Information Retrieval & Social Web

CS 525/DS 595

Worcester Polytechnic Institute

Department of Computer Science

Instructor: Prof. Kyumin Lee

Project

- 3 or 4 person team
- Dates:
  - [7%] Project Proposal: March 18 by 11:59pm
  - [8%] Project website: April 24 by 11:59pm
  - [16%] Final project presentation: April 25 in-class

- https://canvas.wpi.edu/courses/7874/discussion_topics/31731

Previous Class…

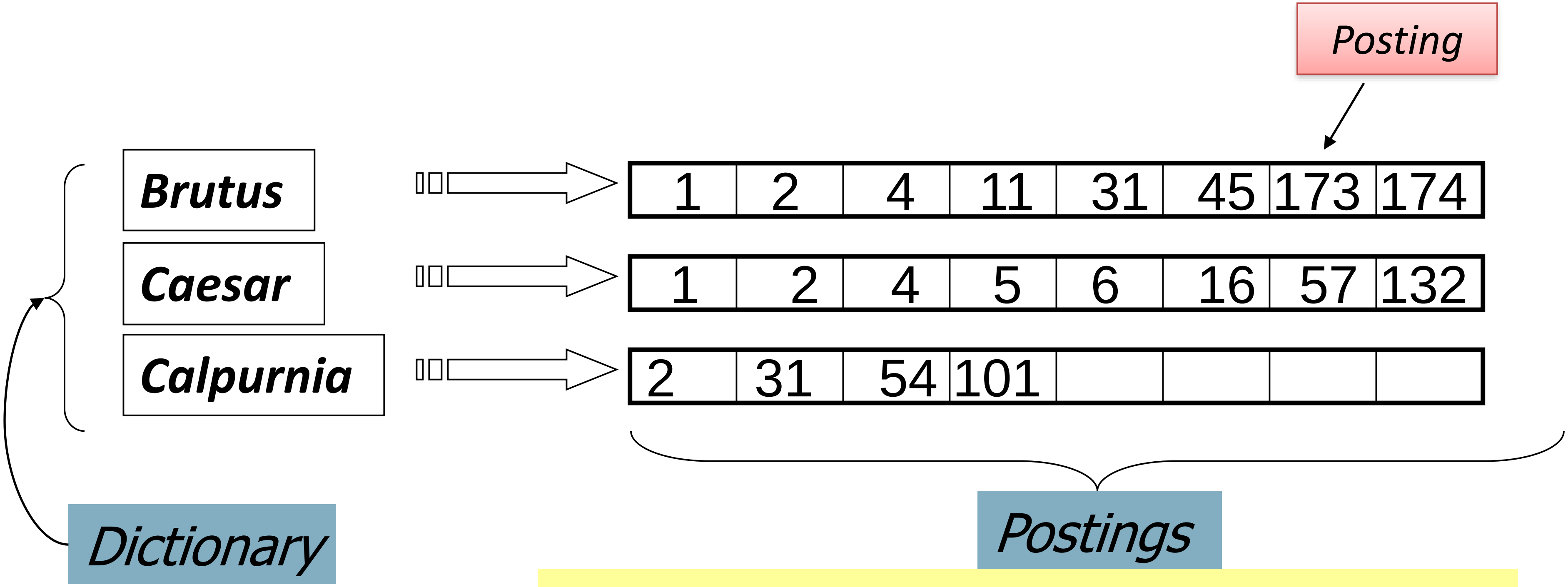
Boolean Retrieval Model

Previous Class…

Boolean Retrieval Model

Inverted index

Inverted index



Brutus	1	2	4	11	31	45	173	174
Caesar	1	2	4	5	6	16	57	132
Calpurnia	2	31	54	101				

Sorted by docID (more later on why).

Boolean queries: More general merges

- Exercise: Adapt the merge for the queries:

  Brutus *AND NOT* ***Caesar***

  Brutus *OR NOT* ***Caesar***

Can we still run through the merge in time O(x+y)?

What can we achieve?

Exercise Solution

- Brutus AND NOT Caesar
  - Time is O(x+y). Instead of collecting documents that occur in both postings lists, collect those that occur in the first one and not in the second

- Brutus OR NOT Caesar
  - Time is O(N) (where N is the total number of documents in the collection) assuming we need to return a complete list of all documents satisfying the query. This is because the length of the result list is only bounded by N, not by the length of the postings lists.

Merging

What about an arbitrary Boolean formula?

(Brutus *OR* ***Caesar)*** *AND NOT*

(Antony *OR* ***Cleopatra)***

- Can we always merge in “linear” time?
  - Linear in what?
- Can we do better?

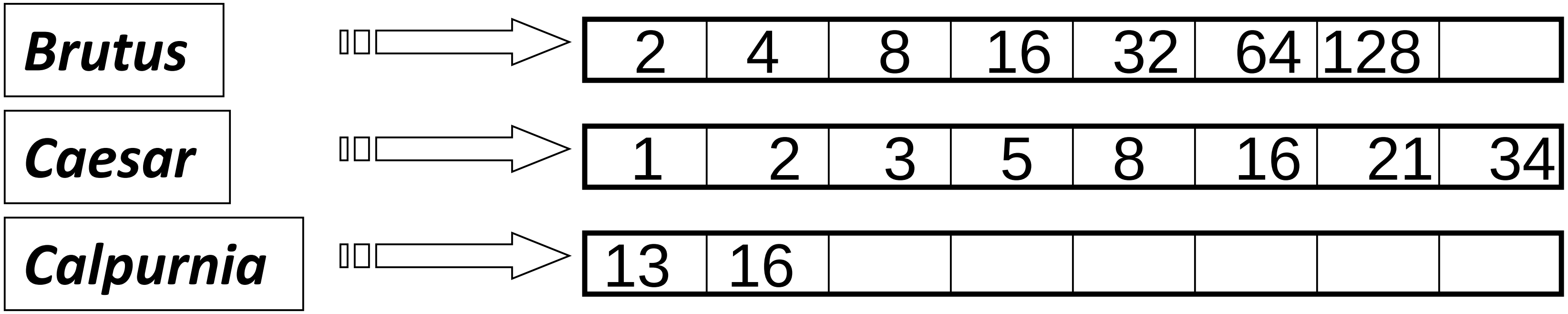
Solution

- We can always intersect in O(qN) where q is the number of query terms and N the number of documents, so the intersection time is linear in the number of documents and query terms. Since the tightest bound for the size of the result list is N, the number of documents, one cannot do better than O(N).

