Big Data and Data Mining

Information Retrieval

Flavio Bertini

flavio.bertini@unipr.it

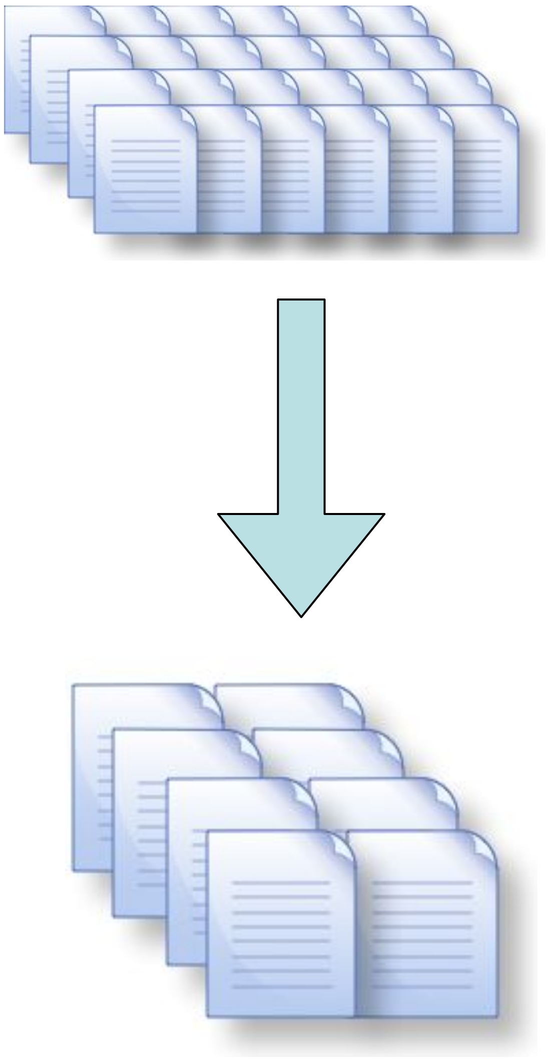
Information Retrieval

- Information Retrieval (IR) is **finding material** (usually documents) of an **unstructured nature** (usually text) that satisfies an **information need** from within **large collections** (usually stored on computers)

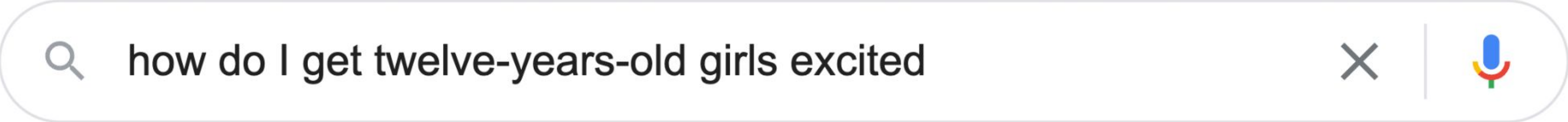
- These days we frequently think first of **web search**, but there are many other cases:
  - E-mail/social media search
  - Searching your laptop
  - Corporate knowledge bases
  - Domain specific search (*i.e.*, Legal information retrieval)
  - Digital Libraries
  - News

Basic Assumptions

- **Document**: unstructured file
- **Collection**: a set of documents
  - Assume it is a static, non-hypertext collection for the moment
- **Query**: set of keywords to express an information need
- **Goal**: retrieve documents with information that is relevant to the user’s information need and helps the user complete a task

