Information Retrieval SS 2024

Exercise 1: Boolean Retrieval, Tolerant Retrieval[1]

Prof. Goran Glavas, Benedikt Ebing, Fabian David Schmidt
Chair for Natural Language Processing

Boolean Retrieval - Inverted Index

Consider the following document collection[2]:

Doc 1 Facebook's new tool is called Graph Search.
Doc 2 A new social graph for LinkedIn users.
Doc 3 Find friends using search on Facebook.
Doc 4 Google's Knowledge Graph lets you search for things, people, or places

1. Draw the term-document incidence matrix for this document collection.
2. Draw the inverted index representation for this document collection, showing the dictionary and the postings.
3. Use both the term-document incidence matrix and the inverted index to compute the results return for the following queries:
   - graph AND search
   - graph AND NOT (google OR facebook)
4. For a conjunctive query, is processing postings lists in order of size guaranteed to be optimal? Explain why it is, or give an example where it isn't.
5. What is the complexity (in big O notation) for a query x AND y when the postings lists are sorted? What if they aren't?
6. How should the Boolean query x AND NOT y be handled? Why is the naive evaluation of the query, i.e. evaluating NOT y first and then x AND NOT y, normally very expensive? How expensive is it (in big O notation)? How expensive is a Boolean query x OR y?
7. For a conjunctive query, is processing postings lists in order of size guaranteed to be optimal? Explain why it is, or give an example where it isn't.

Boolean Retrieval - Skip Pointers

1. Why are skip pointers not useful for queries in the form x OR y?
2. We have a two-word query. For one term the postings list consists of the following 16 entries:

$$[4, 6, 10, 12, 14, 16, 18, 20, 22, 32, 47, 81, 120, 122, 157, 180]$$

[1] Exercise tasks based on "An Introduction to Information Retrieval" by Manning, Raghavan and Schütze

[2] Assume the text is preprocessed by lowercasing and stopword removal (`https://www.ranks.nl/stopwords`). Also consider *Facebook's* as two tokens: *Facebook* and *'s*.

and for the other one it is the one entry postings list:

[47]

Work out how many comparisons would be done to intersect the two postings lists with the following two strategies. Briefly justify your answers:

(i) Using standard postings lists

(ii) Using postings lists stored with skip-pointers, with a skip length of $\sqrt{L}$, as suggested in the second lecture (s.35).

3. Consider a postings intersection between this postings list, with skip pointers:

and the following intermediate result postings list (which hence has no skip pointers:

3 5 89 95 97 99 100 101

(a) How often is a skip pointer followed?

(b) How many postings comparisons will be made by this algorithm while intersecting the two lists?

(c) How many postings comparisons are made if the postings lists are intersected without the use of skip pointers?

Phrase and Proximity Queries

1. Shown below is a portion of a positional index in the format: term: doc1: ⟨position1, position2,...⟩; doc2: ⟨ position1, position2, ... ⟩; etc.

angels: 2: ⟨36,174,252,651⟩; 4: ⟨12,22,102,432⟩; 7: ⟨17⟩;
fools: 2: ⟨1,17,74,222⟩; 4: ⟨8,78,108,458⟩; 7: ⟨3,13,23,193⟩;
fear: 2: ⟨87,704,722,901⟩; 4: ⟨13,43,113,433⟩; 7: ⟨18,328,528⟩;
in: 2: ⟨3,37,76,444,851⟩; 4: ⟨10,20,110,470,500⟩; 7: ⟨5,15,25,195⟩;
rush: 2: ⟨2,66,194,321,702⟩ 4: ⟨9,69,149,429,569⟩; 7: ⟨4,14,404⟩;
to: 2: ⟨47,86,234,999⟩; 4: ⟨14,24,774,944⟩; 7: ⟨199,319,599,709⟩;
tread: 2: ⟨57,94,333⟩; 4: ⟨15,35,155⟩; 7: ⟨20,320⟩;
where: 2: ⟨67,124,393,1001⟩; 4: ⟨11,41,101,421,431⟩; 7: ⟨16,36,736⟩;

Which document(s), if any, meet each of the following queries, where each expression within quotes is a phrase query

(i) “fools rush in"

(ii) “fools rush in" AND “angels fear to tread"

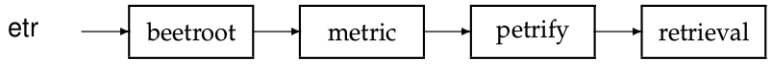
2. Consider the following fragment of a positional index with the same format:

   Gates: 1: ⟨3⟩; 2: ⟨6⟩; 3:⟨2,17⟩; 4: ⟨1⟩
   IBM: 4: ⟨3⟩; 7: ⟨14⟩;
   Microsoft: 1: ⟨1⟩; 2: ⟨1,21⟩; 3: ⟨3⟩; 5: ⟨16,22,51⟩;

   (a) Describe the set of documents that satisfy the query Gates /2 Microsoft
   (b) Describe each set of values for k for which the query Gates /k Microsoft returns a different set of documents as the answer.

Tolerant retrieval

1. Write down the entries in the permuterm index dictionary that are generated by the term *Retrieval*.
2. If you want to search for s*ng in a permuterm wildcard index, what key(s) would one do the lookup on?
3. Consider the following example of a postings list in a 3-gram index.



   Why is it useful to have the vocabulary terms in the postings lexicographically ordered?

4. We want to compute the Levenshtein edit distance between *Frodo* and *Gondor*. Consider the sub-problem of computing the distance between *G* and *Frod*. What are the costs for insertion, deletion and replacement respectively.
5. Write down the full 6×5 array of distances between all prefixes as shown in the lecture 3. What is the minimum edit distance between *Frodo* and *Gondor*.
6. What is the Levenshtein-Damerau distance between *hill* and *goblin*? Write down the solution in the same tabular format from the previous task.
7. What is the Jaccard coefficient between the word *bord* and each of *lord*, *morbid*, and *sordid* when we treat them as bigrams?