- But… still we can reduce computation time even though time complexity is still O(N). How?

Query optimization

- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.
- What is the best order for query processing?



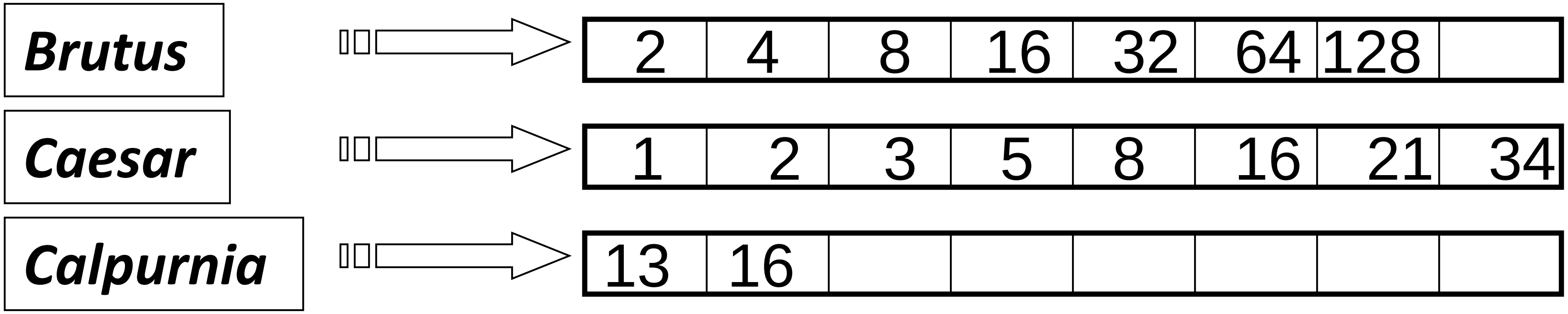
Brutus	2	4	8	16	32	64	128	
Caesar	1	2	3	5	8	16	21	34
Calpurnia	13	16						

Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

Query optimization example

- Process in order of increasing freq:
  - *start with smallest set, then keep cutting further*.

This is why we kept document freq. in dictionary



Brutus	2	4	8	16	32	64	128	
Caesar	1	2	3	5	8	16	21	34
Calpurnia	13	16						

Execute the query as (***Calpurnia*** *AND* ***Brutus)*** *AND* ***Caesar***.

More general optimization

- e.g., *(**madding** OR **crowd**) AND (**ignoble** OR **strife**)*
- Get doc. freq.'s for all terms.
- Estimate the size of each *OR* by the sum of its doc. freq.'s (conservative).
- Process in increasing order of *OR* sizes.

Exercise

- Recommend a query processing order for

***(tangerine** OR **trees)** AND*
***(marmalade** OR **skies)** AND*
(kaleidoscope** OR **eyes)

Term	Freq
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812

Exercise Solution

- Using the conservative estimate of the length of unioned postings lists, the recommended order is: (kaleidoskope OR eyes) (300,321) AND (tangerine OR trees) (363,465) AND (marmalade OR skies) (379,571)

What's ahead in IR? Beyond term search

- What about phrases?
  - ***Stanford University***
- Proximity: Find ***Gates*** *NEAR* ***Microsoft***.
  - Need index to capture position information in docs.
- Zones in documents: Find documents with (*author* = ***Ullman***) *AND* (text contains ***automata***).

Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
  - Usually more seems better
- Need term frequency information in docs

Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Often we want to rank/group results
  - Need to measure proximity from query to each doc.
  - Need to decide whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

