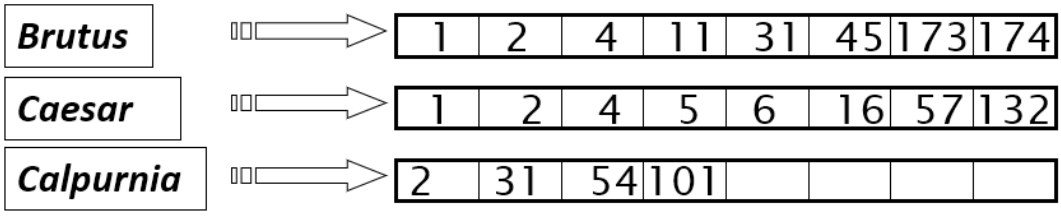
The Inverted Index

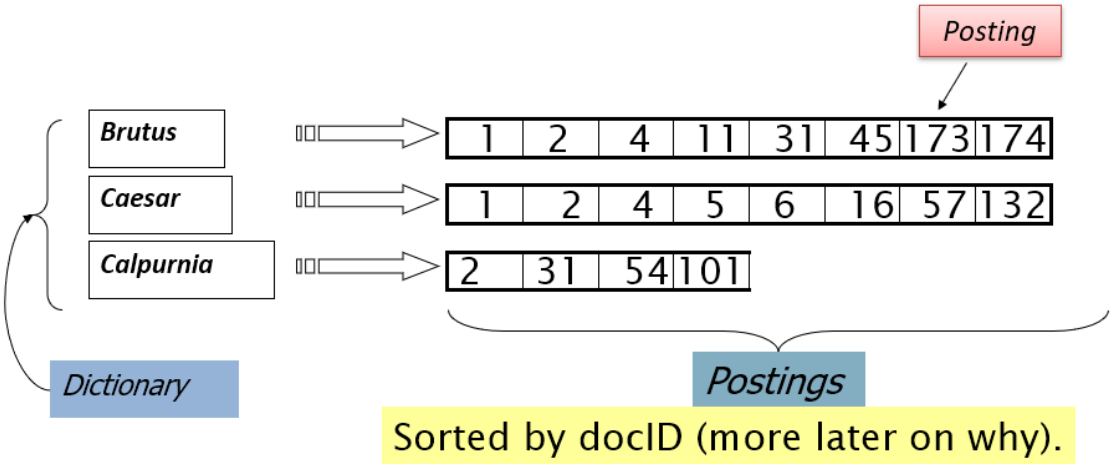
Inverted Index

- For each tern t, we must store a list of all documents that contain t.
  - identify each doc by a docID, a document serial number
- Can we used fixed-size arrays for this?

  → 비효율적. 많은 document에 포함되는 단어가 있는 반면, 몇개의 document에만 포함되는 단어가 있는데 같은 사이즈의 고정된 array를 할당하는 것은 굉장히 비효율적

