# Phrase queries and positional indexes

IR system이 어떻게 phrase queries를 문서에 mapping시킬지가 point

## Phrase queries

- We 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 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
- For this, it no longer suffices to store only < term:docs > entries
  즉, < term:docs >구조로는 2개의 단어(phrase)를 포함하는 문서는 찾을 수 있지만 2개의 단어가 인접해있는 문서만을 찾을 수는 없음

## Solution 1 : Biword indexes

- index every consecutive pair of terms in the test as a phrase
- For example the test "Friends, Romans, Countrymen" would generate the biwords
    - friends romans
    - romans coutrymen

      → (**friends romans**) AND (**romans coutrymen**)
- Each of these biwords (like freinds romans, romans country) is now a dictionary term
- Two-word phrase query-processing is now immediate.

## Longer phrase queries

- Longer phrases can ve processed by breaking then down
- **stanford university palo alto** can be broken into the Boolean query on biwords:
  (**stanford university**) And (**university palo**) And (**palo alto**)
  즉, **A B C D**라는 phrase query가 들어올 때 word를 두개씩 묶어 (**A B**)  AND (**B C**) AND (**C D**)이 True인 document를 찾는다
- can have false positives
  즉, **A B C D**라는 phrase의 의미를 파악하고 관련 document를 찾는 것이 아니라 **A B**, **B C**, **C D** phrase이 포함되는 document를 찾는 것이기에 false positives가 생길 수 있다.

## Issues for biword indexes

- False positives
- **Index blowup** due to bigger dictionary
    - infeasible for more than biwords, big even for them
- Biword indexes are **not** the **standard solution** but can be part of a **compound strategy**

## Extended biwords

기존 biword indexes 방법은 단어 2개가 의미,관계가 없는데 묶어지게 될 수 있음
→ pos tag를 이용하여 의미, 관계가 있는 단어를 묶자 (False positives 해결)

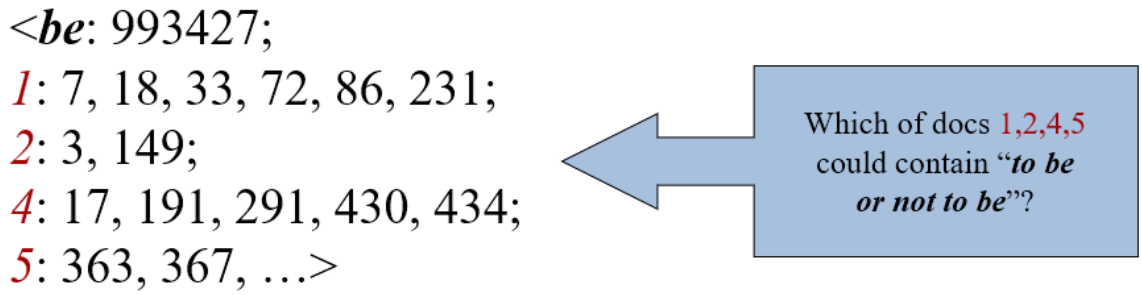
- parse the indexed text and perform part-of-speech-tagging(POST)
- bucket the terms into Nouns("명사")(N) and articles("관사")/prepositions("전치사") (X)
- now deem any string of terns of the form NX*N to be an extended biword
  - example : catcher(**N**) in(**X**) the(**X**) rye(**N**)
- query processing : parse it into N's and X's
  - segment query into enhanced biwords
  - look up index

## Solution 2 : Positional indexes

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

  <**term**, **the number of docs** containing term ;
  doc1 : position1, position2 … ;
  doc2 : position 1, position2 … ;
  etc.>

## Positional index example




- use a **merge algorithm** recursively at the document level
- but we now need to deal with more than just equality
  ex : query가 "informational retrieval"일 때  어떤 문서가 "informtion"의 postion이 24, 37일 때 "retrieval"는 25, 38이여야 "informational retrieval"을 포함하는 문서라고 인식해야 함

## 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, 17,74,222,251; 4 : 8,16,190,**429,433**; 7:13,23,191;...
  - **be** :

        - 1:17,19; 4:17,191,291,**430,434**; 5:14,19,101;...
- proximity search에도 사용될 수 있음

## Proximity queries

- Limit! /3 statute /3 federal /2 tort
    - /k means "within k words of"
- positional indexes can be used for such queries; biword indexes cannot.
- exercise : adapt the linear merge of postings to handle proximity queries. can you make it work for any value of k?
    - this is a little tricky to do correctoly and efficiently
    - there's likely to be a problem on it

## Positional index size

- 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**
- need an entry for each occurrence, not just one per document
- index size depends on **average document size**
    - example : consider a term with frequency 0.1%

| document size | postings | positional postings |
|---|---|---|
| 1000 | 1 | 1 |
| 100,000 | 1 | 100 |

document size가 100,000일 때 postings에 비해 positional postings가 100배가 되는 이유 : 한 term이 문서 내에 약 100(100,000*0.001)번 정도 나왔을 때 posings은 한번 나오던 100번 나오던 단어 하나만을 저장하지만 positional postings은 백번 나오면 그 백번 나왔던 위치를 모두 기억해야 하기에

출처 : stanford IR 강의(https://www.youtube.com/watch?v=QVVvx_Csd2I&list=PLaZQkZp6WhWwoDuD6pQCmgVyDbUWl_ZUi&index=6)