Clustering, classification and ranking

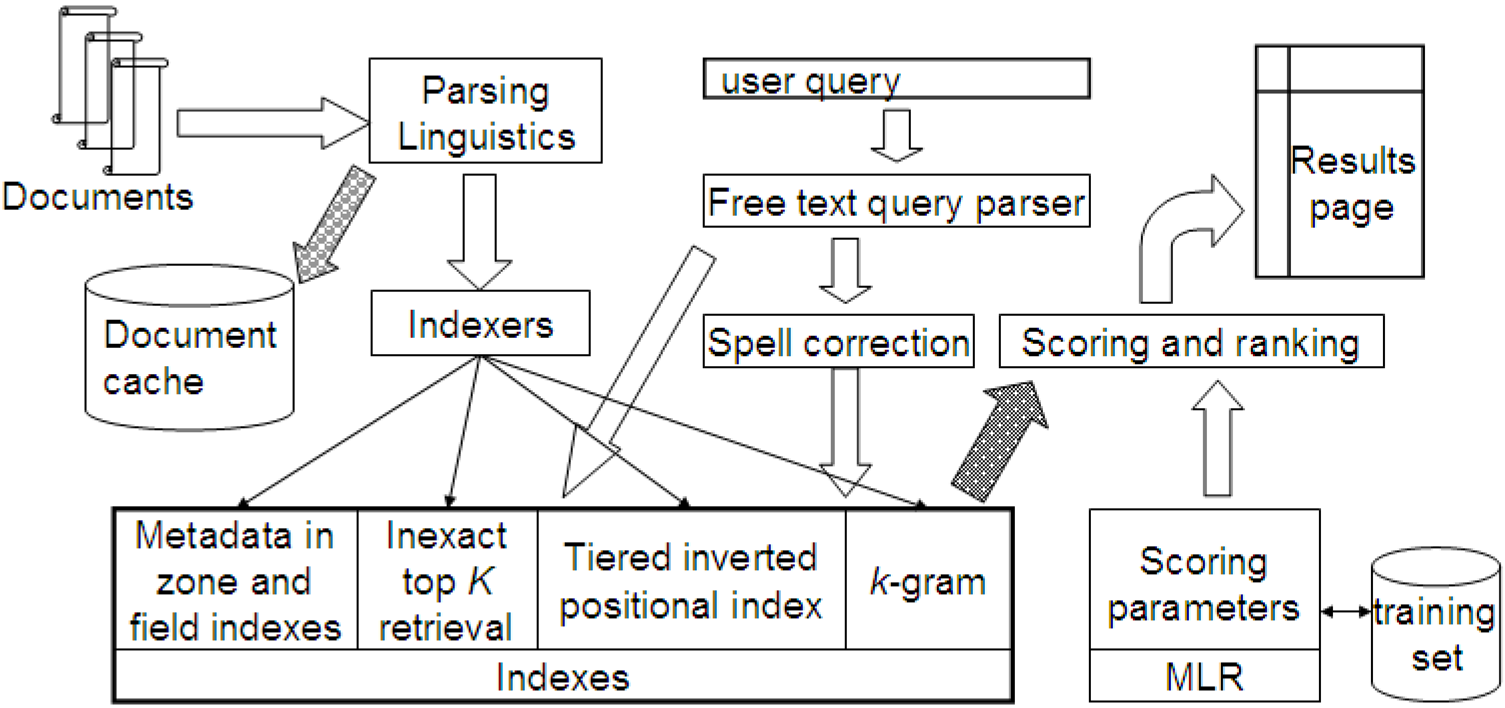
- Clustering: Given a set of docs, group them into clusters based on their contents.
- Classification: Given a set of topics, plus a new doc *D*, decide which topic(s) *D* belongs to.
- Ranking: Can we learn how to best order a set of documents, e.g., a set of search results

Exercise: Build Inverted Index and Run Boolean Retrieval

- Doc1: The winning ticket in Florida was sold at a Publix Supermarket.
- Doc2: The winning numbers were 08, 27, 34, 04 and 19, and the Powerball was 10.
- Doc3: With three winning tickets, the lump sum will be $187.2 million.