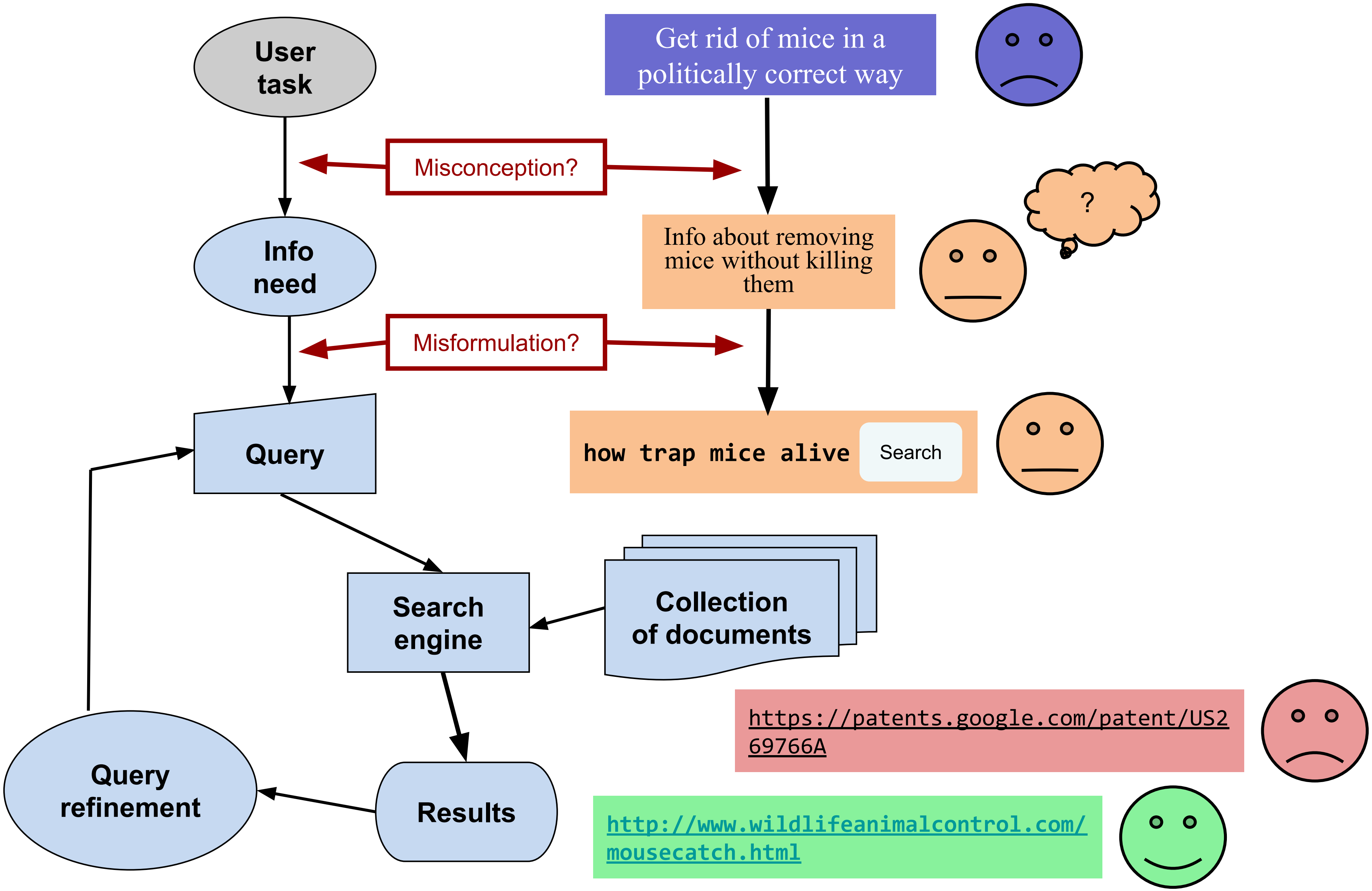


Misconception/Misformulation



how do I get twelve-years-old girls excited

Classic Search Model



User task

Misconception?

Info need

Misformulation?

Query

Search engine

Collection of documents

Results

Query refinement

Get rid of mice in a politically correct way

Info about removing mice without killing them

?

how trap mice alive

Search

https://patents.google.com/patent/US269766A

http://www.wildlifeanimalcontrol.com/mousecatch.html

What is a document?

