

# Preprocessing: Phrase Queries

## **COMP3009J: Information Retrieval**

Dr. David Lillis ([david.lillis@ucd.ie](mailto:david.lillis@ucd.ie))

UCD School of Computer Science  
Beijing Dublin International College

# Phrase Queries

- Want to be able to answer queries 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
- For this, it is not enough to store only  
<term : docs> entries

# A first attempt: 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 (bi- is a prefix used to mean that there are two of something).
  - ***friends romans***
  - ***romans countrymen***
- Each of these biwords is now a dictionary term.
- Two-word phrase query-processing is now immediate.

# Longer phrase queries

- “*stanford university palo alto*” can be broken into the Boolean query on biwords:

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

Without the documents themselves, we cannot verify that the docs matching the above Boolean query do contain the exact 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 it appears in the documents.

<**term**, number of docs containing **term**;  
doc1: position1, position2 ... ;  
doc2: position1, position2 ... ;  
etc.>

# Positional index example

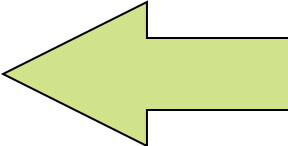
<*be*: 993427;

*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: we look for terms that follow each other.

# Positional index size

- You can compress position values.
- **BUT**, a positional index expands postings storage *substantially*
- Nevertheless, a positional index is now standardly used because of the power and usefulness of phrase queries.



# Rules of thumb

- A positional index is 2–4 times as large as a non-positional index.
- Positional index size 35–50% of volume of original text.
- Caveat: all of this holds for “English-like” languages.
- Often, a combination of selected biwords and a positional index are used, with good results.