εγγραφα καθως δεν εχουμε με κατι να τα συγκρινουμε.Αυτο που μας ενδιαφερει να κρατησουμε απο αυτο το ερωτημα αρχικα ειναι οι επιδοσεις οσο αφορα τους χρονους και τις μετρικες.Επομενως εχουμε:

min AP: 0.1
doc map: a5ca937ee45e45ac464163a63d84f4d454d676b5, 0,
521027c14a7ef622a4f03106799706d0824e192a, 0,
1b80416cc2b05954941ac7e7dcbcc358c10e5ace, 0,
30950db8a2cae3630057efe731b85f7b567848b8, 0,
80e16e823ac781051eb6c0a3ec2915773690e7ba, 0,
286f0650438d0cc5123057909662d242d7bbce07, 0,
553c99ef036cb1113fb1ba390c0e531b3f7828b8, 0,
436c3119d16ce2e3c243ffe7a4a1a5dc40b128aa, 0,
1849317564290990 9196e99b90a6af0bf3ef803b, 0,
2231f44be9a8472a46d8e8a628b4e52b9a8f44e0, 1,
15b0e598d9692d77aa33370dd3a1a47ba5f99aa6 , 0

max AP: 0.7000000000000001
doc map: 1d464ea76572e85603b4fe607f09c3953fef1aa9, 1,
316663d96332cdff9bd221ee3ee53b3cbeabbd60, 0,

47ee62088bb39c11c09130110ffcf5f3bd436764, 0,
9e5e226fe10becab0d0793cff4dca5fc4a0b5aaf, 1,
c04a2c5d59d793a42750c842dfc6e7eb1bc93ab9, 1,
1f41a574f58114afcab90eeaa4fc34df265bbd0b, 0

min nDCG: 0.2890648263178878
doc map: a5ca937ee45e45ac464163a63d84f4d454d676b5, 0,
521027c14a7ef622a4f03106799706d0824e192a, 0,
1b80416cc2b05954941ac7e7dcbcc358c10e5ace, 0,
30950db8a2cae3630057efe731b85f7b567848b8, 0,
80e16e823ac781051eb6c0a3ec2915773690e7ba, 0,
286f0650438d0cc5123057909662d242d7bbce07, 0,
553c99ef036cb1113fb1ba390c0e531b3f7828b8, 0,
436c3119d16ce2e3c243ffe7a4a1a5dc40b128aa, 0,
1849317564290990919 6e99b90a6af0bf3ef803b, 0,
2231f44be9a8472a46d8e8a628b4e52b9a8f44e0, 1,
15b0e598d9692d77aa33370dd3a1a47ba5f99aa6 , 0
max nDCG: 1.0
doc map: 907d06da6e0d84ac1798fe97ad3292277169868a, 1,
5fecbb175f956c246fd265afba3e454fba473d5e, 1,
b622f4341120beb77f242540b17056f15d1a103f, 1,
dbcae756c4a0d1122c32e44834067c88b9080cd6, 1,
fe323ace4ff38bc4edf53839654f8caee279ce91, 1,
bc6c44bd2a11ee6fcdef0451d63774dfef97b7a7, 1

Παρατηρηση:Απο κατω βλεπουμε οτι το Mean AP Και το Average AP,καθως και το mean nDCg και το Average nDCG εχουν τις ιδιες τιμες καθως ολα τα entries του judgements file εχουν 1 Query και ο λογος που κανει το mean να διαφερει απο το Average τυπο ειναι οτι εχει εναν εξτρα παραγοντα που προσμετραει το πληθος των queries για ενα entry.Οταν ομως |Q|==1 Mean_type_Score==Average_type_Score.

mean AP: 0.6632522778256311
Average AP: 0.6632522778256311
mean nDCG: 0.7841503036869304
Average nDCG: 0.7841503036869304

Εδω γενικα παρατηρουμε οτι η Ακριβεια δεν εχει ασχημη μεση τιμη,μαλιστα θα μπορουσε να αποκαλεστει και σχετικα ικανοποιητικη.Ωστοσο βλεπουμε πως εχουμε και ενα minimum AP το οποιο ειναι πολυ κακο σαν στατιστικο αλλα γενικα αν επικεντρωθουμε στην Μεση Τιμη των αποτελεσματων και οχι σε καθε ενα απο αυτα μεμονωμενα(γιατι το 1/635 ειναι ενα μικρο μερος του συνολου) θα δουμε πως τα

αποτελεσματα δεν ειναι ασχημα.Απο την αλλη,θελουμε παντα να βρισκουμε και τα χειροτερα ωστε να μπορουμε να τα μελεταμε για να βλεπουμε τι πηγε στραβα και να βρισκουμε εναν τροπο που να καλυπτει καλυτερα τον χρηστη μας(για τις μελλοντικες χρησεις).

Συνεχιζοντας αν παρατηρησουμε το nDCG του οποιου ο βαθμος προκυπτει με βαση την θεση του καθε εγγραφου θα δουμε πως εχουμε μια πολυ κακη τιμη 0.29 για το χειροτερο οπου το συναφες μας εγγραφο εμφανιζεται στην προτελευταια θεση.Απο την αλλη εχουμε και μια πολυ καλη περιπτωση οπου ολα χαρακτηριστηκαν relevant απο τον χρηστη το οποιο ειναι πολυ ευχαριστο και ιδανικο γεγονος.Τελος αν παρατηρησουμε την Μεση τιμη θα δουμε οτι γενικα εχουμε πολυ καλα αποτελεσματα τιμης υψους 0.78.Αυτο σημαινει οτι τα περισσοτερα entries εχουν τα εγγραφα τους στο πιο πανω απο το πρωτο μισο της λιστας τους(Προχειρη εκτιμηση).

*Παρατηρηση:Κατι που θα μπορουσε να ειναι μη επιθυμητο ειναι πολλα εγγραφα να ειχαν στις πρωτες θεσεις τα relevant documents και αλλα στις τελευταιες θεσεις και ετσι θα καταληγαμε σε μια στατιστικη η οποια θα ειχε μετριες τιμες ως αποτελεσματα αλλα στην πραγματικοτητα θα αποτελουνταν απο ακραιες περιπτωσεις(πρωτες και τελευταιες θεσεις).Φυσικα το θετικο με αυτο το γεγονος ειναι οτι θα ειχαμε καποια παρα πολυ καλα και αυτα που θα κατηγοριοποιοντουσαν ως “κακα” θα μπορουσαμε να τα απομονωσουμε για να τα μελετησουμε και να τα βελτιωσουμε. *

Περα απο μετρικες για το B2 εχουμε κρατησει και την χειροτερη επιδοση(βαση χρονου).Επομενως εχουμε:

Για το vsm:

Time for whole set of queries for vsm evaluation: 4316564664528 -> ~72 minutes

Worst Time for vsm: 84264982110 ->~84.2 seconds
With Query: style of presentation of the results obtained using Coq
Worst time for vsm Reading the info: 84264672107 ns-> ~84 seconds
Worst time for vsm calculating the scores: 296942 ns-> 0.000296942 seconds
Worst time for vsm Sorting the list: 6592 ns-> $6.59200 * 10^{-6}$ seconds

Για το Okapi-bm25:

Time for whole set of queries for bm25 evaluation: 4360133665055 -> ~72 minutes

Worst Time for bm: 82006991950 -> ~82 seconds
With Query: style of presentation of the results obtained using Coq
Worst time for Okapi-BM25 Reading the info: 82006715510 -> 82 seconds

Worst time for Okapi-BM25 calculating the scores: 264505 -> 0.000264505 seconds
Worst time for Okapi-BM25 Sorting the list: 4816-> 4.81600 * 10^{-6} seconds

Βλεπουμε οτι το χειροτερο μας query πηρε 1 λεπτο και 22~24 δευτερολεπτα.Αυτο ειναι πολυ αργο δεδομενου οτι ο χρηστης θελει αμεσα τα αποτελεσματα.Απο την αλλη δεν γνωριζω τι αλλο μπορουσαμε να κανουμε ωστε να παρουμε πιο γρηγορα τα αποτελεσματα μας.Κοιτωντας το αρχειο βλεπω πως υπαρχουν μονο 6 εγγραφα επομενως η ιδεα οτι μπορει να ηταν 10-20 ή και 30 οπως σε αλλες περιπτωσεις ειναι εκτος.Ωστοσο, βλεπουμε οτι αυτο που καθυστερησε τοσο πολυ ηταν η αναγνωση των δεδομενων απο την μνημη,πραγμα που σημαινει οτι ειτε το αρχειο ηταν απο τα μεγαλυτερα ειτε οτι το query ηταν πολυ μεγαλο.Δεδομενου οτι δεν διαβαζουμε δεδομενα απο τον φακελο Contents(μεταβλητα πεδια) μπορουμε να πουμε οτι η πρωτη περιπτωση ειναι εκτος,επομενως ο λογος ειναι οι πολλοι οροι που επρεπε να μελετηθουν και να χρησιμοποιηθουν για τον υπολογισμο του score του καθε εγγραφου.Κοιτωντας προχειρα τον Judgements file μπορουμε να πουμε με βεβαιοτητα οτι τα περισσοτερα queries μετα απο stemming/stopwords εχουν πληθος ορων 2~4,ενω αυτο το query μετα απο stemming/stopwords αποτελειται απο 6 ορους.