- Examples
  - web pages, email, books, news stories, scholarly papers, text messages, Word, Powerpoint, PDF, forum postings, patents, instant messaging sessions, etc.
- Common properties
  - Significant text content
  - Mostly unstructured
  - Some structure (*e.g.*, title, author, date for papers, subject, sender, destination for email)

Data Retrieval vs IR (1/2)

- A Data Retrieval system (*i.e.*, DBMS) deals with data of a **well defined** structure (**database schema**)
- An Information Retrieval system deals with texts written in **natural language** (*i.e.*, English), often non structured and ambiguous (**text documents**)

Database record	Text document
Content is structured in typed fields	Content is unstructured
Values consistent with their defined data types	Can contain any kind of data (codes, dates, prices) all in text format
Data is stored in pre-defined formats	Information is expressed in natural language

Data Retrieval vs IR (2/2)

- A Data Retrieval system (*i.e.*, DBMS) retrieve data using a formally defined **query language** (*i.e.*, SQL, relational algebra, etc)
- An Information Retrieval system retrieve documents by means of a set of keywords (improperly called **query** or **more properly search**) expressed in natural language

Query language	Search language
Based on a formal grammar	Based on keywords from natural language
The answer is not dependent on the system implementation	Can be interpreted differently from different IR systems
Produces as an answer a predictable **set** of records which are **true** under the query	Produces a **ranked list** of results based on the **relevance** to the query

Comparing text

- **Comparing** the query text to the document text and determining what is a good match is the core issue of information retrieval
- Exact matching of words is not enough
  - Many different ways to write the same thing in a “natural language” like English
    - e.g., does a news story containing the text “bank director in Amherst steals funds” match the query?
  - Some stories will be better matches than others
- A **relevance** notion is needed instead of exact matching
- Different **retrieval models** are based on different relevance notions

Retrieval Models

- **Classic models**
    - Boolean retrieval
    - Vector Space Model
- **Probabilistic Models**
    - BM25 (Best Match 25)
    - Language models
- **Combining evidence**
    - Inference networks
    - Learning to Rank
- Progress in retrieval models has corresponded with improvements in effectiveness

Boolean Retrieval Model (1/2)

- The Boolean retrieval model relies on asking a query that is a Boolean expression:
  - Boolean Search are queries using AND, OR and NOT to combine search terms
    - Views each document as **a set of words (Bag-of-words)**
    - Is precise: document **matches condition or not**
  - Perhaps the simplest model to build an IR system
- Primary commercial retrieval tool for three decades until the early 1990s (arrival of World Wide Web)
- Many search systems you use still are Boolean:
  - Email, library catalog, Mac OS X Spotlight

Boolean Retrieval Model (2/2)

- Which plays of Shakespeare contain the words

  Brutus AND Caesar BUT NOT Calpurnia?
- One could "*grep*" all of Shakespeare's plays for Brutus and Caesar, then strip out lines containing Calpurnia?
- Why is that not the answer?
  - Slow (for large corpora)
  - NOT Calpurnia is non-trivial
- We need a way to represent documents as **sets of words**

William Shakespeare

Term-document incidence matrix

Documents

Words

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

1 if play contains word, 0 otherwise

Remember our search:

Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

Incidence vectors

Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

- So we have a 0/1 vector for each word
- To **answer** search: take the vectors for Brutus, Caesar and Calpurnia (the last complemented!) ➔ bitwise AND

110100 **AND** 110111 **AND** 101111 = **100100**

Words

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

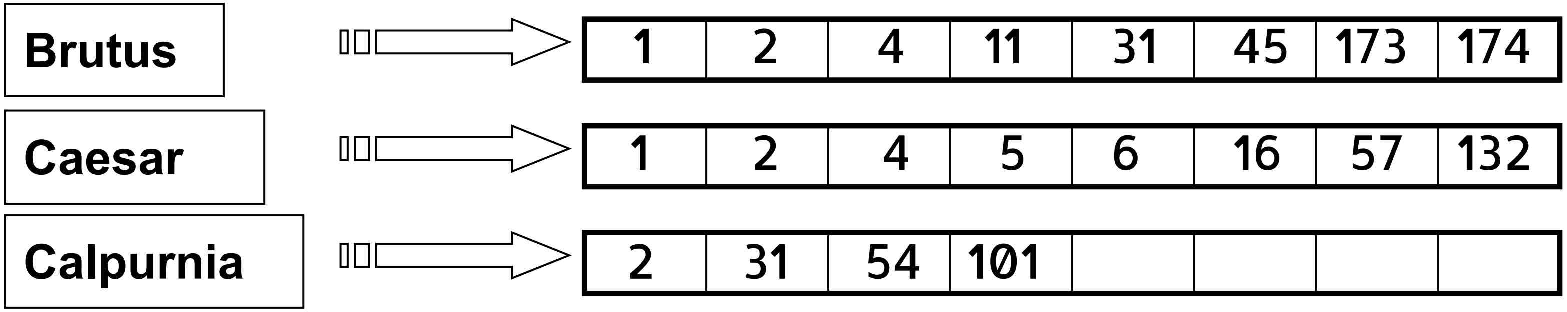
Answer:	**1** Anthony and Cleopatra	**0** Julius Caesar	**0** The Tempest	**1** Hamlet	**0** Othello	**0** Macbeth

