Αναφορά – Β φάση

Implementation

A) Δομή του index

Κάθε ευρετήριο αποτελείται από 6 αρχεία:

documents_meta.idx:	postings.idx
Document ID (integer) Weight Max term frequency (max TF) Number of tokens Citations pagerank Authors pagerank Size (size in documents.idx) Offset to documents.idx	term frequency (DF) document ID (integer)
documents.idx:	**vocabulary.idx**
Year Title size Author size Author IDs size Journal name size Title Author Author IDs Journal name	term document frequency (DF) offset to postings.idx
documents_id.idx	**meta.idx**
docID (string)	avgdl pagerank damping factor use_stopwords use_stemmer number of documents pagerank_threshold index timestamp

Οι εγγραφές στα documents_id.idx, documents_meta.idx, documents.idx εμφανίζονται με τη σειρά ανάγνωσης των JSON εγγραφών κατά τη δημιουργία του ευρετήριου. Αυτή η σειρά παράγει άμεσα και έναν αύξων αριθμό ο οποίος είναι το document ID (integer) που εμφανίζεται στα postings.idx και documents_meta.idx. Επιπλέον, για κάθε όρο τα postings του εμφανίζονται πάλι με αύξουσα σειρά document ID.

Για να διαβάσουμε δεδομένα από το ευρετήριο ακολουθείται η παρακάτω διαδικασία:

1. Το vocabulary.idx φορτώνεται στη μνήμη σε κατάλληλη δομή η οποία μας επιτρέπει να διαβάσουμε το document frequency (DF) καθώς και το postings.idx offset.
2. Το documents_meta.idx, documents_id.idx γίνονται memory mapped άρα έχουμε άμεση πρόσβαση.
3. Γνωρίζουμε ότι οι εγγραφές στο postings.idx είναι σταθερού μεγέθους. Αρα τα postings για κάποιο όρο θα έχουν μέγεθος (DF) * (posting size)
4. Από το postings μπορούμε να διαβάσουμε το document ID (integer). Επιπλέον γνωρίζουμε ότι οι εγγραφές στο document_meta.idx ανά document έχουν σταθερό μέγεθος. Αρα το offset στο documents_meta.idx είναι (document ID) * (document_meta entry size).
5. Διαβάζουμε από το documents_meta.idx και βρίσκουμε το offset στο documents.idx αλλά και το μέγεθος της εγγραφής σε αυτό για κάθε document.
6. Για να διαβάσουμε το doc_id (40 char string) από το documents_id.idx μας αρκεί το document ID (integer) γιατί κάθε doc_id έχει σταθερό μέγεθος. Αρα το doc_id θα έχει offset 40 * (document ID)

B) Εκτέλεση

Η κεντρική κλάση είναι η Themis από την οποία έχουμε τη δυνατότητα:

- Είτε να συμπληρώσουμε κώδικα και να καλέσουμε τις κλάσεις που εμείς θέλουμε.
- Είτε να περάσουμε ως πρώτη παράμετρο το string ‘gui’ ώστε να ενεργοποιήσουμε το GUI που παρέχει πρόσβαση στις περισσότερες λειτουργίες που χρειαζόμαστε.

Για όσες παραμέτρους δεν υπάρχει η δυνατότητα αλλαγής από το GUI, η αλλαγή γίνεται από το config file.

C) Config

Το config διαβάζεται πάντα εκ νέου όταν:

- Φορτώνεται το index στη μνήμη.
- Δημιουργείται νέο index.