Τελος,κλεινοντας με το B2,εχουμε και τις μεσες τιμες των χρονων που παιρνουν τα μοντελα vsm και Okapi-bm25 για να βαθμολογησουν τα queries.Αρα:

Average time for vsm: 6797739629 -> ~6.8 seconds
Average time for bm: 6866352228 -> ~6.9 seconds

Πραγμα το οποιο θα μπορουσαμε να πουμε πως ειναι εν μερει λογικο αν υπαρχουν και αλλα queries που παιρνουν χρονο σαν το χειροτερο(πανω απο λεπτο).Μια μικρη παρατηρηση εδω ειναι οτι το vsm ειχε κατα λιγο χειροτερο χρονο απο το bm αλλα το bm εχει χειροτερο μεσο χρονο,πραγμα που σημαινει πως κατα κυριο λογο ειναι λιγο πιο αργο.

***Παρατηρηση:Παρατηρωντας τα score που επιστρεφουν τα μοντελα παρατηρησα οτι το Okapi εχει το αρνητικο οτι επιστρεφει κατα κυριο λογο το ιδιο score στα κοντινα εγγραφα ενω το vsm εχει μεγαλυτερη λεπτομερεια.Δηλαδη:

Model: VSM
Time for VSM evaluation: 20347203348
Results:
DocumentId: 79ea00f3039de076687375f5ef4370b33c168fdb Score: 0.4406
DocumentId: ff7a28fe2ed3d5ec32a54b7bbe62736c7a7c8c8c Score: 0.2895
DocumentId: 3306e5ad0bceb94855a286c7c6c9328998a56f52 Score: 0.2143
DocumentId: ae9a21cec1ad84d46c1f735eb9e4eb263117e65b Score: 0.1579

DocumentId: d1a3dd698e5b547614a4ccdf74b159703a4624dd Score: 0.1385

Model: OkapiBM25
Time for OkapiBM25 evaluation: 18776847970
Results:
DocumentId: 79ea00f3039de076687375f5ef4370b33c168fdb Score: 21.9051
DocumentId: ff7a28fe2ed3d5ec32a54b7bbe62736c7a7c8c8c Score: 15.9255
DocumentId: 3306e5ad0bceb94855a286c7c6c9328998a56f52 Score: 15.5591
DocumentId: d1a3dd698e5b547614a4ccdf74b159703a4624dd Score: 13.9936
DocumentId: ae9a21cec1ad84d46c1f735eb9e4eb263117e65b Score: 13.4783
Δεν ισχυει παντα αυτο,αλλα σε μερικες περιπτωσεις βλεπεις οτι το vsm εχει μεγαλυτερες διακυμανσεις σε αποτελεσματα ενω το vsm εχει πολυ κοντινα για το ιδιο query.***

Τα αρχεια που χρησιμοποιηθηκαν για αυτα τα αποτελεσματα ειναι τα:

B2results_models.idx με τελικο μεγεθος(size): ~385KB
B2metrics_models.idx με τελικο μεγεθος(size): ~2.6KB

B3:

Εχω ακολουθησει 2 προσεγγισεις για το ερωτημα B3.

1)Χρησιμοποιουμε λεξεις απο τον τιτλο και συνωνυμα λεξεων απο τον Τιτλο μαζι με το query.Η ιδεα ειναι οτι εφοσον εχουμε καποια relevant documents μπορουμε να διαβασουμε απο το περιεχομενο τους και να παρουμε μερικες λεξεις οι οποιες εχουν καποιο ουσιαστικο νοημα και καποια συνωνυμα τους.Ο στοχος μας ειναι να αυξησουμε το βαθμο συναφειας προσθετοντας στο Query μας λεξεις που υπαρχουν μεσα στα relevant documents και καποιων συνωνυμων τους ωστε να προσπαθησουμε να καλυψουμε και μεγαλυτερο ευρος εναλλακτικων επιλογων.Επειδη στο τελος,στοχος μας ειναι να βελτιωσουμε την ανταποκριση του συστηματος για αυτα τα εγγραφα,το βαρος των ορων που παιρνουμε απο τον τιτλο εχουν βαρος 2.0 αντι για 1.0.

2)Χρησιμοποιουμε και παλι λεξεις απο τον τιτλο,αλλα τωρα χρησιμοποιουμε και αντωνυμα των λεξεων που βρισκονται στον τιτλο.Η ιδεα ειναι, οτι εν μερει ειναι αναμενομενο οτι οταν εχουμε ενα συναφη ορο απο τον τιτλο ενος συναφους εγγραφου,τα αντωνυμα να θελουμε(ελπιζουμε) να εχουν χαμηλο score για τα relevant documents.Οποτε,στοχος μας ειναι να βρουμε αντωνυμα που υποθετικα θα μειωσουν ελαχιστα το score των relevant documents,αλλα απο την αλλη θα μειωσουν ακομα πιο πολυ ολων των υπολοιπων εγγραφων τα

οποια δεν ειναι relevant.Με αυτην την ιδεα,επιτρεπουμε και το αρνητικο score το οποιο ομως το βρισκω δυσκολο να επιτευχθει με το γεγονος οτι

QueryTerms+TitleTerms>>AntonymTerms

Σε Αυτην την περιπτωση εχουμε Initial Antonym Terms Weight == -1.0 αντι για 1.0.Απο την αλλη,επειδη δεν θελουμε να μειωθει και το Score των documents,αυξανουμε το βαρος των ορων του τιτλου απο 2.0 που ηταν στην πρωτη περιπτωση σε 2.5.

Οι δυο προσεγγισεις ξεχωριζονται απο το οτι η μια χρησιμοποιει antonyms και η αλλη synonyms.Οποτε εχουμε:

VSM:

Query: federated learning
Getting Titles time: 2303853
Getting Synonyms time: 472242834
Getting Antonyms time: 595543760

B2:

Time for VSM evaluation: 13732976826
Results:
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 0.426
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 0.4017
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 0.2997
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 0.2283

B3:

Synonyms Used: acquisition study eruditeness erudition with weight: 1.0
Title terms used: learning survey learning learning with weight: 2.0
Time for VSM evaluation: 42468590993
Results:
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 0.44374
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 0.3304
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 0.29584

DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 0.24383

Antonyms Used: inefficiency with weight: -1.0
Title terms used: learning survey learning learning with weight: 2.5
Time for VSM evaluation: 42468590993
Results:
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 0.44374
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 0.3304
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 0.29584
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 0.24383

Για Query: clarias batrachus heavy metals
Getting Titles time: 2216526
Getting Synonyms time: 539271272
Getting Antonyms time: 507706227

B2:

Time for VSM evaluation: 3680897923
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 0.5695
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 0.5578
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 0.5328
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 0.4944
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 0.403

B3:

Synonyms Used: Cd fleshy intense perniciousness appraisal with weight: 1.0
Title terms used: cadmium heavy acute toxicity estimation with weight: 2.0
Time for VSM evaluation: 22562342741
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 0.4456
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 0.44285
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 0.39184
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 0.32099
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 0.31981

Antonyms Used: distribution with weight: -1.0
Title terms used: cadmium heavy acute toxicity estimation with weight: 2.5
Time for VSM evaluation: 22562342741
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 0.4456
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 0.44285
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 0.39184
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 0.32099
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 0.31981

Παρατηρουμε οτι αν και η προσεγγιση φαινοταν αρκετα πρακτικη(και πολλα υποσχομενη),στην πραξη απετυχε.Τα αποτελεσματα μας ειναι 99% αρνητικα(τουλαχιστον τα τοσα που ειδα για να βρω τιποτα θετικο). Ωστοσο,μπορουμε να κανουμε καποιες υποθεσεις του γιατι εγινε αυτο.Βλεπουμε οτι :
1)Το πληθος των ορων εχει αυξηθει κατα παρα πολυ,πραγμα που σημαινει το σε συμπιεσμενες εκδοσεις του εγγραφου δεν υπαρχει μεγαλη πιθανοτητα η προσεγγιση μας να ανταπεξελθει.
2)Λαθος μου,αλλα επιδιωξα να εχω μεγαλυτερο πληθος διαφορετικων ορων παρα πολλων ιδιων με εξτρα βαρος.Αν ειχα επιτρεψει να υπαρχουν παρομοιοι οροι τοτε θα ειχαμε περισσοτερους ορους οπου θα ειχαν ολοι τους μια κοινη ιδιοτητα,οτι περιεχονται ολοι τους σε καποιο εγγραφο.