Term-document Matrix dimension

- Consider *N* = 1 million documents, each with about 1000 words
- Avg 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents
- Say there are *M* = 500K *distinct* terms among these
- M x N = 500K x 1M matrix has half-a-trillion 0’s and 1’s
- Actually, it has no more than one billion 1’s
  - matrix is **extremely sparse**
- What’s a better representation?
  - **We only record the 1 positions**

Inverted Index

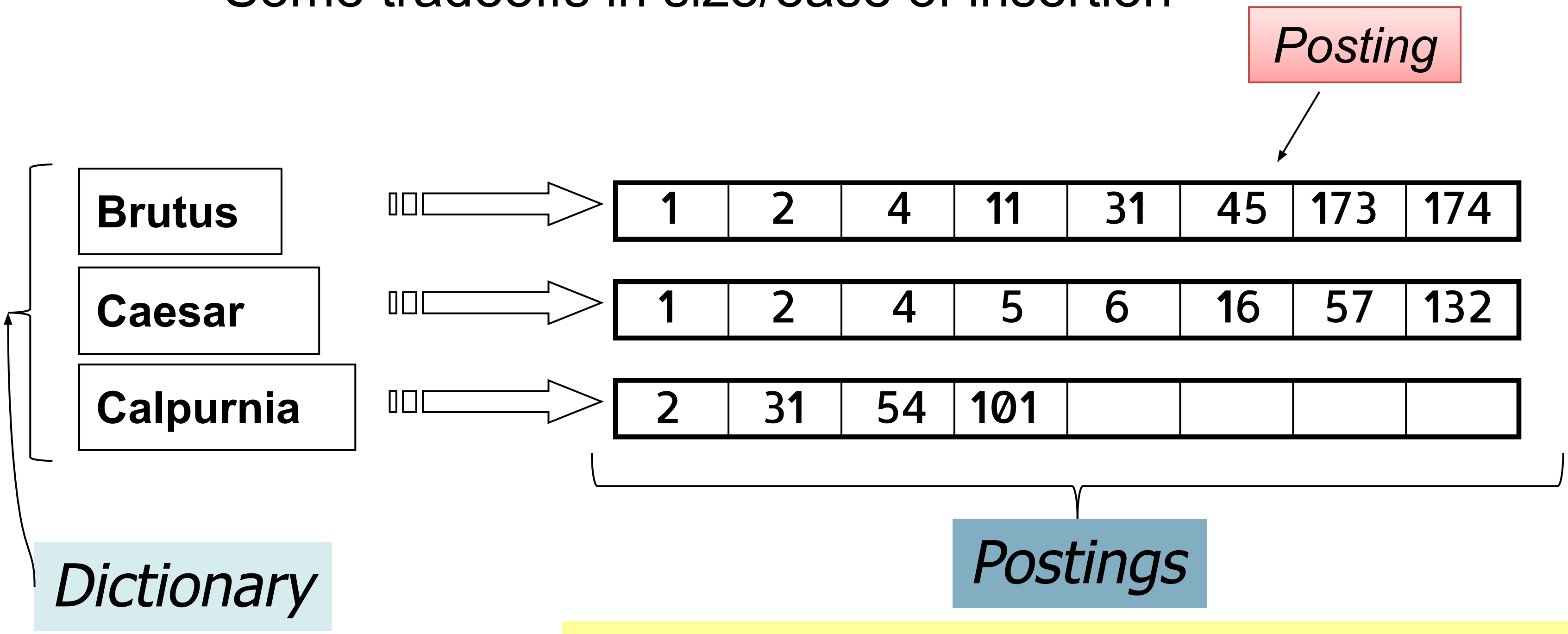
- Using the same example as before, we want to improve the time required to come to an answer and lessen the memory usage
- For each term *t*, we must store a list of all documents that contain *t*
  - Identify each doc by a **docID**, a document serial number
- Can we use fixed-size arrays for this?
  - What happens if the word Caesar is added to document 14?