D) Αναζήτηση (Class Search)

a) Ο χρήστης μπορεί να διαλέξει να δει οποιαδήποτε από τα παρακάτω πεδία για ένα document.

```
TITLE, AUTHORS_NAMES, AUTHORS_IDS, YEAR, JOURNAL_NAME,
CITATIONS_PAGERANK, VSM_WEIGHT, TOKEN_COUNT, MAX_TF, DOCUMENT_SIZE
```

Η αναζήτηση μέσω GUI εμφανίζει μόνο τα top-10 αποτελέσματα. Στην περίπτωση που το μοντέλο αναζήτησης είναι ένα εκ των VSM/Okapi, μετά την τελική διάταξη των αποτελεσμάτων επιστρέφονται μόνο τα top-50 αποτελέσματα. Αλλιώς επιστρέφονται όλα τα αποτελέσματα. Για την ώρα η αλλαγή αυτών των περιορισμών δε μπορεί να γίνει από το GUI αν και είναι υπολοποιημένες οι απαραίτητες συναρτήσεις.

Αναζήτηση μπορεί να γίνει χρησιμοποιώντας παράλληλα και κάποιο query expansion dictionary. Για κάθε όρο του αρχικού query προστίθεται 1 επιπλέον όρος με βάρος 0.5.

Για τη δημιουργία του τελικού query, οι όροι που εμφανίζονται πολλαπλές φορές συνενώνονται σε ένα όρο ο οποίος έχει τελικό βάρος το άθροισμα όλων των βαρών.

E) Αξιολόγηση

Κάθε νέα αξιολόγηση σώζεται στο INDEX_PATH με το προκαθορισμένο όνομα που εμφανίζεται στο config file. Ο διαχωρισμός των διαφορετικών evaluation μεταξύ τους γίνεται ως:

- Κάθε αρχείο έχει το timestamp της ημερομηνίας που ξεκίνησε η αξιολόγηση στο τέλος του ονόματος του.
- Κάθε αρχείο περιλαμβάνει στην αρχή του όλο το configuration για τη συγκεκριμένη αξιολόγηση, πχ stemming, query expansion model κλπ.

F) Indexing

Κατά τη διαδικασία του indexing παράγονται διάφορα προσωρινά αρχεία τα οποία χρησιμοποιούνται σε μετέπειτα στάδια με κύριο στόχο τη μείωση του συνολικού χρόνου και της μνήμης που απαιτείται. Η σειρά των βημάτων είναι τέτοια ώστε να ελαχιστοποιείται το overhead σε χώρο στη μνήμη/δίσκο.

Αναλυτικά η διαδικασία έχει ως:

1. Ανάγνωση των αρχείων της συλλογής και δημιουργία partial index. Για κάθε document γίνονται οι κατάλληλες εγγραφές στα documents.idx, documents_meta.idx, documents_id.idx ενώ κρατάμε στη μνήμη τα κατάλληλα δεδομένα που προορίζονται για τα vocabulary.idx, posting.idx. Παράλληλα για κάθε document γίνεται εγγραφή σε προσωρινό αρχείο όλων των <term, TF>. Το αρχείο αυτό χρησιμοποιείται κατά τον υπολογισμό των βαρών για το Vector space model.
2. Εγγραφή των partial vocabulary/posting στο δίσκο. Η ταξινόμηση των όρων γίνεται μόνο μια φορά, ακριβώς πριν γίνουν οι εγγραφές στο δίσκο.
3. Ακολουθεί το merge των των partial vocabularies το οποίο γίνεται σε ένα πέρασμα. Παράλληλα για κάθε όρο γίνεται εγγραφή σε προσωρινό αρχείο όλων των <partial index ID, DF> στα οποία εμφανίζεται. Το αρχείο αυτό χρησιμοποιείται κατά το merge των partial postings.
4. Τα partial vocabularies σβήνονται όλα μετά το τέλος του merge τους.
5. Ακολουθεί ο υπολογισμός των βαρών για το Vector space model. Εδώ χρησιμοποιείται το προσωρινό αρχείο που δημιουργήθηκε στο 1 το οποίο και σβήνεται μετά το πέρας της διαδικασίας. Για την επιτάχυνση της διαδικασίας το documents_meta.idx γίνεται memory mapped.
6. Ακολουθεί το merge των partial postings σε ένα πέρασμα. Εδώ χρησιμοποιείται το προσωρινό αρχείο που δημιουργήθηκε στο 4 το οποίο και σβήνεται μετά το πέρας της διαδικασίας.
7. Τα partial postings σβήνονται όλα μετά το τέλος του merge τους.
8. Υπολογισμός των Pagerank σκορ.