Εκτος αυτων,Βλεπουμε οτι ο χρονος για να παρουμε τα αντωνυμα και συνωνυμα ειναι μικροτερος του δευτερολεπτου,αλλα λογω του οτι εχουμε ενα τεραστιο πληθος ορων τωρα,ενα query καθυστερει υπερβολικα πολυ για να βαθμολογηθει.Αρα αυτος ο τροπος και θετικα αποτελεσματα να εφερνε δεν θα ηταν καθολου αποδεκτος χρονικα.

Μια λυση που θα μπορουσαμε να επιδιωξουμε ειναι να μειωσουμε το πληθος των ορων σε N/2,οπου N το πληθος των ορων απο τιτλους.Εκτος απο αυτο θα μπορουσαμε να μειωσουμε και το πληθος των συνωνυμων.Το αλλο πραγμα που θα μπορουσαμε να κανουμε ειναι να επιδιωκουμε να παιρνουμε μονο κοινους ορους ετσι ωστε στο τελος αντι να εχουμε 10 διαφορετικους συνωνυμους να εχουμε 5 ιδιος οι οποιοι θα χρειαστει να βαθμολογηθουν μια φορα και λογω του οτι θα ειναι μεσα στο εγγραφο θα αυξανουν το σκορ του συστηματος γενικα.

Για τα αντωνυμα το μονο που μπορω να πω ειναι οτι το πληθος τους ειναι πολυ μικρο και δεν επηρεαζει το αποτελεσμα μας καθολου.Προσπαθησα να βρω παραπανω αλλα δεν τα καταφερα.

Απο την αλλη δεν θελαμε τα αντωνυμα να μειωνουν το σκορ των εγγραφων μας πραγμα που μπορει να ειναι και παρα πολυ θετικο.Δεν εχουμε ομως αλλα εγγραφα για να δουμε πως επηρεασε αυτη η προσεγγιση το σκορ τους.Αυτο που θα μπορουσαμε να κανουμε για τα αντωνυμα ειναι να αυξησουμε το αρνητικο βαρος τους ωστε να μειωνουν ακομα πιο πολυ τα υπολοιπα documents.

Για το BM:

Για το Okapi γνωριζουμε πως το score δεν επηρεαζει,επομενως η εννοια αντωνυμο με αρνητικο score δεν υφισταται στο okapi.

Για Query: ehr phenotype framework

B2:

Time for OkapiBM25 evaluation: 6734925926
Results:
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 36.6703
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 34.8922
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 34.7084

B3:

Synonyms Used: rating depository worldly with weight: 1.0
Title terms used: evaluation metadata temporal with weight: 2.0
Time for VSM evaluation: 28681674035
Results:
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 53.2428
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 50.66478
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 41.15926

Για Query: clarias batrachus heavy metals

B2:

Time for OkapiBM25 evaluation: 3578924938
Results:
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 80.4916
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 78.021

DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 76.1253
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 71.6851
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 52.6821

B3:

Synonyms Used: Cd fleshy intense perniciousness appraisal with weight: 1.0
Title terms used: cadmium heavy acute toxicity estimation with weight: 2.0
Time for VSM evaluation: 22562342741
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 106.24301
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 97.65251
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 91.08668
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 81.89348
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 71.68515

Για Query: humanoid joint

B2:

Time for OkapiBM25 evaluation: 2540433017
Results:
DocumentId: edb1e8888894b4fe4b34a1d2fc7a4d43557a6874 Score: 38.3603
DocumentId: 7311004496d67326420f9ecb2a6a2de5aa341b10 Score: 37.2442
DocumentId: 81fa762530e53d88c0456d27803fe2496b89cfea Score: 35.8671
DocumentId: 5fa4a43bc0b1fc391e590eaa5cf61fbfff7755da Score: 24.7775

B3:

Synonyms Used: gesture motion entire force with weight: 1.0
Title terms used: human movement full push with weight: 2.0
Time for VSM evaluation: 21934553980
Results:
DocumentId: 7311004496d67326420f9ecb2a6a2de5aa341b10 Score: 75.16002
DocumentId: 81fa762530e53d88c0456d27803fe2496b89cfea Score: 74.35498
DocumentId: edb1e8888894b4fe4b34a1d2fc7a4d43557a6874 Score: 63.61502
DocumentId: 5fa4a43bc0b1fc391e590eaa5cf61fbfff7755da Score: 44.39608

Για το μοντελο Okapi παρατηρουμε οτι τα αποτελεσματα εχουν πολυ μεγαλη βελτιωση οσον αφορα το score τους.Φυσικα παλι πρεπει να κοιταξουμε με τι κοστος(χρονου) αποκτησαμε αυτο το κερδος στο score.Οποτε εχουμε:

Για Query: ehr phenotype framework

Εχουμε μια αυξηση των [12,19] περιπου μοναδων και για τα 3 εγγραφα αλλα την μεγαλυτερη την παρουσιαζει το 3ο.Ωστοσο το κοστος σε χρονο για να πετυχουμε μια τετοια βελτιωση ειναι περιπου 20 δευτερολεπτα.Πολυ ακριβος χρονος,για μια ομως μεγαλη βελτιωση.

Για Query: clarias batrachus heavy metals

Παρομοιως και εδω,βλεπουμε οτι το κοστος ειναι γυρω στα 19 δευτερολεπτα για μια μεγαλη βελτιωση.

Για Query: humanoid joint

Παρομοιως και εδω,μονο που εδω το κοστος ειναι μεν 19 δευτερολεπτα,αλλα τα score ειναι σχεδον διπλασια απο αυτο της αρχικης εκδοσης.

Τελος μπορουμε να πουμε πως υπαρχει μια βελτιωση το οποιο ειναι αναμενομενο δεδομενου οτι για μεγαλυτερο πληθος ορων θα αυξηθει το score,απλα αυτη η βελτιωση ειναι πολυ ακριβη χρονικα.

Τελος εχουμε και καποιες μετρικες για τα μοντελα μας:

Worst Time for vsm: 134059104809 with
Query: machine learning with
Synonyms: apprehension information study simple machine debut
and worst title terms: understanding data survey machine introduction machine active importance practical extreme flow support correlation machine system kernel genetic learn ml
Worst time for vsm Reading the info: 134058767154
Worst time for vsm calculating the scores: 323812
Worst time for vsm Sorting the list: 7884

With The Antonym Feature:
Worst Time for vsm: 91130623886 with
Query: machine learning
with Antonyms: declassification

and worst title terms: understanding data survey machine introduction machine active importance practical extreme flow support correlation machine system kernel genetic learn ml
Worst time for vsm Reading the info: 91130321812
Worst time for vsm calculating the scores: 290459
Worst time for vsm Sorting the list: 7679

With the Synonym Feature:
Worst Time for Okapi-BM25: 124327689223
with Query: machine learning
with Synonyms: apprehension information study simple machine debut and worst title terms: understanding data survey machine introduction machine active importance practical extreme flow support correlation machine system kernel genetic learn ml
Worst time for Okapi-BM25 Reading the info: 124327418859
Worst time for Okapi-BM25 calculating the scores: 255325
Worst time for Okapi-BM25 Sorting the list: 7519

With The Antonym Feature:
Worst Time for Okapi-BM25: 88906519678
with Query: machine learning
with Antonyms: declassification
and worst title terms: understanding data survey machine introduction machine active importance practical extreme flow support correlation machine system kernel genetic learn ml
Worst time for Okapi-BM25 Reading the info: 88906225735
Worst time for Okapi-BM25 calculating the scores:
281513 Worst time for Okapi-BM25
Sorting the list: 8305

B3models_info.idx of size ~900KB

B3metrics_info.idx of size 2.8KB

B4:

Για αυτο το ερωτημα λογω περιορισμου χρονου ακολουθησαμε μια πολυ πιο απλη τακτικη,ωστοσο οχι τελειως αδιαφορη.Αυτο που επιδιωξαμε εδω ειναι να δουμε αν μπορουμε να χρησιμοποιησουμε μονο ορους(initial Query, Synonyms of Query) απο το query μας.Εδω ακολουθησαμε δυο προσεγγισεις οπου δεν διαφερουν ιδιαιτερα μεταξυ τους.

1)Παιρνουμε τους ορους του query μας και βρισκουμε για καθε ορο και ενα συνωνυμο.Επειτα Υπολογιζουμε το score.