- Doc1 AND Doc2 AND Doc3
- Doc1 OR Doc2

What’s next …

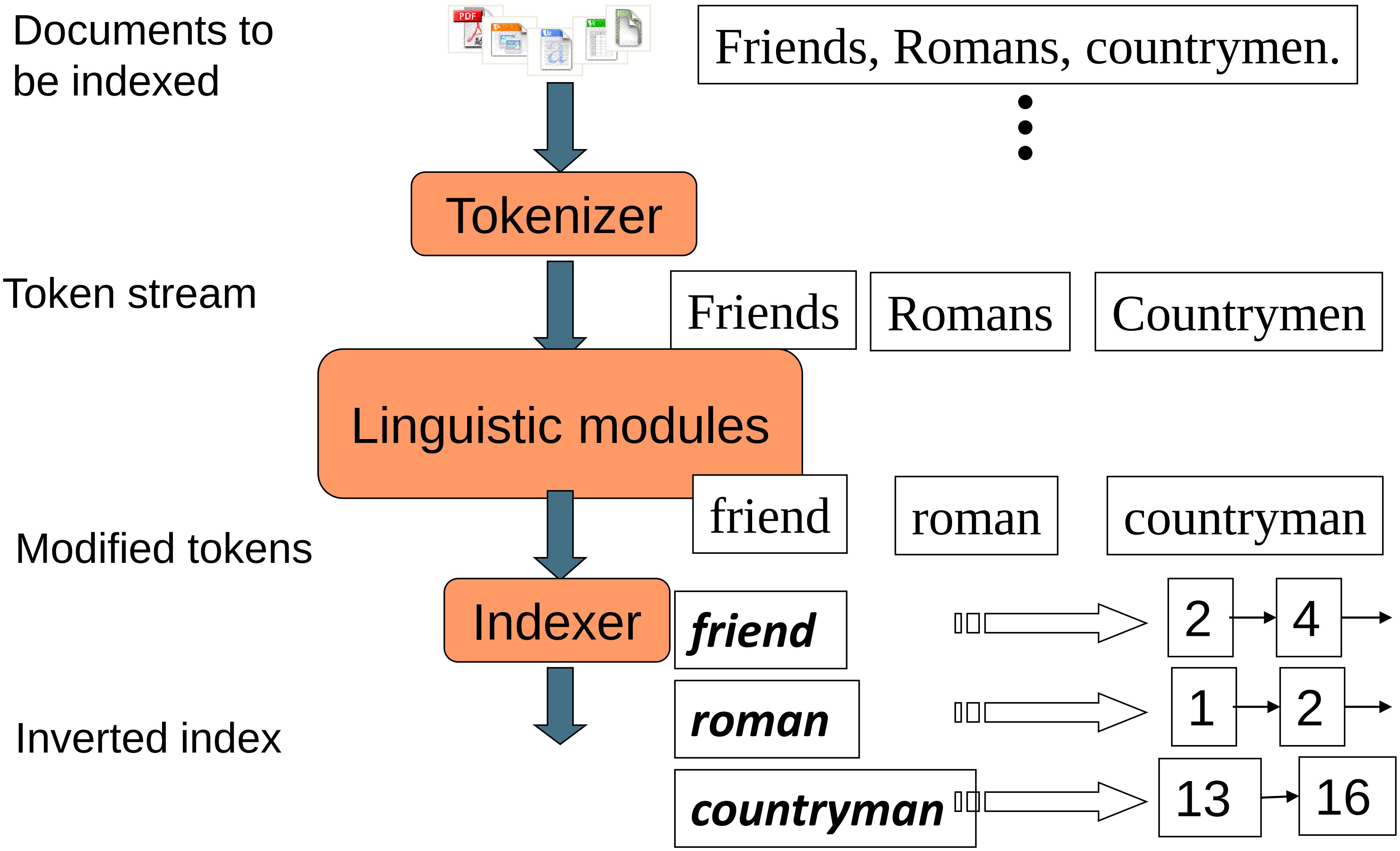

Documents
Parsing Linguistics
Document cache
Indexers
user query
Free text query parser
Spell correction
Scoring and ranking
Results page
Metadata in zone and field indexes
Inexact top *K* retrieval
Tiered inverted positional index
k-gram
Indexes
Scoring parameters
MLR
training set

Our assumptions so far

- We know what a document is
- We know what a term is
  - In reality, it can be complex

- So… We’ll look at how we define and process the vocabulary of terms in a collection

Recall the basic indexing pipeline



Documents to be indexed

Friends, Romans, countrymen.

Tokenizer

Token stream

Friends | Romans | Countrymen

Linguistic modules

Modified tokens

friend | roman | countryman

Indexer

Inverted index

friend → 2 → 4

roman → 1 → 2

countryman → 13 → 16

Initial stages of text processing

- Tokenization
  - Cut character sequence into word tokens
    - Deal with ***"John's"***, ***a state-of-the-art solution***
- Normalization
  - Map text and query term to same form
    - You want ***U.S.A.*** and ***USA*** to match
- Stemming
  - We may wish different forms of a root to match
    - ***authorize***, ***authorization***
- Stop words
  - We may omit very common words (or not)
    - ***the, a, to, of***

Parsing a document

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?
  - (CP1252, UTF-8, …)

Each of these is a classification problem, which we will study later in the course.

But these tasks are often done heuristically …

Complications: Format/language

- Documents being indexed can include docs from many different languages
  - A single index may contain terms from many languages.
- Sometimes a document or its components can contain multiple languages/formats
  - French email with a German pdf attachment.
- What is a unit document?
  - A file?
  - An email? (Perhaps one of many in a single mbox file)
  - An email with 5 attachments?
  - A group of files (PPT or LaTeX in HTML)

Tokenization

- Input: “***Friends, Romans and Countrymen***”
- Output: Tokens
  - ***Friends***
  - ***Romans***
  - ***Countrymen***
- A token is an instance of a sequence of characters
- Each such token is now a candidate for an index entry, after further processing
  - Described below
- But what are valid tokens to emit?

Why tokenization is difficult -- even in English

- Example: *Mr. O’Neill thinks that the boys’ stories about Chile’s capital aren’t amusing.*
- **Tokenize this sentence**

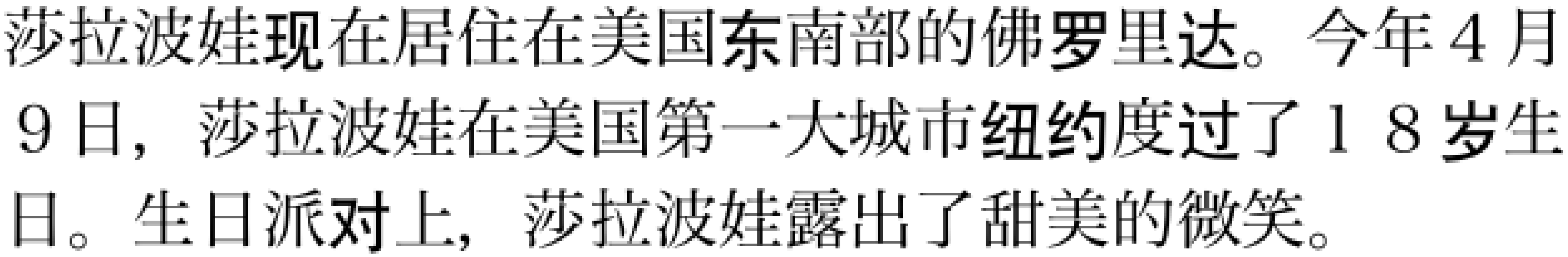
One word or two? (or several)

- Hewlett-Packard
- State-of-the-art
- co-education
- the hold-him-back-and-drag-him-away maneuver
- data base
- San Francisco
- Los Angeles-based company
- cheap San Francisco-Los Angeles fares
- York University vs. New York University

Numbers

- 3/12/91
- 12/3/91
- Mar 12, 1991
- B-52
- 100.2.86.144
- (800) 234-2333
- 800.234.2333

Chinese: No whitespace



莎拉波娃现在居住在美国东南部的佛罗里达。今年4月9日，莎拉波娃在美国第一大城市纽约度过了18岁生日。生日派对上，莎拉波娃露出了甜美的微笑。

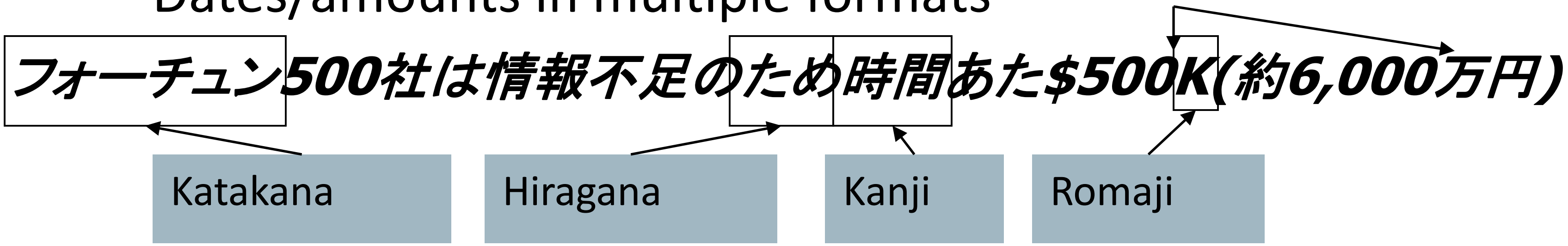
Ambiguous segmentation in Chinese

和尚

- Can be treated as one word meaning “monk” or as two words meaning “and” and “still”

Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

End-user can express query entirely in hiragana!