G) Pagerank

Η διαδικασία περιλαμβάνει μόνο τον υπολογισμό των σκορ με βάση το γράφο των citations. Στον γράφο δε συμπεριλαμβάνονται ακμές που δείχνουν στον ίδιο κόμβο ενώ διπλές ακμές προς άλλους κόμβους θεωρούνται ως μια ακμή.

Για την μείωση των απαιτήσεων σε μνήμη και χρόνο, ακολουθήκαν τα παρακάτω βήματα:

1. Αρχικά το documents_id.idx γίνεται memory mapped και κάθε document ID (string) αντιστοιχείται σε ένα int ID. Αυτό θα μας επιτρέψει να χρησιμοποιήσουμε arrays για την αποθήκευση του γράφου αντί hash map.
2. Στη συνέχεια γίνεται ένα πέρασμα πάνω από τη συλλογή το οποίο έχει διπλό σκοπό. 1^{ον} να φιλτράρει τα document ID τα οποία δεν εμφανίζονται καθόλου στη συλλογή. 2^{ον} να γράψει όλα τα απαραίτητα δεδομένα του γράφου σε προσωρινό αρχείο. Από τη στιγμή που έχουμε την αντιστοίχηση string -> int από το 1, οι εγγραφές στο αρχείο αυτό περιλαμβάνουν μόνο ακέραιους.
3. Στη συνέχεια αρχικοποιούμε το γράφο διαβάζοντας τα δεδομένα που γράψαμε στο 2. Ο γράφος αποτελείται από ένα array από κόμβους. Κάθε κόμβος αποθηκεύει το σκορ του, το πλήθος των OUT κόμβων, και ένα array με τους IN κόμβους.
4. Ακολουθεί ο υπολογισμός των σκορ.

H) Επιπλέον λειτουργικότητα

- Κλάση S2TextualEntryCharHistogram: Διαβάζει μια συλλογή και δημιουργεί ένα ιστόγραμμα των χαρακτήρων που εμφανίζονται σε κάθε document και της συχνότητας εμφάνισης τους. Είναι χρήσιμη αν θέλουμε να βρούμε delimiters για tokenization των documents.
- Κλάση CreateCitationsGraph: Διαβάζει μια συλλογή, δημιουργεί τον πλήρες γράφο των citations και υπολογίζει στατιστικά στοιχεία για αυτόν.

Indexing the collection (108GB, 47M documents)

Σύστημα:

- Intel i9-i9900k
- 64GB RAM
- 1GB SSD

Δημιουργήθηκαν 2 ευρετήρια, με stemming και δίχως. Τα stopwords ήταν πάντα ενεργοποιημένα.

Κάθε partial index περιλάμβανε 200K documents (συνολο 235 partial indexes).

Για το pagerank χρησιμοποιήθηκε damping factor = 0.85 και threshold = 1e-8 < 1/#documents = 2e-8. Ο γράφος των citations είναι αραιός: 30M sinks και 348M OUT ακμές.

Stemming enabled

Μεγέθη τελικών αρχείων	Χρόνοι
Documents_meta.idx: 2.09GB Documents_id.idx: 1.74GB Documents.idx: 9.47GB Vocabulary.idx: 356MB Postings.idx: 26.5GB Total: 40.2GB	Partial indexes + documents*.idx: 1hr 39m Merge partial vocabularies: 1m 16s Merge partial postings: 2m 7s VSM weights: 14m 8s Create pagerank graph: 38m 6s Pagerank iterations: 45 → 4m 45s Total: 2h 40m

Για τα ενδιάμεσα αρχεία χρειάστηκαν επιπλέον 16.2GB στο δίσκο (πέραν του χώρου που καταλαμβάνει το τελικό Index).