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

Posting-lists

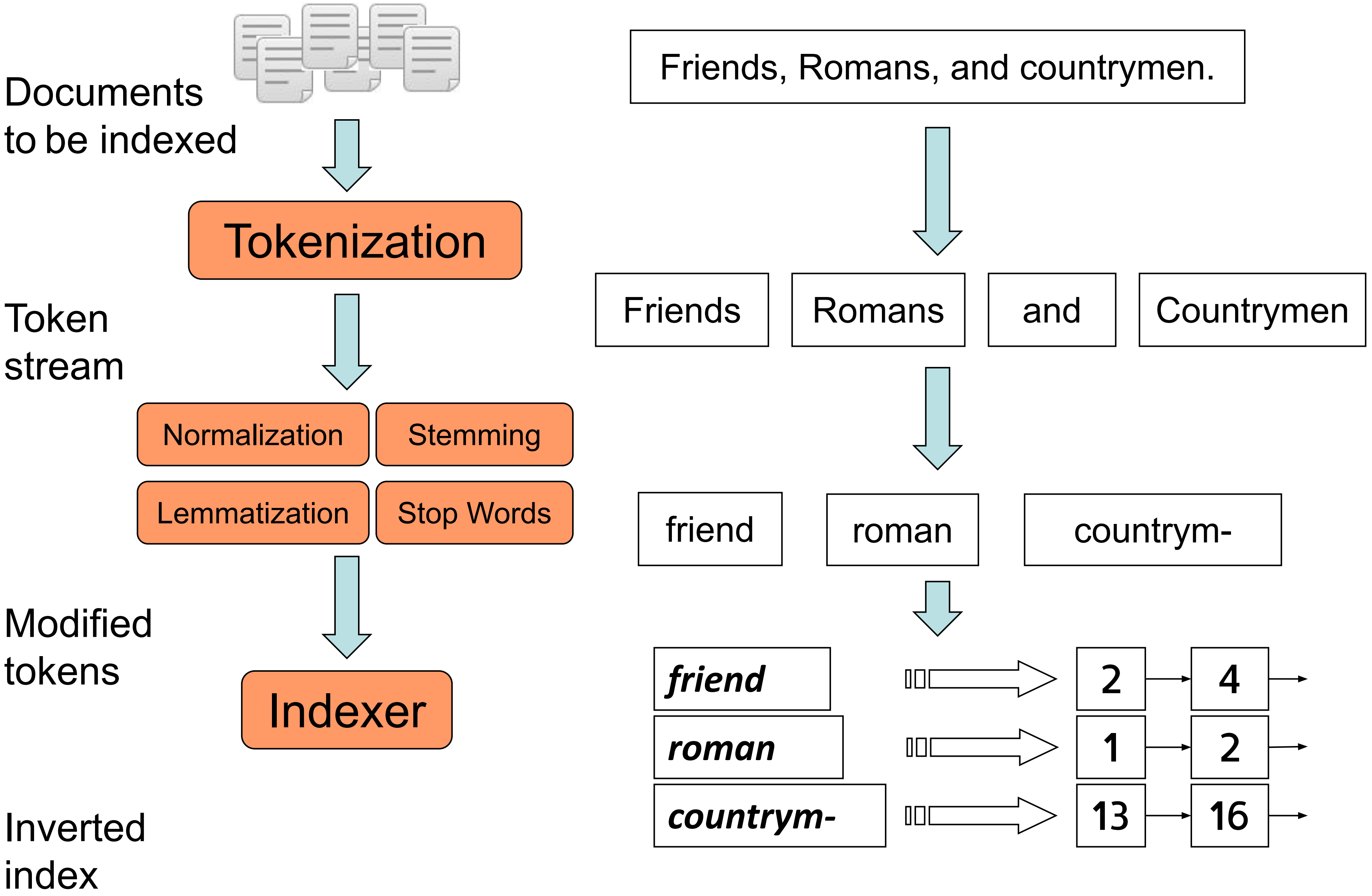
- We need variable-size postings lists
  - On disk, a continuous run of postings is normal and best
  - In main memory, can use linked lists or variable length arrays
    - Some tradeoffs in size/ease of insertion