2)Οι συνωνυμοι οροι εχουν βαρος 2.0 αντι για 1.0.

Η Αληθεια ειναι οτι ηθελα να κανω μια παρομοια προσεγγιση σαν το B3 με αντωνυμα και συνωνυμα και επειτα να συγκρινω κατα εναν βαθμο και τα μοντελα μεταξυ τους,αλλα δεν βρηκα τροπο να παρω αντωνυμα(καποια μεθοδο).Επισης δεν θεωρω τελειως κακη ιδεα το να χρησιμοποιησω δοκιμασω και αυτην την προσεγγιση ετσι ωστε να ξερω αν ειναι καθολου βιωσιμη.

Αν ομως ηθελα να κανω κατι ωστε να κανω τις προσεγγισεις του B4 πιο ωραιες και πιο ερευνητικες(αν ειχα τον χρονο),θα δοκιμαζα να παρω οπως και στο B3 του ορους απο τον τιτλο,και επειδη το B4 ειναι μοντελο διανυσματικων συγκρισεων,θα συγκρινα το similarity των ορων(εχει συναρτηση),και οι οροι που θα ειχαν similarity μεγαλυτερου καποιου βαθμου που θα εθετα εγω, θα παιρνουν εξτρα βαρυτητα.πχ gia 0<sim<0.3 weight 1.0, 0.3<= sim <0.6 weight 2.0 και ουτω καθεξης.Αυτο που θα ηταν ιδιαιτερα αξιοσημειωτο με αυτον τον τροπο ευρεσης ορων ειναι οτι οι οροι που θα διαλεγαμε για να προσθεσουμε στο επερωτημα μας θα ειναι ΣΙΓΟΥΡΑ συγκρισιμοι και relevant μεταξυ τους για την σχεση query theme(term) <-> document term.

Παρακατω θα αναφερθουμε στα αποτελεσματα μας και στο που καταληγουμε για αυτην την απλοικη προσεγγιση.

Πριν αναφερθουμε στα αποτελεσματα να προσθεσω πως εδω εχω κανει ενα

λαθος,το οτι αυξησα το βαρος μερικων ορων το οποιο δεν θα προσφερει τιποτα στο μοντελο Okapi καθως δεν χρησιμοποιει το βαρος του ορου.Ωστοσο,τα συνωνυμα μετρανε.

Παρατηρωντας τους φακελους με τα δεδομενα B2models_results και B4_models_results μπορουμε να συγκρινουμε τα αποτελεσματα των αλλαγων μας.Μερικα δειγματα:

Για vsm:

Για Query: federated learning

B2:

Time for VSM evaluation: 13732976826
Results:
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 0.426
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 0.4017
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 0.2997
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 0.2283

B4:(Απο κατω βλεπετε τις δυο προσεγγισεις με feature==false/true)

Synonyms Used: supervalu teaching
Double Weight Feature: false
Time for VSM evaluation: 6053167643
Results:
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 0.1765
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 0.1664
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 0.1242
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 0.0946

Synonyms Used: supervalu teaching
Double Weight Feature: true
Time for VSM evaluation: 6427366591
Results:
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 0.0945
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 0.0891
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 0.0665
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 0.0507

Για Query: ehr phenotype framework

B2:

Time for VSM evaluation: 5182776552
Results:
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 0.3971
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 0.3909
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 0.3312

B4:

Synonyms Used: eh genotype implementation
Double Weight Feature: false
Time for VSM evaluation: 9657589494
Results:
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 0.2867
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 0.2823
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 0.2486

Synonyms Used: eh genotype implementation
Double Weight Feature: true
Time for VSM evaluation: 9429820802
Results:
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 0.1837
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 0.1808
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 0.1654

Για Query: clarias batrachus heavy metals

B2:

Time for VSM evaluation: 3680897923
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 0.5695
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 0.5578
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 0.5328
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 0.4944
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 0.403

B4:

Synonyms Used: penaeus light copper
Double Weight Feature: false
Time for VSM evaluation: 6370377585
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 0.4785
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 0.4538
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 0.4335
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 0.4022
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 0.3279

Synonyms Used: penaeus light copper
Double Weight Feature: true
Time for VSM evaluation: 6232803122
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 0.3479
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 0.3198
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 0.3055
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 0.2834
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 0.231

Για Query: doped zno rod solar cell

B2:

Time for VSM evaluation: 19042595914
Results:
DocumentId: 5fbe8156812d8ef52558b2964ec1bdf927be6401 Score: 0.3748
DocumentId: 88dcb2023c151abeb94fc9a87be880eca883079d Score: 0.3144
DocumentId: 5cab3a82976f58d5cf8d87bc82c0ba830829f91c Score: 0.1331

*B4:

Synonyms Used: nylon nanorods ken photovoltaic cells
Double Weight Feature: false
Time for VSM evaluation: 15064027573
Results:

DocumentId: 5cab3a82976f58d5cf8d87bc82c0ba830829f91c Score: 0.3743
DocumentId: 88dcb2023c151abeb94fc9a87be880eca883079d Score: 0.3444
DocumentId: 5fbe8156812d8ef52558b2964ec1bdf927be6401 Score: 0.2569

Synonyms Used: nylon nanorods ken photovoltaic cells
Double Weight Feature: true
Time for VSM evaluation: 14798269380
Results:
DocumentId: 5cab3a82976f58d5cf8d87bc82c0ba830829f91c Score: 0.4044
DocumentId: 88dcb2023c151abeb94fc9a87be880eca883079d Score: 0.2953
DocumentId: 5fbe8156812d8ef52558b2964ec1bdf927be6401 Score: 0.1641

Για Query: Does exogenous growth hormone improve athletic performance?

B2:

Time for VSM evaluation: 43402780003
Results:
DocumentId: 65170c43f27c6eab32fee1df468bc8ddba790257 Score: 0.3619
DocumentId: e34a3d7e41d9f5cc9930c84d5c979b755b4c868d Score: 0.1973

*B4:

Synonyms Used: endogenous increase hormones improving sporting performances
Double Weight Feature: false
Time for VSM evaluation: 44734359361
Results:
DocumentId: 65170c43f27c6eab32fee1df468bc8ddba790257 Score: 0.4261
DocumentId: e34a3d7e41d9f5cc9930c84d5c979b755b4c868d Score: 0.1804

Synonyms Used: endogenous increase hormones improving sporting performances
Double Weight Feature: true
Time for VSM evaluation: 44113277320
Results:
DocumentId: 65170c43f27c6eab32fee1df468bc8ddba790257 Score: 0.4093
DocumentId: e34a3d7e41d9f5cc9930c84d5c979b755b4c868d Score: 0.1529

Αυτα ειναι μερικα δειγματα τα οποια δεν παρθηκαν τυχαια.Εγω διαλεξα μερικα για να μπορεσω να αναφερθω σε καποια συγκεκριμενα γεγονοτα που παρατηρησα.Φυσικα ομως,περα απο αυτα τα επιλεγμενα απο μενα

αποτελεσματα θα αναφερθω και στην γενικη εικονα των αποτελεσματων αυτων των προσεγγισεων(του B4).Απλα δεν ανεβαζω πολλα δειγματα για να μην γεμισω τον φακελο με επαναλαμβανομενη πληροφορια για διαφορετικα queries.

Παρατηρωντας τα αποτελεσματα κατεληξα στο οτι αυτες οι δυο προσεγγισεις ειναι κατα κυριο λογο και πιο αργες,και μειωνουν το score των εγγραφων μας που θεωρουμε relevant.Επομενως δεν ειναι αποδεκτες σαν προσεγγιση για βελτιωση του συστηματος.Επισης παρατηρουμε εδω οτι το double Weight κανει ακομα χειροτερα την κατασταση,περιπου υποδιπλασιαζοντας το score σε συγκριση με το normal weight με τιμη 1.0 που εχουν οι λεξεις μας.Φυσικα μπορουμε να πουμε οτι ενας απο τους λογους που δεν δουλευουν αυτες οι προσεγγισεις ειναι διοτι δεν υπαρχουν πολλες λεξεις μεσα στα εγγραφα μας για να υπαρξει αξιοποιηση των εξτρα λεξεων που επεκτεινουν το επερωτημα(απο τιτλο,απο συνωνυμα κτλπ).