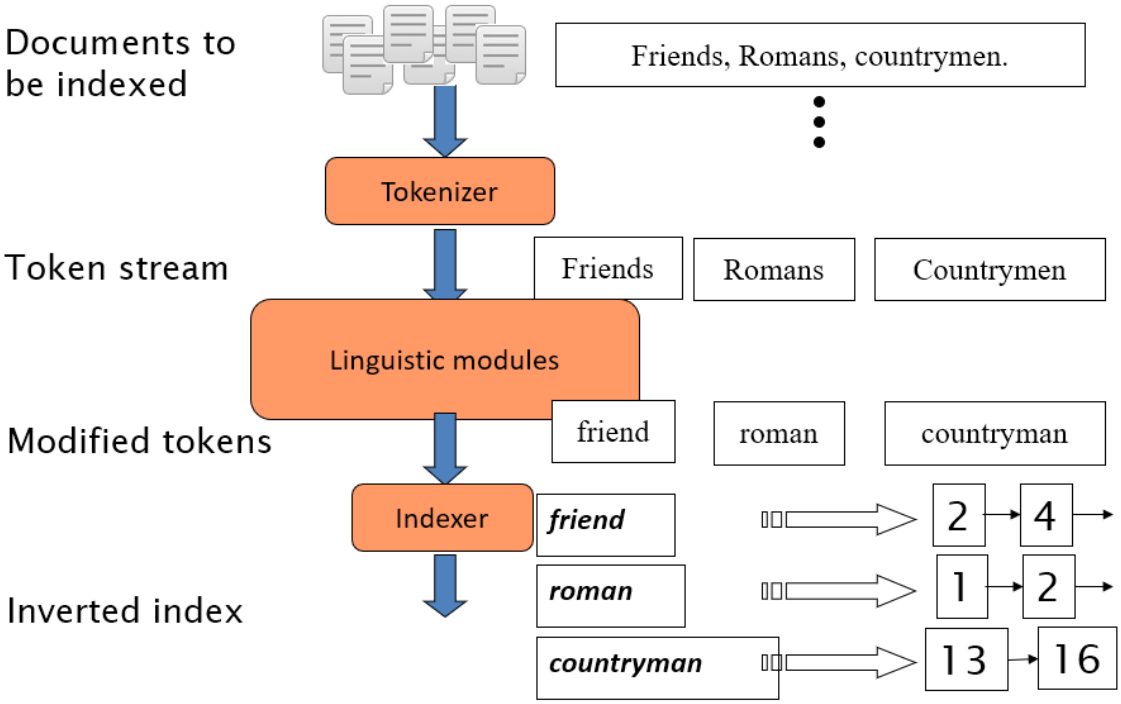


- We need variable-size **posting lists**
  - posting
    - a posting : a word-document pair
    - the postings : the sum of all the postings list
    - a posting은 docID로 정렬되어야 함
  - dictionary와 postings의 차이점
    - dictionary : relatively small, but it's normally essential that it's in memory. On disk, a continuous run of postings is normal and best
    - postings : large, but at least for something like a small scale enterprise search engine. normally be stored on disc.
    - dictionary가 document 보다 접근 빈도수가 많다 → 하드디스크보단 메인메모리에 저장
      - dictionary를 메인메모리에서 사용한다면 **해싱**이용
      - dictionary를 하드디스크에서 사용한다면 **B+트리**이용



Inverted Index Construction (전체적인 구조)

Inverted index construction



- each of those documents : a sequence of characters.
- token stream: document → a sequence of word tokens
- modified tokens : the tokens → some kind of canonical form (표준형, 표준꼴, 정규형)
  - 즉, matching이 잘 되도록 형태를 바꿈 (normalization, stemming등이 해당)
  - being lower cased
  - being stemmed to remove the plural ending...
- inverted index

Initial Stages of text processing

- tokenization
  - cut character sequence into word tokens
- normalization
  - map text and query term to same form
    - you want **U.S.A** and **USA** to match. 즉, U.S.A와 USA가 같은 단어로 matching 되기 위해 U.S.A를 USA로 변환하던가 USA를 U.S.A로 변환. → same form 사용
- stemming (가지치기 : 접두사, 접미사 등등)
  - we may wish different forms of a root to match
    - authorize, authorizatioin → authoriz. being mapped to the same stem.
- stop words
  - we may omit very common words
    - the, a, to, of, be
  - 하지만, 불용어를 사용하지 않았을 때 나타나는 단점 때문에 주로 사용함
    - ex) To Be Or Not To Be라는 노래 제목이 query인 경우 불용어를 제거하면 남아있는 단어가 없음

Indexer steps: Token sequence

- Sequence of (Modified token, Document ID) pairs.

Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Indexer steps: Sort

- Sort by terms
  - And then docID

Core indexing step

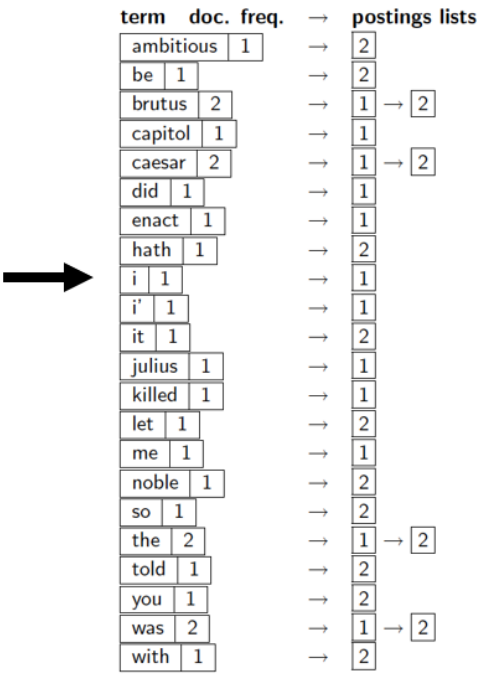
Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

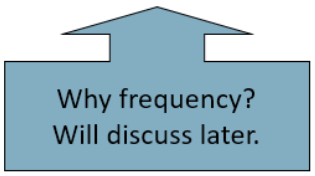
Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexer steps: Dictionary & Postings

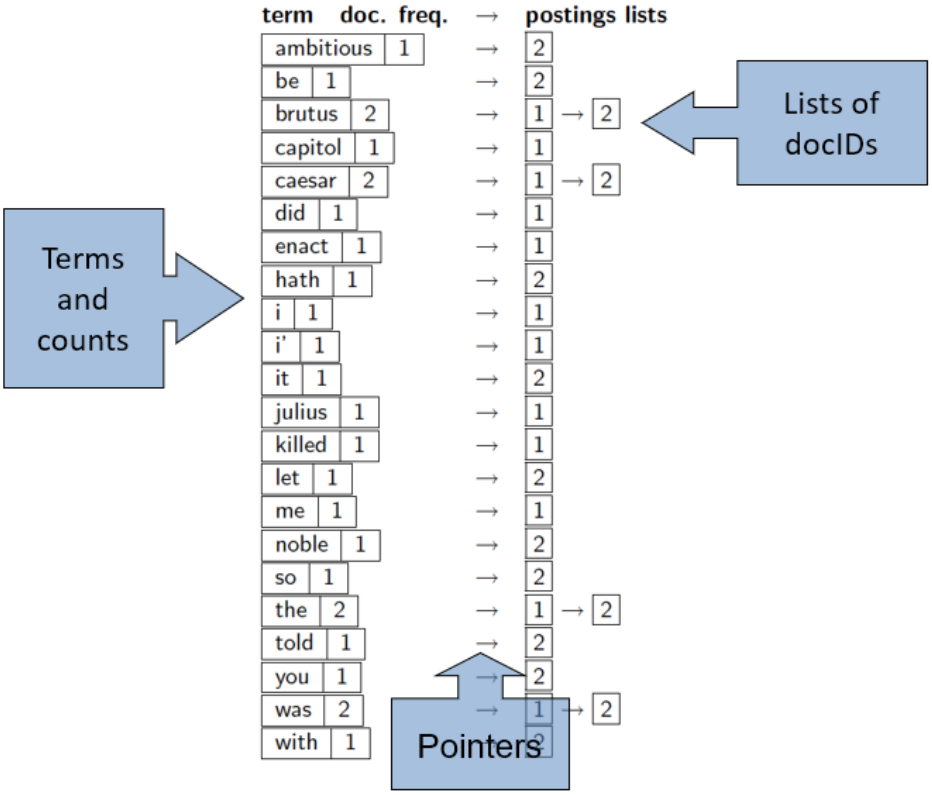
- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2





Where do we pay in storage?



출처 : 스탠포드 IR 강의 https://www.youtube.com/watch?v=Wf6HbY2PQDw&list=PLaZQkZp6WhWwoDuD6pQCmgVyDbUWl_ZUi&index=3