Language issues in French

- ***L'ensemble*** → one token or two?
  - ***L*** ? ***L’*** ? ***Le*** ?
  - Want ***l’ensemble*** to match with ***un ensemble***

Bidirectionality in Arabic

- Arabic (or Hebrew) is basically written right to left, but with certain items like numbers written left to right
- Words are separated, but letter forms within a word form complex ligatures

استقلت الجزائر في سنة 1962 بعد 132 عاما من الاحتلال الفرنسي.

← → ← → ← start

- ‘Algeria achieved its independence in 1962 after 132 years of French occupation.’

Normalization

- Need to “normalize” words in indexed text as well as query words into the same form
  - We want to match ***U.S.A.*** and ***USA***
- We most commonly implicitly define **equivalence classes** of terms
  - e.g., deleting periods to form a term
- Alternative is to do asymmetric expansion:
  - Enter: *window*   Search: *window, windows*
  - Enter: *windows*   Search: *Windows, windows*
  - Enter: *Windows*   Search: *Windows*
- Potentially more powerful, but less efficient

Normalization: other languages

- Accents: e.g., French ***résumé*** vs. ***resume*.**
- Most important criterion:
  - How are your users like to write their queries for these words?

- Even in languages that standardly have accents, users often may not type them
- German: *Tuebingen* vs. *Tübingen*
  - Should be equivalent

Normalization: other languages

- Need to "normalize" indexed text as well as query terms into the same form
  - **7月30日 vs. 7/30**
- Character-level alphabet detection and conversion
  - Tokenization not separable from this.
  - Sometimes ambiguous:

Morgen will ich in MIT …

Is this German "mit"?

Case folding

- Reduce all letters to lower case
  - exception: upper case in mid-sentence?
    - e.g., General Motors
    - Fed vs. fed
    - SAIL vs. sail
  - Often best to lower case everything, since users will use lowercase regardless of ‘correct’ capitalization…

Stop words

- With a stop list, you exclude from the dictionary entirely the commonest words. Intuition:
  - They have little semantic content: *the, a, and, to, be*
  - There are a lot of them: ~30% of postings for top 30 words
- But the trend is away from doing this:
  - Good compression techniques means the space for including stop words in a system is very small
  - Good query optimization techniques mean you pay little at query time for including stop words.
  - You need them for:
    - Phrase queries: “King of Denmark”
    - Various song titles, etc.: “Let it be”, “To be or not to be”
    - “Relational” queries: “flights to London”

Lemmatization

- Reduce inflectional/variant forms to base form
- Example: *am, are, is* → *be*
- Example: *car, cars, car's, cars'* → *car*
- Example: the boy's cars are different colors → the boy car be different color
- Lemmatization implies doing “proper” reduction to dictionary headword form (the lemma).

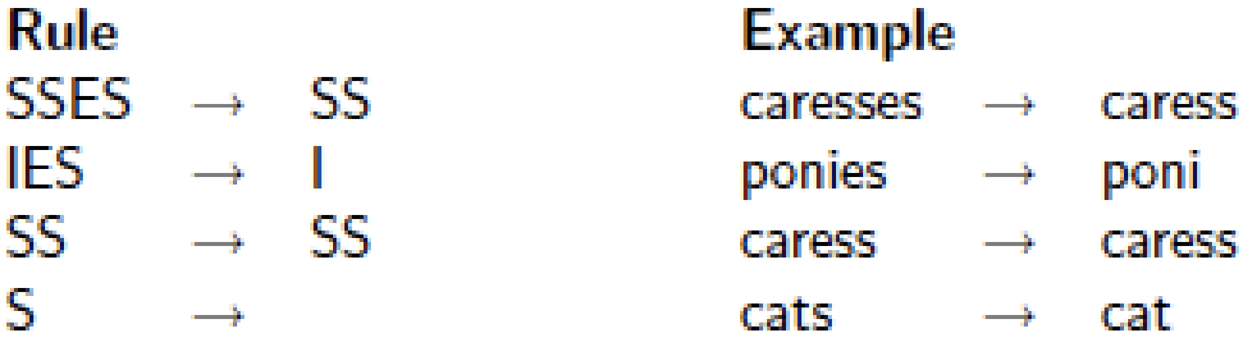
Stemming

- Reduce terms to their “roots” before indexing
- “Stemming” suggests crude affix chopping
- language dependent
- Example: ***automate(s), automatic, automation*** all reduced to ***automat***.