Αν κοιταξουμε τα δειγματα που εχω παραπανω,το πρωτο με Query: “federated learning” εχει πολυ αρνητικο αποτελεσμα.Με το απλο μοντελο VSM χωρις επεκτασεις εχουμε σκορς κοντα στο 0.4 ενω με συνωνυμα και με Double Weight feature παρατηρουμε στην πρωτη περιπτωση υποδιπλασιασμο του σκορ και στην δευτερη επεκταση(Double Weight) παρατηρουμε υποτετραπλασιασμο του αρχικου σκορ.Αυτο μπορουμε πολυ ευκολα να καταλαβουμε οτι ειναι ακριβως το αντιθετο απο αυτο που θελουμε.Τουλαχιστον,μπορω να πω πως τετοιου ειδους υποβαθμισεις ηταν σπανιες ετσι οπως μελετουσα τα αποτελεσματα.Κατα κυριο λογο ειχαμε μια μειωση της κλιμακας οπως του Query: “clarias batrachus heavy metals”.Δηλαδη μεταξυ [0.08,~0.1].Φυσικα και αυτες ειναι μη-επιθυμητες καθως δεν βελτιωνουν το αρχικο μας αποτελεσμα.Τελος,τα ευχαριστα ειναι οτι υπαρχουν και μερικες περιπτωσεις οπου οντως μια τοσο απλοικη προσεγγιση εδωσε θετικα αποτελεσματα(Τα επερωτηματα που εχουν στο B4 αστερισκο ειναι αυτα).

Στο Query:”doped zno rod solar cell”, παρατηρουμε οτι το τελευταιο document που ειχε 0.13 στο B2 στο B4 εχει 0.37.Το πρωτο ομως μειωνεται κατα ~0.12 και το μεσαιο αυξανεται κατα 0.03.Η μειωση του πρωτου δεν ειναι ευχαριστη,αλλα η γενικη εικονα βελτιωνεται,πραγμα που αξιο αναφορας.

Στο ιδιο επερωτημα,αλλα τωρα με την επεκταση Double Weight βλεπουμε οτι το τελευταιο απο το αρχικο ειναι πρωτα και αυξανεται και αλλο με την Double Weight επεκταση,ωστοσο ριχνει τα αλλα 2,πραγμα που σημαινει οτι το συνωνυμο που διαλεξαμε ενισχυει τα σκορ τους,αλλα αν του δωσουμε πολυ

μεγαλο βαρος θα αρχισει να λειτουργει με αρνητικο τροπο για τα υπολοιπα εγγραφα.

Στο Query: “Does exogenous growth hormone improve athletic performance?”

Και εδω βλεπουμε την ιδια περιπτωση με παραπανω,οπου τα συνωνυμα με Νορμαλ βαρος αυξανουμε το ενα document και μειωνουν ελαχιστα το αλλο.Ωστοσο και εδω ισχυει οτι το εξτρα βαρος στα συνωνυμα δρα τελειως αρνητικα.

Αυτες ηταν οι γενικες παρατηρησεις οσον αφορα την απλοικη προσεγγιση στην επεκταση του επερωτηματος για το μοντελο VSM στο ερωτημα Β4,μπορουμε να πουμε οτι δεν ειχε επιτυχια,και εκτος αυτου, οτι εδρασε αρνητικα στα εγγραφα που θελαμε να βελτιωσουμε.

Για το BM:

Οπως αναφεραμε παραπανω εδω η επεκταση Double Weight δεν πρεπει να επηρεαζει το μοντελο Okapi καθως δεν χρησιμοποει τα βαρη των λεξεων,αλλα το μηκος του εγγραφου,οπως και την μεση τιμη μηκους, την συχνοτητα της λεξης και το πληθος εγγραφων που περιεχουν την λεξη αυτη.

Για Query: federated learning

B2:

Time for OkapiBM25 evaluation: 7519858052
Results:
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 26.9327
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 24.75
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 18.5657
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 12.7218

B4:

Synonyms Used: supervalu teaching
Double Weight Feature: false

Time for OkapiBM25 evaluation: 7759769063
Results:
DocumentId: 276194e96ebd620b5cff35a9168bdda39a0be57b Score: 26.9327
DocumentId: 8b419080cd37bdc30872b76f405ef6a93eae3304 Score: 24.75
DocumentId: 8c442dab45400bc99ac63195a06fd531d13407fe Score: 18.5657
DocumentId: a25fbcbbae1e8f79c4360d26aa11a3abf1a11972 Score: 12.7218

Για Query: ehr phenotype framework

B2:

Time for OkapiBM25 evaluation: 6734925926
Results:
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 36.6703
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 34.8922
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 34.7084

B4:

Synonyms Used: eh genotype implementation
Double Weight Feature: false
Time for OkapiBM25 evaluation: 9354019287
Results:
DocumentId: a707e32031acf4c844b2d80089c42001c8fed790 Score: 39.6314
DocumentId: 00e17bd820ffdddfeca041f31ae1691d18f6970c Score: 36.6703
DocumentId: 05525ab2c6b4d38491dc7d740594067c6fa22b1d Score: 34.8922

Για Query: clarias batrachus heavy metals

B2:

Time for OkapiBM25 evaluation: 3578924938
Results:
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 80.4916
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 78.021
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 76.1253
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 71.6851
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 52.6821

B4:

Synonyms Used: penaeus light copper
Double Weight Feature: false
Time for OkapiBM25 evaluation: 6198630855
Results:
DocumentId: 0f7cde37e225c73827c3db5ba2dfea3ea465d56e Score: 85.2203
DocumentId: 7e485438975eaf0ac64f4277631767424df9405b Score: 80.4916
DocumentId: b173d4413f300f8482604255108b2343cf9f1812 Score: 76.1253
DocumentId: 647c960bb5ed7bc1b8d6aefca2d6e42339083df7 Score: 71.6851
DocumentId: c99ce7508aa34e88834203222babc2b519901acb Score: 52.6821

Για Query: doped zno rod solar cell

B2:

Time for OkapiBM25 evaluation: 19316172210
Results:
DocumentId: 5fbe8156812d8ef52558b2964ec1bdf927be6401 Score: 53.3154
DocumentId: 88dcb2023c151abeb94fc9a87be880eca883079d Score: 45.3624
DocumentId: 5cab3a82976f58d5cf8d87bc82c0ba830829f91c Score: 21.3145

B4:

Synonyms Used: nylon nanorods ken photovoltaic cells
Double Weight Feature: false
Time for OkapiBM25 evaluation: 14802021690
Results:
DocumentId: 88dcb2023c151abeb94fc9a87be880eca883079d Score: 68.0961
DocumentId: 5cab3a82976f58d5cf8d87bc82c0ba830829f91c Score: 58.3575
DocumentId: 5fbe8156812d8ef52558b2964ec1bdf927be6401 Score: 53.3154

Για Query: Does exogenous growth hormone improve athletic performance?

B2:

Time for OkapiBM25 evaluation: 35704605208
Results:
DocumentId: e34a3d7e41d9f5cc9930c84d5c979b755b4c868d Score: 43.6026
DocumentId: 65170c43f27c6eab32fee1df468bc8ddba790257 Score: 38.9094

B4:

Synonyms Used: endogenous increase hormones improving sporting performances
Double Weight Feature: false
Time for OkapiBM25 evaluation: 43828959662
Results:
DocumentId: 65170c43f27c6eab32fee1df468bc8ddba790257 Score: 50.2564
DocumentId: e34a3d7e41d9f5cc9930c84d5c979b755b4c868d Score: 47.5117

Για Query: burmese pythons affecting the everglades everglades

B2:

Time for OkapiBM25 evaluation: 5840894434
Results:
DocumentId: 7639d2ebe0e6efcbf39010613935c871f84aac6d Score: 54.7229
DocumentId: 18a558f43033bc02610799e1c908094a97ffa3d3 Score: 30.4145
DocumentId: e9f6afc54089adae0e8cc3d79fa2f2d395df0e1a Score: 29.5755
DocumentId: 02b039b249c95ad35d5f5441a12e77c77c39c489 Score: 18.6301

B4:

Synonyms Used: burma iguanas affected swamp swamp
Double Weight Feature: false
Time for OkapiBM25 evaluation: 5342220590
Results:
DocumentId: 7639d2ebe0e6efcbf39010613935c871f84aac6d Score: 54.7229
DocumentId: 18a558f43033bc02610799e1c908094a97ffa3d3 Score: 30.4145
DocumentId: e9f6afc54089adae0e8cc3d79fa2f2d395df0e1a Score: 29.5755
DocumentId: 02b039b249c95ad35d5f5441a12e77c77c39c489 Score: 18.6301

Για Query: nutrition and talent identification sports

B2:

Time for OkapiBM25 evaluation: 6369238386
Results:
DocumentId: 892c41f004b6a7ad939860bbee22446c01882055 Score: 55.6146
DocumentId: 4b18421585dd9c00d2543b4821fb2ce7982d2291 Score: 52.1982
DocumentId: 8733ced557e395fbc4632d49db824fa78f52e9e2 Score: 44.6965
DocumentId: 955cf9cb5fabc37a6e1a0601de0adb0f0d834663 Score: 42.4419
DocumentId: 1d296dafd70338b4824e82719dfea6b1343ddafc Score: 39.8778
DocumentId: d4cbcef63140f66cc63874cc2485e7cdadc50821 Score: 31.1725