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)

Inverted Index Construction



Documents to be indexed

Tokenization

Token stream

Normalization

Stemming

Lemmatization

Stop Words

Modified tokens

Indexer

Inverted index

Friends, Romans, and countrymen.

Friends

Romans

and

Countrymen

friend

roman

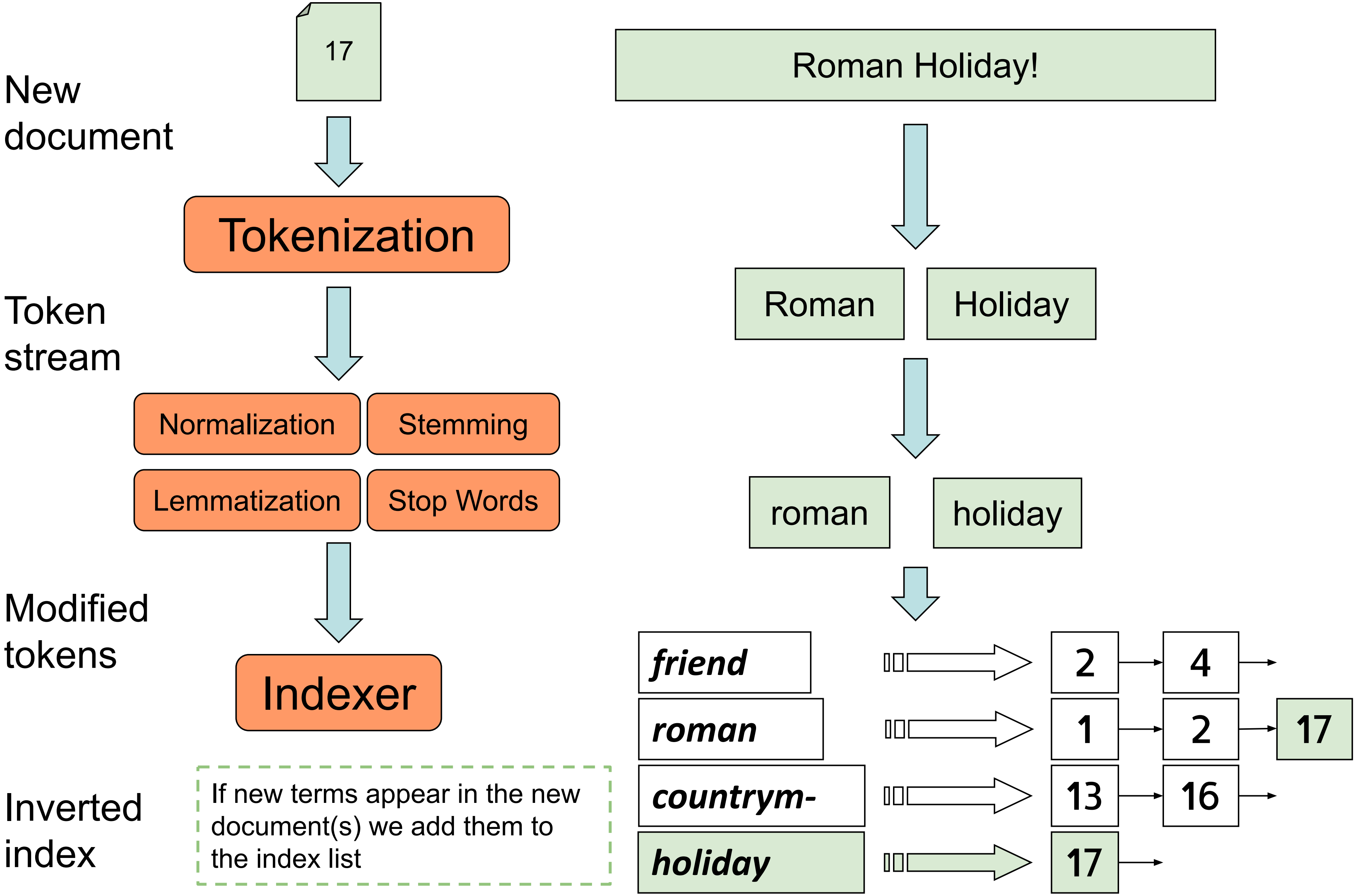
countrym-

friend → 2 → 4 →

roman → 1 → 2 →

countrym- → 13 → 16 →

Inverted Index Updating



If new terms appear in the new document(s) we add them to the index list

Text preprocessing

- 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 "normal" form
    - You want ***U.S.A.*** and ***USA*** to match
- Stemming & Lemmatization
  - 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***

Linguistic Modules

Tokenization

- Sequence of `<modified token, document ID>` pairs

docID 1