Porter Stemming Algorithm

- Most common algorithm for stemming English
- Results suggest that it is at least as good as other stemming options
- Contains 5 phases of reductions
- Phases are applied sequentially
- Each phase consists of a set of commands.
  - Sample command: Delete final ement if what remains is longer than 1 character
  - replacement → replac
  - cement → cement

Porter stemmer:
A few rules



Rule			Example		
SSES	→	SS	caresses	→	caress
IES	→	I	ponies	→	poni
SS	→	SS	caress	→	caress
S	→		cats	→	cat

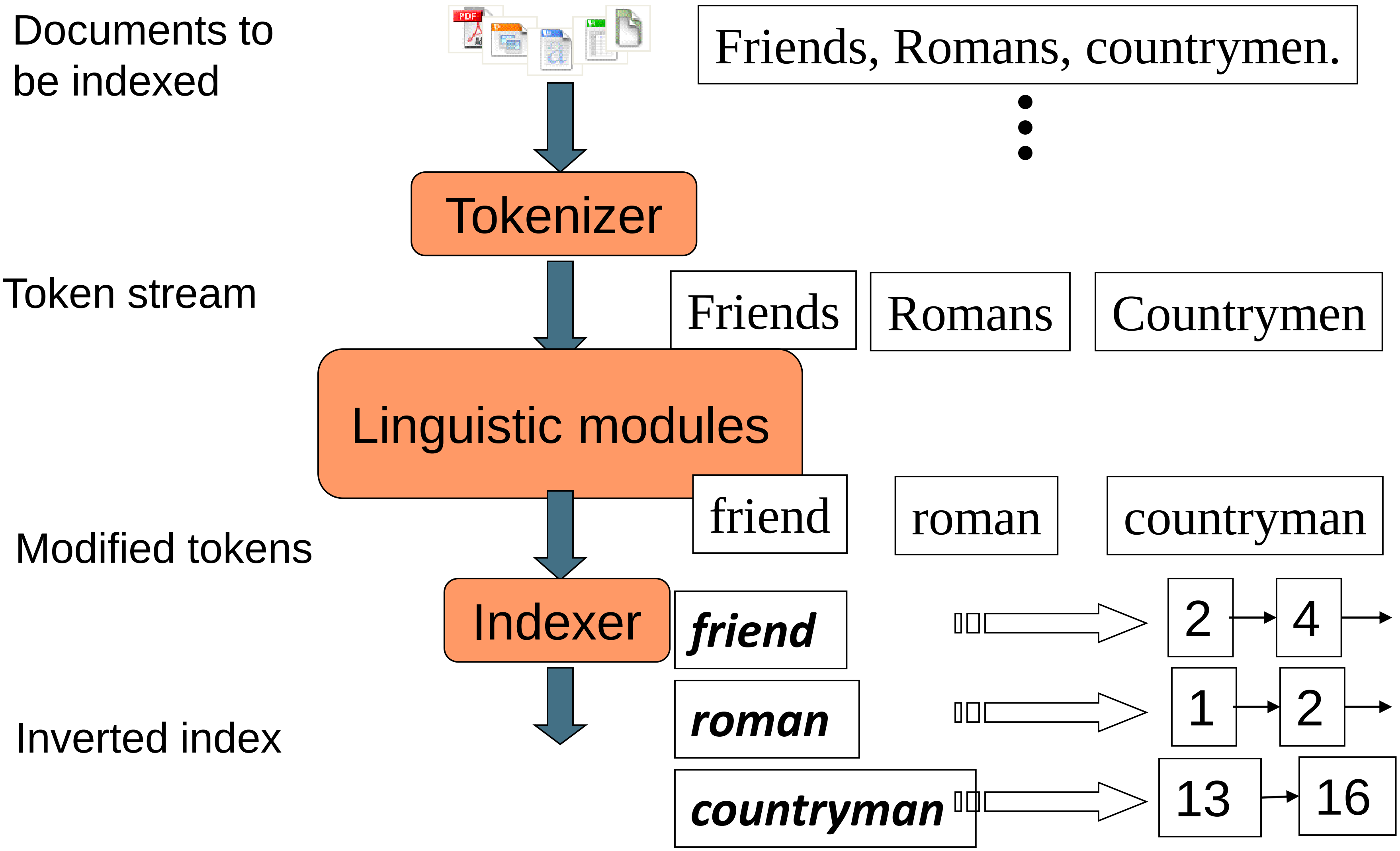
Three stemmers: A comparison

- **Sample text**: Such an analysis can reveal features that are not easily visible from the variations in the individual genes and can lead to a picture of expression that is more biologically transparent and accessible to interpretation
- **Porter stemmer**: such an analysi can reveal featur that ar not easili visibl from the variat in the individu gene and can lead to a pictur of express that is more biolog transpar and access to interpret
- **Lovins stemmer**: such an analys can reve featur that ar not eas vis from th vari in th individu gen and can lead to a pictur of expres that is mor biolog transpar and acces to interpres
- **Paice stemmer**: such an analys can rev feat that are not easy vis from the vary in the individ gen and can lead to a pict of express that is mor biolog transp and access to interpret

Does stemming improve effectiveness?

- In general, stemming increases effectiveness for some queries, and decreases effectiveness for others.
- Porter Stemmer equivalence class **oper** contains all of **operate operating operates operation operative operatives operational.**
- Queries where stemming hurts: “operational AND research”, “operating AND system”, “operative AND dentistry”

Recall the basic indexing pipeline



Documents to be indexed

Friends, Romans, countrymen.