B4:

Synonyms Used: nutritional talents identifying sport
Double Weight Feature: false
Time for OkapiBM25 evaluation: 12060814554
Results:
DocumentId: 892c41f004b6a7ad939860bbee22446c01882055 Score: 61.3073
DocumentId: 4b18421585dd9c00d2543b4821fb2ce7982d2291 Score: 52.1982
DocumentId: 955cf9cb5fabc37a6e1a0601de0adb0f0d834663 Score: 46.4636
DocumentId: 8733ced557e395fbc4632d49db824fa78f52e9e2 Score: 44.6965
DocumentId: 1d296dafd70338b4824e82719dfea6b1343ddafc Score: 39.8778
DocumentId: d4cbcef63140f66cc63874cc2485e7cdadc50821 Score: 31.1725

Αυτα ειναι μερικα δειγματα τα οποια κατα κυριο λογο επιλεχθηκαν ως:
1)Τα ιδια με το VSM για να συγκρινουμε τα αποτελεσματα των διαφορετικων μοντελων υπο ιδιες συνθηκες
2)Μερικα διαφορετικα(τυχαια επιλεγμενα) για μεγαλυτερο πληθος αποτελεσματων.

Παρατηρωντας τα αποτελεσματα μας βλεπουμε πως αυτη η προσεγγιση στο μοντελο Okapi ειχε τα ακριβως αντιθετα αποτελεσματα συγκριση με το μοντελο VSM.Σε γενικες γραμμες βλεπουμε μια βελτιωση του σκορ και αν οχι βελτιωση βλεπουμε διατηρηση της θεσης των εγγραφων μας.Αυτο που πρεπει τωρα να μας κεντρισει το ενδιαφερον ειναι,ποσο πιο αργο εγινε το Query Evaluation για να παρουμε αυτα τα καλυτερα αποτελεσματα.Εχουμε:

Για Query: ehr phenotype framework

Παρατηρουμε μια αυξηση μεγεθους [2.0,3.0] με χρονο 2.5 δευτερολεπτων παραπανω.Η ποσοτητα που αυξηθηκε δεν ειναι αρκετη ωστε να κρινουμε την επιδοση ως αποτελεσματικη.Ωστοσο δεν ειναι και η χειροτερη.

Για Query: clarias batrachus heavy metals

Παρατηρουμε μια αυξηση του δευτερου εγγραφου κατα 7 μοναδες.Ειναι μια μεγαλη βελτιωση για αυτο το συγκεκριμενο εγγραφο αλλα το κοστος που επρεπε να πληρωσουμε ειναι παλι περιπου 2.5 δευτερολεπτα.

Για Query: doped zno rod solar cell

Παρατηρουμε μια αυξηση σε ⅔ εγγραφα.Το πρωτο παραμενει σταθερο με σκορε 53.31 ενω τα αλλα 2 αυξανονται κατα 13 και 12 μοναδες το οποιο ειναι ενα αξιοσημειωτο γεγονος.Εκτος αυτου,αυτο που ειναι το πιο παραξενο ειναι οτι για καποιον λογο ο χρονος μειωθηκε κατα 4.5 δευτερολεπτα αντι να αυξηθει.Μπορει να οφειλεται στην καταπονηση της CPU αυτο αλλα δεν μπορουμε να πουμε με σιγουρια.Το μονο που μπορουμε να πουμε οτι τα B2 B4 ηταν ερωτηματα που δεν τα τρεξαμε μαζι επομενως δεν υπηρχε κατι στην RAM.

Για Query: Does exogenous growth hormone improve athletic performance?

Παρατηρουμε και εδω μια μεγαλη βελτιωση του δευτερου κατα 11.5 μοναδες περιπου και του πρωτου κατα 4 μοναδες με κοστος χρονου 8 δευτερολεπτων.Ευχαριστω βελτιωση,αλλα ο χρονος ειναι υπερβολικα πολυ μεγαλος για μια τετοια βελτιωση.

Για Query: burmese pythons affecting the everglades everglades

Παρατηρουμε οτι τα σκορ διατηρουνται αλλα το B4 ολοκληρωθηκες πιο γρηγορα.

Για Query: nutrition and talent identification sports

Βλεπουμε μια αυξηση των 1 και 3(θεσεις) εγγραφων με κοστος χρονου περιπου 6 δευτερολεπτα.Δεν ειναι ικανοποιητικο αποτελεσμα καθως τα συναφη ειναι 6 και βελτιωνονται μονο τα 2 και μαλιστα σε αρκετο χρονο. Μετα απο ολα αυτα μπορουμε να πουμε οτι μεταξυ των δυο μοντελων,η προσεγγιση αυτη ειχε καποια θετικα αποτελεσματα για το μοντελο Okapi σε συγκριση με το μοντελο VSM.Φυσικα ο χρονος αυξηθηκε πολυ αλλα σε

αντιθεση με το VSM,υπηρχαν διαφορα αποτελεσματα που ηταν βελτιωμενα σε συγκριση με αυτα της απλης εκδοσης του επερωτηματος.Τελος ολοκληρωνοντας τις παρατηρησεις μας για το ερωτημα B4 και των προσεγγισεων μας για καθε ενα απο τα μοντελα μας ακολουθουνε μερικες μετρικες και η κατακλειδα οτι αυτη η προσεγγιση ειναι μη αποτελεσματικη για το μοντελο VSM και πολυ μετριο για το Okapi.

Μια προταση που θα μπορουσαμε να κανουμε για να βελτιωθει αυτη η προσεγγιση ειναι να εκμεταλλευτουμε το γεγονος οτι το GloveModel ειναι διανυσματικο και υποστηριζει similarity checking,πραγμα που σημαινει οτι θα μπορουσαμε να παιρνομε λιγοτερους ορους οι οποιοι θα εχουν εξτρα βαρος για το vsm αναλογα με το similarity και εξτρα ορους αναλογα με το similarity για το Okapi(χωρις αλλαγη βαρους).

Επομενο βημα θα ηταν κιολας να επιδιωξουμε να διαβασουμε ορους απο τον Φακελο και να συνδυασουμε τους ορους του εγγραφου με τους ορους του query,μαζι με την παραπανω λογικη και να βγαλουμε μια σχεση η οποια θα εγγυαται οτι το query μας και οι εξτρα οροι θα ειναι Relevant(Τωρα επιλεγουμε μερικους ορους τυχαια).

Μερικες Μετρικες για το B4:

Time for whole set of queries for vsm evaluation without the Double Weight feature: 4922054543477-> ~82 minutes

Time for whole set of queries for Okapi-BM25 evaluation without the Double Weight feature: 4868812471217 -> ~81 minutes

Time for whole set of queries for vsm evaluation with the Double Weight feature: 4886454698359 -> ~81 minutes

Time for whole set of queries for Okapi-BM25 evaluation with the Double Weight feature: 4863243985041 ~81 minutes

Without The Feature:
Average time for vsm: 7751266997 -> ~7.7 sec

Worst Time for vsm: 96062094323 -> ~96 sec ->~ 1min 36 sec
With Query: Data Mining research papers on based on association rules
Worst time for vsm Reading the info: 96061793387 -> ~96 sec
Worst time for vsm calculating the scores: 292227

Worst time for vsm Sorting the list: 2494

With The Feature:
Average time for vsm: 7695204249

Worst Time for vsm: 96093666824
With Query: Data Mining research papers on based on association rules
Worst time for vsm Reading the info: 96093384630
Worst time for vsm calculating the scores: 274591
Worst time for vsm Sorting the list: 2548

Without The Feature:
Average time for Okapi-BM25: 7667421214 -> ~7.6 seconds

Worst Time for Okapi-BM25: 95400746646 -> ~95 sec -> ~1 min 35 sec
With Query: Data Mining research papers on based on association rules
Worst time for Okapi-BM25 Reading the info: 95400490124 -> ~95 sec
Worst time for Okapi-BM25 calculating the scores: 246287 -> ~0.00025 sec
Worst time for Okapi-BM25 Sorting the list: 2801

With The Feature:
Average time for Okapi-BM25: 7658651944 -> ~7.6 seconds
Worst Time for Okapi-BM25: 95342179324 -> ~95 sec -> ~1 min 35 sec
With Query: Data Mining research papers on based on association rules
Worst time for Okapi-BM25 Reading the info: 95341923123 -> ~95 sec
Worst time for Okapi-BM25 calculating the scores: 247446
Worst time for Okapi-BM25 Sorting the list: 2909

Η μεσος χρονος υπολογισμος απεχει μονο 1 δευτερολεπτο.Δεν ειναι ασχημη τιμη,αλλα για το vsm δεν υπηρχει βελτιωση ενω για το Okapi ειχαμε καποιες βελτιωσεις.