Stemming disabled

Μεγέθη τελικών αρχείων	Χρόνοι
Documents_meta.idx: 2.09GB Documents_id.idx: 1.74GB Documents.idx: 9.47GB Vocabulary.idx: 401MB Postings.idx: 28.4GB Total: 42.1GB	Partial indexes + documents*.idx: 1hr 26m Merge partial vocabularies: 1m 25s Merge partial postings: 2m 34s VSM weights: 16m 34s Create pagerank graph: 37m 59s Pagerank iterations: 45 → 4m 55s Total: 2h 30m

Για τα ενδιάμεσα αρχεία χρειάστηκαν επιπλέον 23.4GB στο δίσκο (πέραν του χώρου που καταλαμβάνει το τελικό Index).

Όλα τα αποτελέσματα βρίσκονται στο φάκελο results.

Evaluation

Stemming enabled

	Avg. of Avg. precision	Avg. of nDCG	Avg. search time per 1M results	Avg. search time per query
VSM	**0.7003**	**0.8046**	**0.75s**	**1.97s**
VSM/Pagerank	0.7032	0.8073	0.78s	2.0s
VSM/Glove	0.7012	0.8054	0.8s	2.8s
VSM/extJWNL	0.7011	0.8054	0.75s	2.5s
VSM/Glove/Pagerank	0.7046	0.8084	0.82s	2.9s
VSM/extJWNL/Pagerank	0.7042	0.8082	0.77s	2.6s
Okapi	**0.7063**	**0.8084**	**0.51s**	**1.34s**
Okapi/Pagerank	0.7159	0.8169	0.72s	1.88s
Okapi/Glove	0.7069	0.8085	0.55s	1.93s
Okapi/extJWNL	0.7074	0.8091	0.5s	1.68s
Okapi/Glove/Pagerank	0.7163	0.8169	0.77s	2.7s
Okapi/extJWNL/Pagerank	0.7169	0.8180	0.72s	2.4s

Stemming disabled

	Avg. of Avg. precision	Avg. of nDCG	Avg. search time per 1M results	Avg. search time per query
VSM	**0.6994**	**0.8030**	**0.77s**	**1.33s**
VSM/Pagerank	0.7025	0.8059	0.76s	1.3s
VSM/Glove	0.6987	0.8029	0.81s	2.0s
VSM/extJWNL	0.7002	0.8040	0.74s	1.54s
VSM/Glove/Pagerank	0.7034	0.8069	0.84s	2.1s
VSM/extJWNL/Pagerank	0.7032	0.8069	0.78s	1.62s
Okapi	**0.7043**	**0.8053**	**0.53s**	**0.92s**
Okapi/Pagerank	0.7134	0.8133	0.76s	1.31s
Okapi/Glove	0.6997	0.8018	0.6s	1.51s
Okapi/extJWNL	0.7043	0.8051	0.52s	1.09s
Okapi/Glove/Pagerank	0.7111	0.8114	0.8s	2.0s
Okapi/extJWNL/Pagerank	0.7126	0.8129	0.72s	1.51s

Conclusion

Απ' τη στιγμή που οι διαφορές μεταξύ των σκορ ανά μέθοδο αναζήτησης είναι πολύ μικρές, η παρακάτω ανάλυση αφορά όλες τις μεθόδους συγκεντρωτικά.

Για τα queries με μηδενικό σκορ κάποιες από τις αιτίες είναι:

- Οι όροι του query δεν εμφανίζονται στα documents που έχουν κριθεί ως συναφή, πχ. υπάρχει query 'genomestrip' αλλά μόνο ο όρος 'genome' εμφανίζεται σε αυτά. Εδώ ενδεχομένως να βοηθούσε κάποιο pattern matching.