Tokenizer

Token stream

Friends Romans Countrymen

Linguistic modules

Modified tokens

friend roman countryman

Indexer

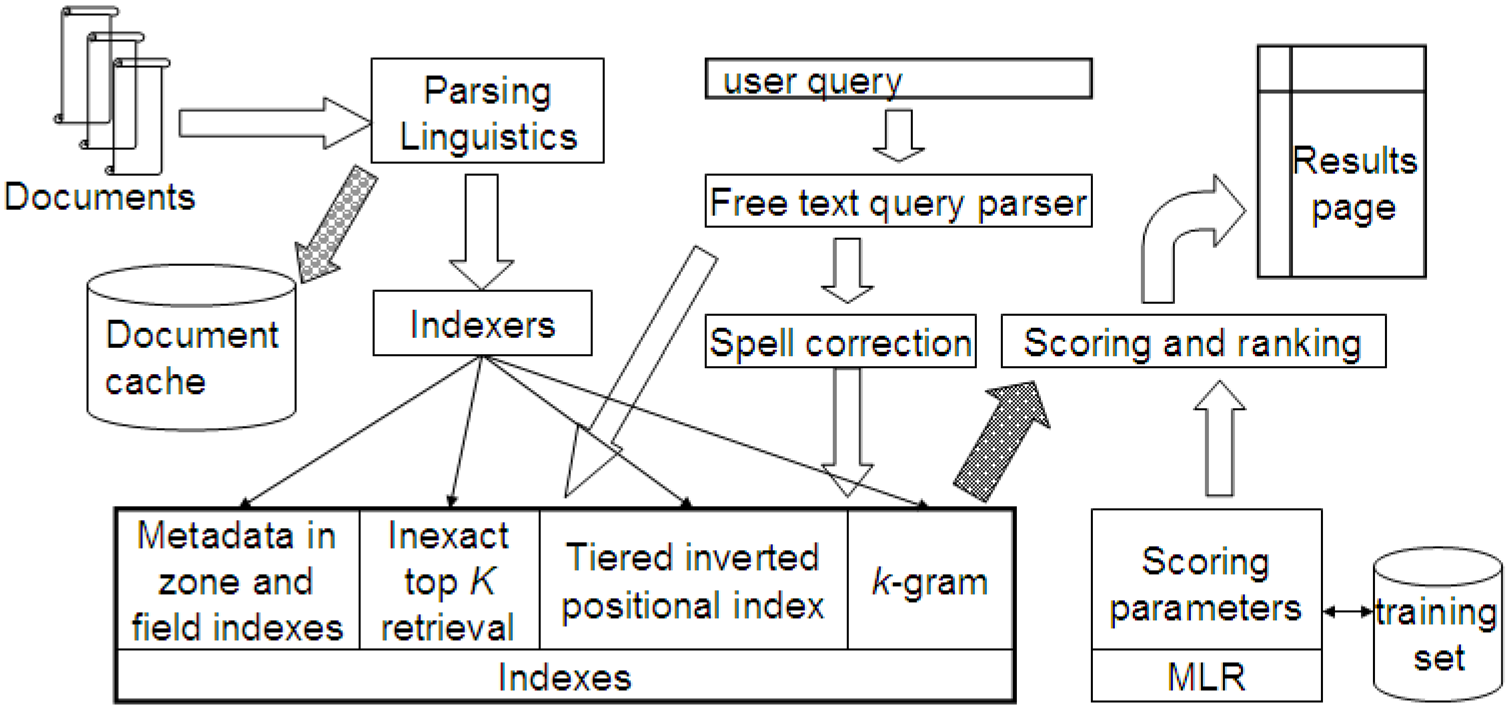
Inverted index

friend → 2 → 4 →

roman → 1 → 2 →

countryman → 13 → 16

Big Picture


Documents
Parsing Linguistics
user query
Free text query parser
Results page
Document cache
Indexers
Spell correction
Scoring and ranking
Metadata in zone and field indexes
Inexact top *K* retrieval
Tiered inverted positional index
k-gram
Indexes
Scoring parameters
MLR
training set

HW1

https://canvas.wpi.edu/courses/7874/assignments/47726

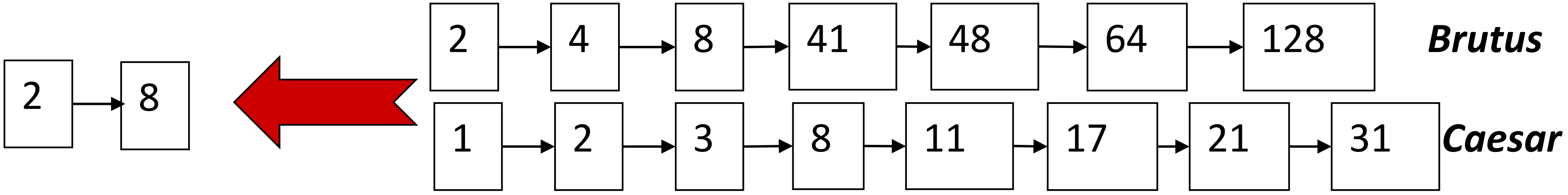
Next…

- Need a better index than simple <term: docs>
- How can we improve on our basic index?
  - **Skip pointers**: faster postings merges
  - **Positional index**: Phrase queries and Proximity queries
  - **Permuterm index**: Wildcard queries
  - **k-gram index**: Wildcard queries and spell correction

Faster postings merges:
Skip pointer

Recall basic merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries



2 → 8

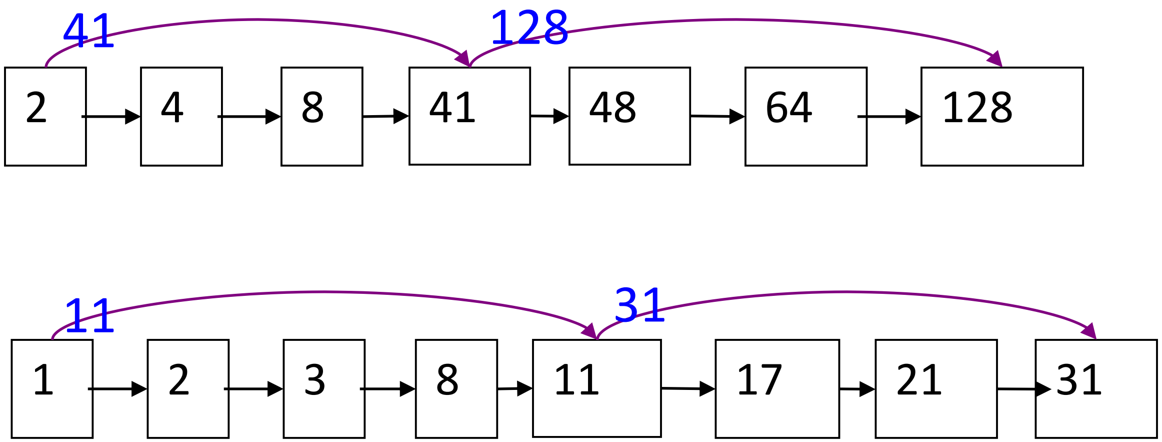
2 → 4 → 8 → 41 → 48 → 64 → 128 ***Brutus***

1 → 2 → 3 → 8 → 11 → 17 → 21 → 31 ***Caesar***

If the list lengths are *m* and *n*, the merge takes O(*m+n*) operations.

