Phrase Queries

Phrase queries

- We want to be able to answer queries such as ***"stanford university"*** – as a phrase
- Thus the sentence *"The inventor Stanford never went to university"* is not a match.
  - Many more queries are *implicit phrase queries*
- For this, it no longer suffices to store only

  <*term* : *docs*> entries

Solution 1: Biword indexes

- Index every consecutive pair of terms in the text as a phrase
- For example the text “Friends, Romans, Countrymen” would generate the biwords
  - ***friends romans***
  - ***romans countrymen***
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate.

Longer phrase queries

- Longer phrases can be processed by breaking them down
- ***stanford university palo alto*** can be broken into the Boolean query on biwords:

stanford university *AND* ***university palo*** *AND* ***palo alto***

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.

Can have false positives!

Issues for biword indexes

- False positives, as noted before
- Index blowup due to bigger dictionary
  - Infeasible for more than biwords, big even for them

- Biword indexes are not the standard solution (for all biwords) but can be part of a compound strategy

Solution 2: Positional indexes

- In the postings, store, for each ***term*** the position(s) in which tokens of it appear:

<***term****,* number of docs containing ***term***;

doc1: position1, position2 … ;

doc2: position1, position2 … ;

etc.>

Positional index example

<***be***: 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …>

Which of docs 1,2,4,5 could contain "***to be or not to be***"?

- For phrase queries, we use a merge algorithm recursively at the document level
- But we now need to deal with more than just equality

Processing a phrase query

Given the postings list for the word "be":

1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367, …

Which of documents 1, 2, 4, and 5 could contain “to be or not to be”?

A. Documents 1, 4, and 5.
B. Documents 4 and 5.
C. Any of them. We can’t tell.
D. Just document 4.

Processing a phrase query

Answer: Option B : Documents 4 and 5.

We need "be" to occur at indices where exactly 3 words are between them (that are not "be" itself) for the document to possibly contain the phrase “to be or not to be” - documents 4 and 5 are the only documents where this occurs (e.g. 430 and 434 in doc 4 and 363 and 367 in doc 5).

Processing a phrase query

- Extract inverted index entries for each distinct term: ***to, be, or, not.***
- Merge their *doc:position* lists to enumerate all positions with "***to be or not to be***".
  - ***to****:*
    - *2*:1,17,74,222,551; *4*:8,16,190,429,433; *7*:13,23,191; ...
  - ***be****:*
    - *1*:17,19; *4*:17,191,291,430,434; *5*:14,19,101; ...
- Same general method for proximity searches

Proximity queries

- LIMIT! /3 STATUTE /3 FEDERAL /2 TORT
  - Again, here, /*k* means “within *k* words of”.
- Clearly, positional indexes can be used for such queries; biword indexes cannot.

Positional index size

- A positional index expands postings storage *substantially*
  - Even though indices can be compressed
- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase and proximity queries … whether used explicitly or implicitly in a ranking retrieval system.

Positional index size

- Need an entry for each occurrence, not just once per document
- Index size depends on average document size

Rules of thumb

- A positional index is 2–4 as large as a non-positional index

- Compressed Positional index size 35–50% of volume of original text

  – Caveat: all of this holds for “English-like” languages

Combination schemes

- These two approaches can be profitably combined
  - For particular phrases (***“Michael Jordan”, “Barack Obama”***) it is inefficient to keep on merging positional postings lists
    - Even more so for phrases like ***“The Who”***
- Williams et al. (2004) evaluate a more sophisticated mixed indexing scheme
  - A typical web query mixture was executed in ¼ of the time of using just a positional index
  - It required 26% more space than having a positional index alone

More and more things are put into index

- Document structure
  - Title, abstract, body, bullets, anchor
- Entity annotation
  - Being part of a person’s name, location’s name