Information Retrieval

Prof: Ehab Ezzat Hassanein

Phrase Queries and Positional Index

Phrase Queries

- We want to b able to answer quires such as “Stanford University: - as a phrase
- Thus the sentence “I went to university at Stanford” is not a match
  - The concept of phrase queries has proven easily understood by users; one of the few “advanced search” ideas that works.
  - Many more queries are implicit phrase queries.
    - ***San Francisco***

<term : docs> entries ***not enough any more!!***

First attempt: Biword indexes

- Index every consecutive pair of terms in the text as a phrase
- For example th 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 immidiate.

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 queries do contain the phrase.

Issues for biword indexes

- False positives, as 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 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: pos1, pos2,.....;
    - doc2: pos1, pos2,.....;
      etc.>

Positional Index Example

- <be: 993353;
  - 1: 7, 18, 33, 86, 231;
  - 2: 3, 184;
  - 4: 17,121, 303, 486, 531;
  - 5: 363, 386, ….>
- For phrase queries, we use a merge algorithm recursively at the document level
- Now we need to deal with more than just equality

Processing a Phrase Query

- The phrase “ **to be or not to be**”

- Extract inverted index entries for each distinct term:

  to, be, or, not

  - **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 hear /k means “within k words of
- Clearly, positional indexes can be used for such queries; biword indexes cannot.

POSITIONAL INTERSECT

- An algorithm for proximity intersection of postings lists p1 and p2 .
- The algorithm finds places where the two terms appear within k words of each other and returns a list of triples giving docID and the term position in p 1 and p 2

```
POSITIONALINTERSECT(p1, p2, k)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then l ← ⟨ ⟩
 5             pp1 ← positions(p1)
 6             pp2 ← positions(p2)
 7             while pp1 ≠ NIL
 8             do while pp2 ≠ NIL
 9                do if |pos(pp1) − pos(pp2)| ≤ k
10                      then ADD(l, pos(pp2))
11                      else if pos(pp2) > pos(pp1)
12                              then break
13                   pp2 ← next(pp2)
14                while l ≠ ⟨ ⟩ and |l[0] − pos(pp1)| > k
15                do DELETE(l[0])
16                for each ps ∈ l
17                do ADD(answer, ⟨docID(p1), pos(pp1), ps⟩)
18                pp1 ← next(pp1)
19             p1 ← next(p1)
20             p2 ← next(p2)
21        else if docID(p1) < docID(p2)
22                then p1 ← next(p1)
23                else p2 ← next(p2)
24  return answer
```

Positional Index Size

- A positional index expands postings storage substantially
  - Even though the indices can be compressed

- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase proximity queries – whether used explicitly or implicitly in a ranking system

Positional Index Size

- Need an entry for each occurrence, not just once per document.
- Index size depends on average document size
  - Average web page has <1000 terms
  - SEC filing, books, even some epic poems … easily 100,000 items.
- Consider a term with frequency 0.1%

Document size	Postings	Positional postings
1000	1	1
100,000	1	100

Rules of thumb

- A positional index is 2-4 as large as a non-positional index
- A positional index size 35-50% of volume of original text> Positional index is about the size of 10% of the original text.
- These rules of thumb will hold for English like language. Different languages may have different results.

Combination Schemes

- These two approaches, the biword and the positional, can be profitably combined.
  - For particular phrases (“Michael Jackson”, “Britney Spears”) It is inefficient to keep on merging positional; lists
    - Even more so for phrases like “The Who”
- Williams et al (2004) evaluated 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