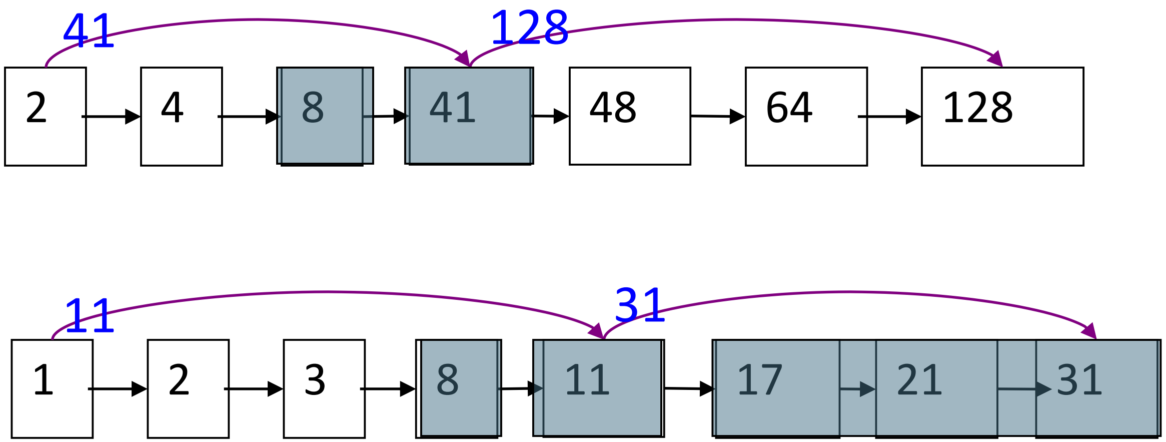
Can we do better?
Yes (if the index isn’t changing too fast).

Augment postings with skip pointers (at indexing time)



- Why?
- To skip postings that will not figure in the search results.
- How?
- Where do we place skip pointers?

Query processing with skip pointers



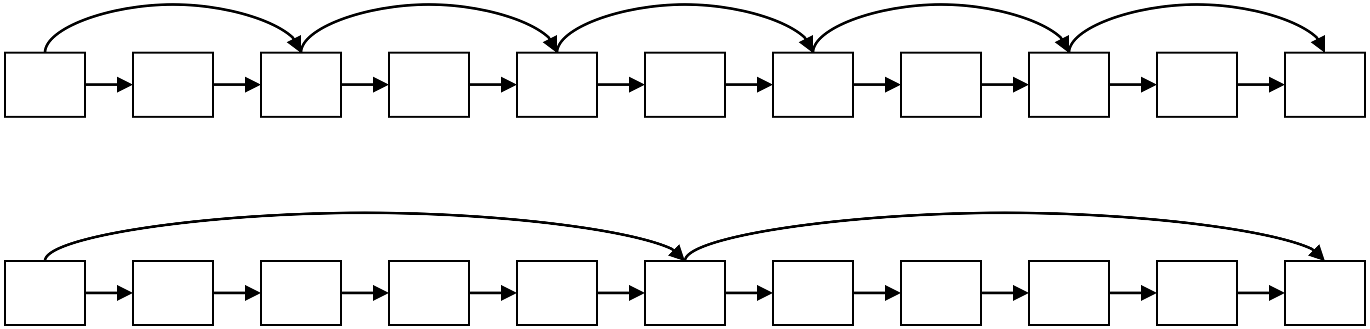
Suppose we’ve stepped through the lists until we process **8** on each list. We match it and advance.

We then have **41** and **11** on the lower. **11** is smaller.

But the skip successor of **11** on the lower list is **31**, so we can skip ahead past the intervening postings.

Where do we place skips?

- Tradeoff:
  - More skips → shorter skip spans ⇒ more likely to skip. But lots of comparisons to skip pointers.
  - Fewer skips → few pointer comparison, but then long skip spans ⇒ few successful skips.



Placing skips

- So… More skips or fewer skips… Where to add skip pointers???
- Simple heuristic: for postings of length *L*, use √*L* evenly-spaced skip pointers
- Easy if the index is relatively static; harder if *L* keeps changing because of updates.

Positional Index

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; about 10% of web queries are phrase queries
- How??

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
  - ***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***

Any problem?

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

Can have false positives!

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***"?

- Can compress position values/offsets
- Nevertheless, this expands postings storage *substantially*

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
  - Here, /*k* means “within *k* words of”.

- Clearly, positional indexes can be used for such queries; biword indexes cannot.

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 filings, books, even some epic poems … easily 100,000 terms
- Consider a term with frequency 0.1%

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

Positional index size

- You can compress position values/offsets
- Nevertheless, a positional index expands postings storage substantially
- Nevertheless, it 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.

Rules of thumb

- A positional index is 2–4 as large as a non-positional index
- Positional index size 35–50% of volume of original text

Positional Indexes: Wrap-up

- With a positional index, we can answer
  - phrase queries
  - proximity queries

Today…

- Need a better index than simple <term: docs>
- How can we improve on our basic index?
  - **Skip pointers**: faster postings merges
  - **Positional index**: Phrase queries and Proximity queries
  - **Permuterm index**: Wildcard queries