- Οι όροι του query εμφανίζονται στα documents διατυπωμένοι διαφορετικά, πχ ψάχνουμε για ‘emule’ ενώ στα documents εμφανίζεται ο όρος ‘eDonkey’. Εδώ ενδεχομένως να βοηθούσε ένα διαφορετικό query expansion.
- Δεν υπάρχει επαρκής πληροφορία στα documents που έχουν κριθεί ως συναφή που να επιτρέπει να κάνουμε σωστή αναζήτηση, πχ. η μόνη πληροφορία που εμφανίζεται είναι abstract: ‘This collaboratively edited knowledgebase provides a common source of data for Wikipedia, and everyone else’ και title: ‘Wikidata: a free collaborative knowledgebase’ με query: ‘blazegraph’ ο οποίος αφορά μια συγκεκριμένη graph database.
- Οι όροι του query δεν εμφανίζονται στα documents που έχουν κριθεί ως συναφή και μόνο από το περιεχόμενο τους μπορούμε να συμπεράνουμε ότι είναι συναφή. Πχ ψάχνουμε για ‘spirotube’ το οποίο είναι το όνομα ενός device για χρήση σε ασθενείς με άσθμα ενώ τα συναφή documents αναφέρουν μόνο τον όρο ‘asthma’. Εδώ ενδεχομένως να βοηθούσε ένα διαφορετικό query expansion.
- Οι όροι του query έχουν ορθογραφικά λάθη, πχ. ‘similarty’ αντί για ‘similarity’. Εδώ θα βοηθούσε κάποιο pattern matching.
- Τα documents έχουν ορθογραφικά λάθη. Αυτό φαίνεται να συμβαίνει συχνά. Εδώ θα βοηθούσε κάποιο pattern matching.
- Οι όροι του query εμφανίζονται στα documents που έχουν κριθεί ως συναφή ανακατεμένοι με άλλα σύμβολα πχ ψάχνουμε για ‘Thiahelicene’ ενώ στα documents εμφανίζεται ο ‘tetrathia[9]helicene’ τον οποίο εμείς σπάσαμε κατά τη διάρκεια του Indexing σε ‘tetrathia’, ‘9’, ‘helicene’. Εδώ ενδεχομένως να βοηθούσε κάποιο pattern matching ή proximity search.

Αλλα πιθανά σενάρια που δικαιολογούν χαμηλά σκορ:

- Κάποια queries έχουν delimiters τα οποία εμείς έχουμε ανγοήσει. Εδώ θα βοηθούσε κάποιο pattern matching ή αύξηση των split delimiters.
- Δεν εκμεταλλευόμαστε το proximity search το οποίο θα βοηθούσε αρκετά αντί να ψάχνουμε απλά για τους όρους του query σε οποιαδήποτε θέση στα documents.
- Χρησιμοποιούμε πάρα πολλά split delimiters, πχ σπάσαμε hyphenated λέξεις στα δύο ενώ δεν έπρεπε. Εδώ ενδεχομένως να βοηθούσε κάποιο proximity search ή η αποφυγή του διαχωρισμού hyphenated κατά περίπτωση.

Γενικότερα φαίνεται ότι το stemming/query expansion/pagerank βοήθησαν ελάχιστα. Οσον αφορά το pagerank αυτό ήταν αναμενόμενο μιας και ο γράφος είναι αραιός. Το stemming γίνεται με τρόπο που λέξεις που δε σχετίζονται καταλήγουν τελικά στην ίδια ρίζα. Το query expansion γίνεται απλοϊκά με την πρόσθεση στο query μόνο του πρώτου όρου από το expansion dictionary.

Τέλος, θα μπορούσαμε να βελτιώσουμε τα αποτελέσματα με την εξαγωγή επιπλέον tokens κατά τη διάρκεια του indexing από το κύριο σώμα των documents (χρήση των JSON πεδίων s2PdfUrl/s2Url).