Τα αρχεια που χρησιμοποιηθηκαν για να παρθουν τα δεδομενα ειναι:

B4results_models.idx με τελικο μεγεθος(size): ~893KB
B4metrics_models.idx με τελικο μεγεθος(size): ~2.6KB

B5:

Τρεχω το documents file, διαβαζω τα hashKeys καθε doc ενα ενα και δημιουργω τα node, ταυτοχρονα τα τοποθετω και στον γραφο. Επειτα ξανατρεχω ολο το collection για να παρω τα citations καθε node,δημιουργωντας ετσι τον γραφο με τα edges.Επειτα προχωραμε στα recursions για τον υπολογισμο του page rank. Χρησιμοποιω μια νεα δομη δεδομενων, ενα HashMap που περιεχει ως κλειδι
το node Id και ως τιμη εχει ενα Pair απο δυο double , ο ενας double αντιπροσωπευει την τιμη page rank που ειχε το συγκεκριμενο node στο προηγουμενο iteration, και ο αλλος περιεχει την τιμη page rank που εχει στο τρεχον iteration. Ξεκιναμε τον υπολογισμο αρχικοποιωντας ολες τις τιμες page rank των node να ειναι ισες με 1/S , οπου S ειναι ο αριθμος των nodes στον γραφο. Στην συνεχεια, διατρεχουμε ολα τα nodes ενα-ενα(εστω το τωρινο node οτι το ονομαζουμε current) και ελεγχουμε τα citations τους, και για καθε citation προσθετουμε την τιμη pagerank που εχει ο
current node δια τον outDegree(current), αποθηκευουμε αυτον τον αριθμο στο double της page rank του τωρινου iteration του citation. Συνεχιζουμε μεχρι να εξαντληθουν τα nodes. Σε καθε iteration επιλεγουμε το node που εχει κανει την μεγαλυτερη αλλαγη και στο τελος του iteration ελεγχουμε αν η μεγαλυτερη αλλαγη ειναι μικροτερη απο το page rank threshold , αν ναι , τοτε το προγραμμα μας τερματιζει, αν ειναι ομως μεγαλυτερη σημαινει οτι πρεπει να συνεχισουμε και σε αλλο iteration.

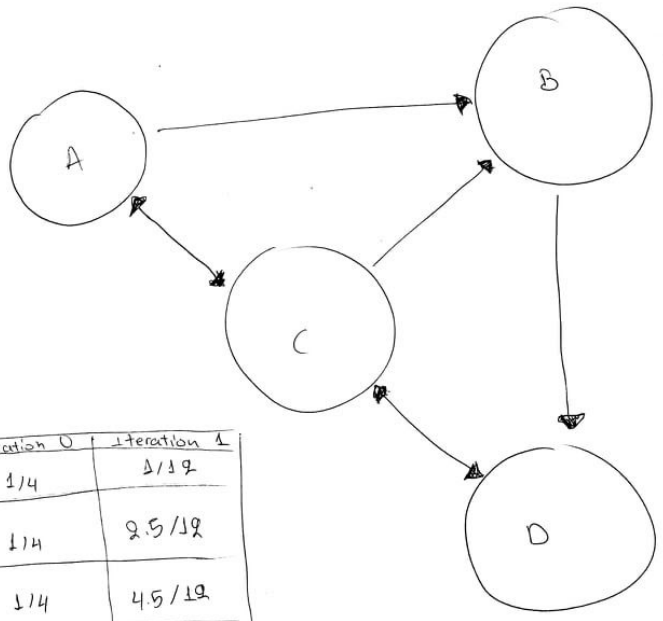
PageRank Threshold: 1.3E-5
Final PageRank value(causing termination): 1.2208E-5
Final Node ID: e36a43ac096b84593d860547589e4fd2e0e0deae
Initial PageRank value: 2.130059E-8
Number of iterations: 8
Number of Edges: 348041657
Number of Nodes: 46947044

Παρακατω βλεπουμε ενα απλο παραδειγμα ενος απλου γραφου:

Το trace του κωδικα υπολογισμου του PageRank εχει ως εξης:
Current Node = A (<0.25,0.0>|<previous_page_rank,current_page_rank>); με outDegree = 2;
  Citation = B(<0.25,0.0>|<previous_page_rank,current_page_rank>);
  current_page_rank(B) = 0.25/2 = 0.125;
  Citation = C(<0.25,0.0>|<previous_page_rank,current_page_rank>);
  current_page_rank(C) = 0.25/2 = 0.125;
Current Node = B (<0.25,0.125>|<previous_page_rank,current_page_rank>); με outDegree = 1;
  Citation = D(<0.25,0.0>|<previous_page_rank,current_page_rank>);
  current_page_rank(D) = 0.25/1 = 0.25;
Current Node = C(<0.25,0.125>|<previous_page_rank,current_page_rank>); με outDegree = 3;
  Citation = A (<0.25,0.0>|<previous_page_rank,current_page_rank>);
  current_page_rank(A) = 0.25/3 = 0.083;
  Citation = B (<0.25,0.125>|<previous_page_rank,current_page_rank>);
  current_page_rank(B) = 0.125 (current_page_rank(B)) + 0.25/3 = 0.125 + 0.083 = 0.208;
  Citation = D(<0.25,0.25>|<previous_page_rank,current_page_rank>);
  current_page_rank(D) = 0.25(current_page_rank) + 0.25/3 = 0.25 + 0.083 = 0.333
Current Node = D(<0.25,0.333>|<previous_page_rank,current_page_rank>);
  Citation = C(<0.25,0.125>|<previous_page_rank,current_page_rank>);
  current_page_rank(C) = 0.125(previous_page_rank) + 0.25/1 = 0.375

Now we are going to reset
Set previous_page_rank of each node equal to current_page_rank and reset current_page_rank to 0.0.
So hashmap looks like
A<0.083,0.0>
B<0.208,0.0>
C<0.375,0.0>
D<0.333,0.0>

Check if the max change is smaller than Page Rank Threshold.
If it is, write the page ranks in the disk.
If it isn't, re-compute page ranks by calling the same method.



	Iteration 0	Iteration 1
A	1/4	1/12
B	1/4	2.5/12
C	1/4	4.5/12
D	1/4	4/12

B6:

Χρησιμοποιωντας το Okapi και απλα αλλαζοντας τον τυπο κατα τον οποιο υπολογιζουμε το score (προσθετωντας και του PageRank στον τυπο αυτο αλλα και τις παραμετρους των weights) υπολογιζουμε τα νεα score. Για να ελενξουμε αν αυτο βοηθαει στην καλυτερη κατανομη των αποτελεσματων ενος query τρεχουμε το προγραμμα μας με TYPE = FULL ετσι ωστε να μπορουμε να συγκρινουμε τα εγγραφα τα οποια ειναι μεσα στα αποτελεσματα καθε query. Ας δουμε μερικα παραδειγματα:
Τρεχωντας με PageRank Weight: 0.3 και RetrievalModel Weight: 0.7
Ας ξεκινησουμε με την πρωτη ερωτηση, βλεπουμε οτι απτο ερωτημα B2 εχουμε τα εξης αποτελεσματα:
Query:Mal/Tirap
1) 9e5e226fe10becab0d0793cff4dca5fc4a0b5aaf
2) c04a2c5d59d793a42750c842dfc6e7eb1bc93ab9
3) 1d464ea76572e85603b4fe607f09c3953fef1aa9

ενω το B6 μας δινει τα αποτελεσματα :

1) 1d464ea76572e85603b4fe607f09c3953fef1aa9 (3ο στο Okapi)
2) c04a2c5d59d793a42750c842dfc6e7eb1bc93ab9 (2ο στο Okapi)
3) 9e5e226fe10becab0d0793cff4dca5fc4a0b5aaf (1ο στο Okapi)

Παρατηρωντας τους τιτλους των εγγραφων, βλεπουμε οτι και οι τρεις περιεχουν την λεξη Tirap αλλα κανενα την λεξη Mal, εδω ειναι που θα μας βοηθησει το PageRank σε συνδυασμο με το Okapi για να δουμε πιο εγγραφο ειναι πιο εγκυρη πηγη και ποιο θα επρεπε να εχουμε καταταξει στην πρωτη θεση. Ενω εχουν μικρες διαφορες στο page rank επηρεαζει πολυ την τελικη τους καταταξη.