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

docID 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

Normalization

- We may 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***
- Result is terms: a term is a (normalized) word type, which is an entry in our IR system dictionary
- We most commonly implicitly define equivalence classes of terms by, for example:
  - deleting periods to form a term
    - ***U.S.A., USA*** → ***USA ??*** | ***naive, naïve*** → ***naive ??***
  - deleting hyphens to form a term
    - ***anti-discriminatory, antidiscriminatory*** → ***antidiscriminatory ??***

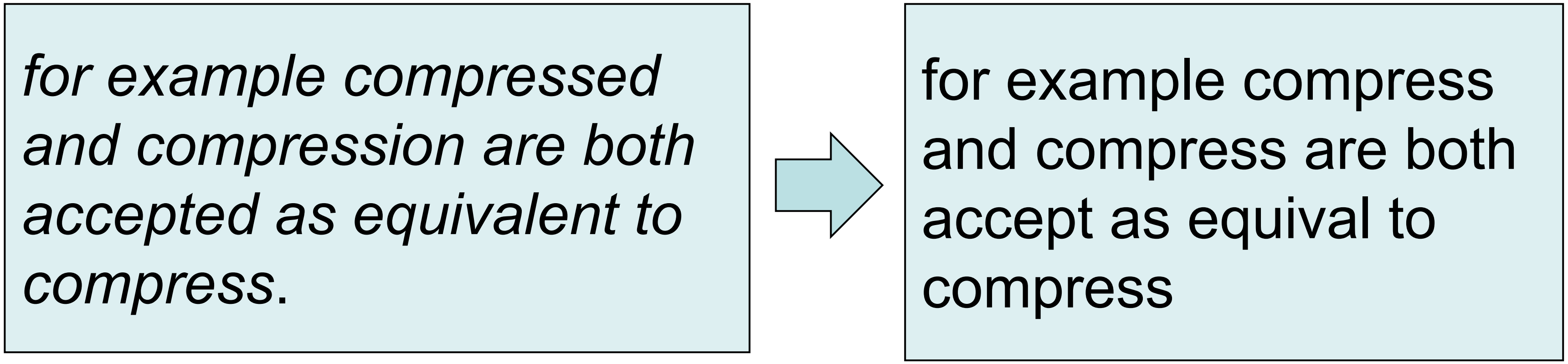
Normalization: Case folding

- 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…

- Longstanding Google example: [fixed in 2011…]
  - Query C.A.T.
  - #1 result is for "cats" (well, Lolcats) not Caterpillar Inc.