Παμε τωρα σε ενα επομενο παραδειγμα, στο οποιο φαινεται το συστημα μας να μην δουλευει σωστα.
Query: federated learning
Εδω το B2 μας δινει τα εξης αποτελεσματα:

1) 276194e96ebd620b5cff35a9168bdda39a0be57b
2) 8b419080cd37bdc30872b76f405ef6a93eae3304
3) 8c442dab45400bc99ac63195a06fd531d13407fe
4) a25fbcbbae1e8f79c4360d26aa11a3abf1a11972

ενω το B6 μας δινει:

1) 8b419080cd37bdc30872b76f405ef6a93eae3304
2) 276194e96ebd620b5cff35a9168bdda39a0be57b
3) a25fbcbbae1e8f79c4360d26aa11a3abf1a11972
4) 8c442dab45400bc99ac63195a06fd531d13407fe

Παλι βλεπουμε καποιες αλλαγες, με το PageRank αλλαζουν τα εγγραφα 1 -> 2 θεση, το οποιο ειναι σωστο, καθως και τα δυο εγγραφα δειχνουν
να ειναι σε μεγαλο βαθμο σχετικα με το query μας, ομως εχουν μεγαλη διαφορα στο page rank , οποτε η αλλαγη μας εδω φαινεται σωστη.
Ομως τα αλλα δυο εγγραφα μας αλλαζουν και αυτα θεσεις, το οποιο ειναι λαθος καθως ενω το 4ο (του B6) εγγραφο περιεχει ολες τις λεξεις
που εχει το query μας στον τιτλο του, το 3ο εγγραφο (του B6) του πηρε την θεση λογω της μεγαλης διαφορας που εχουν στο Page Rank, ναι μεν μπορει
να ειναι καλη πηγη το 3ο εγγραφο αλλα το 4ο φαινεται πιο σχετικο με το query μας.

Ας παμε σε ενα επομενο παραδειγμα:
Εδω εχουμε ενα μεγαλο query, οποτε ειναι και μεγαλυτερος ο αριθμος των αποτελεσματων.
Query: clarias batrachus heavy metals
Στο συγκεκριμενο Query το B2 μας δινει:

1) 7e485438975eaf0ac64f4277631767424df9405b
2) 0f7cde37e225c73827c3db5ba2dfea3ea465d56e
3) b173d4413f300f8482604255108b2343cf9f1812
4) 647c960bb5ed7bc1b8d6aefca2d6e42339083df7
5) c99ce7508aa34e88834203222babc2b519901acb

ενω το B6 μας δινει:

1) c99ce7508aa34e88834203222babc2b519901acb
2) 647c960bb5ed7bc1b8d6aefca2d6e42339083df7
3) 7e485438975eaf0ac64f4277631767424df9405b
4) 0f7cde37e225c73827c3db5ba2dfea3ea465d56e
5) b173d4413f300f8482604255108b2343cf9f1812

Εδω παρατηρουμε παρα πολλες αλλαγες στην καταταξη μας.
Εδω βλεπουμε οτι η πρωτη θεση δεν ειναι σωστη καθως, το 2ο εγγραφο (του B6) περιεχει ολες τις λεξεις του query στον τιτλο του, και του
παιρνει τι θεση το 1ο εγγραφο λογω της διαφορας που υπαρχει στο page rank.
Ομοιως, το ιδιο συμβαινει και με το 4ο εγγραφο, το οποιο

περιεχει ολες τις λεξεις του query, αλλα τερματιζει πολυ χαμηλα στην καταταξη.

Ας δουμε και ενα ακομη συντομο παραδειγμα
Query:cifar
Εδω και τα δυο εγγραφα που εμφανιζονται στην καταταξη περιεχουν την λεξη στον τιτλο τους.
Το B2 συστημα μας δινει:
1) f445493badf53febbaeab340a4fca98d9e4ab7f7
2) bea5780d621e669e8069f05d0f2fc0db9df4b50f

Ενω το B6 συστημα μας δινει:
1) bea5780d621e669e8069f05d0f2fc0db9df4b50f
2) f445493badf53febbaeab340a4fca98d9e4ab7f7

Εδω η αλλαγη μας φαινεται σωστη, καθως και τα δυο εγγραφα περιεχουν την λεξη του query, οποτε την πρωτη θεση πρεπει να την παρει η πιο
εγκυρη πηγη, δηλαδη το εγγραφο με την μεγαλυτερη τιμη στο page rank, δηλαδη το 1ο εγγραφο (του B6)

Υπαρχουν βεβαια και παραδειγματα που δεν υπαρχει καμια αλλαγη και τα συστηματα μας συμφωνουν.
Οπως συμβαινει στο query: ehr phenotype framework στο οποιο query και τα δυο μας συστηματα εμφανιζουν τα αποτελεσματα:
1) 00e17bd820ffdddfeca041f31ae1691d18f6970c
2) 05525ab2c6b4d38491dc7d740594067c6fa22b1d
3) a707e32031acf4c844b2d80089c42001c8fed790

Παραπανω ειδαμε παραδειγματα εκ των οποιων δεν χρειαστηκε καμια αλλαγη στην καταταξη και τα συστηματα μας απο τα ερωτηματα B2 και B6
συμφωνουσαν, ή ειδαμε και παραδειγματα τα οποια κανουν πολλες αλλαγες, αλλα ειναι σωστες αλλαγες, με βαση των συγκεκριμενων queries,
ομως κατα πλειοψηφια οι αλλαγες βασιζονται περισσοτερο στην διαφορα του Page Rank που υπαρχει στα εγγραφα, παρα στο κανονικο score,
οποτε θα τωρα θα δοκιμασουμε να τρεξουμε το ιδιο προγραμμα απλα με αλλα Page Rank και Retrieval Model weights.
Ας δοκιμασουμε με PageRank Weight = 0.2 , Retrieval Model Weight = 0.8
Για να μην επαναλμβανομαστε ας δουμε μερικα απο τα παραδειγματα που αναφεραμε παραπανω:

Για το query: clarias batrachus heavy metals βλεπουμε διαφορα ,καθως , το νεο μας αποτελεσμα εχει διαφορετικη καταταξη απο
το παλιο B6:
1) 7e485438975eaf0ac64f4277631767424df9405b
2) 647c960bb5ed7bc1b8d6aefca2d6e42339083df7
3) 0f7cde37e225c73827c3db5ba2dfea3ea465d56e
4) c99ce7508aa34e88834203222babc2b519901acb
5) b173d4413f300f8482604255108b2343cf9f1812

Για το quert: federated learning εχουμε τα εξης καινουρια αποτελεσματα:
1) 8b419080cd37bdc30872b76f405ef6a93eae3304
2) 276194e96ebd620b5cff35a9168bdda39a0be57b
3) a25fbcbbae1e8f79c4360d26aa11a3abf1a11972
4) 8c442dab45400bc99ac63195a06fd531d13407fe
Εδω παρατηρουμε οτι τα αποτελεσματα δεν αλλαξαν, ενω αλλαξαμε το βαρος του Page Rank παλι δεν καταφερε το εγγραφο 4 να περασει
το εγγραφο 3, καθως η διαφορα τους στο Page Rank ειναι αρκετα μεγαλη και δεν την καλυβει η διαφορα τους στο Okapi Score.

Ας δουμε ακομη ενα τελευταιο παραδειγμα το οποιο δεν ειχαμε δει απο πανω:
Query: gsm arduino
Το B2 μας δινει:
1) ee47eedc17bf770d3ae4866a3d5ff26c023678cd
2) 087c0539f52c4a206f401385d7ffbfc418875af3
3) 7815b52db66ab49a0ed70ccb12aa436845bb4499

Το οποιο παραμενει και μετα το B6 συστημα αναλλοιωτο, καθως η διαφορα στο score υπερβαινει την διαφορα των page rank των
εγγραφων.

Οπως βλεπουμε εδω, εχει αλλαξει η καταταξη και το εγγραφο νουμερο 4, με βαση την απο πανω καταταξη εχει κατεβει 3 θεσεις,
το οποιο και επιδιωκαμε, αφου μειωσαμε το weight του Page Rank

Επειτα κραταμε και καποιες τιμες στον δισκο, για να ξερουμε καποιους χρονους, μεγεθη κλπ.:
Type: FULL
PageRank weight parameter: 0.3
Retrieval model weight parameter: 0.7
Average time per query: 2326447731
Best query: computer

-Evaluation Time: 2192
-Read time: 0
-Score: 0

Worst query: style of presentation of the results obtained using Coq
-Evaluation Time: 93939973718
-Read time: 93939647977
-Score: 302580

Αξιοσημειωτο το οτι το καλυτερο μας query (σε θεμα χρονου παντα) ειναι το "computer", για το οποιο δεν υπαρχει ουτε ενα
αποτελεσμα σε μια συλλογη 120+ GB,απο κει προκυπτουν και τα μηδενικα σε Read time και Score, εγω σοκαριστηκα.