Stemming

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



for example compressed and compression are both accepted as equivalent to compress.

for example compress and compress are both accept as equival to compress

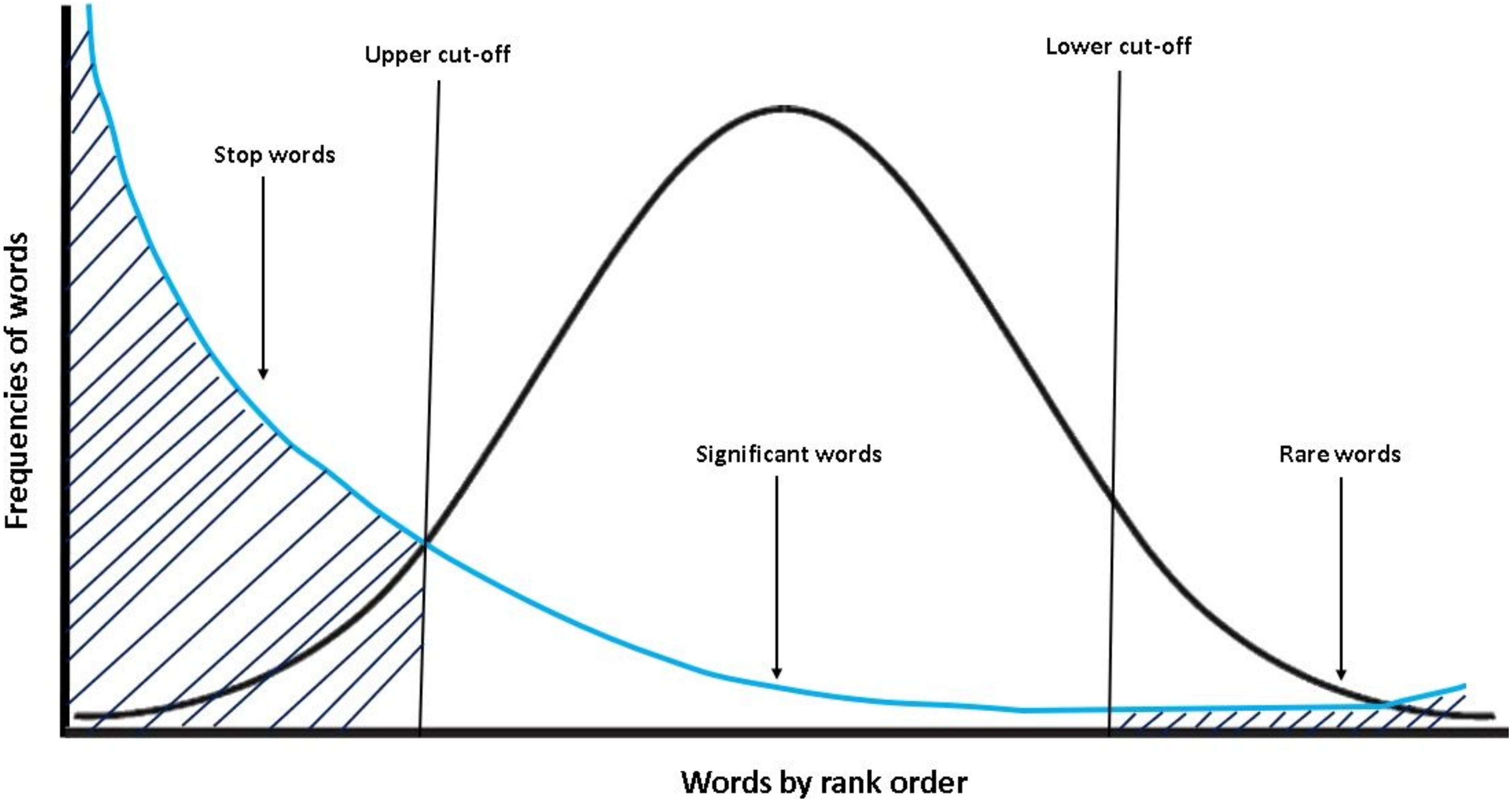
Lemmatization

- Reduce inflectional/variant forms to base form
  - For example:
    - *am, are, is* → *be*
    - *car, cars, car's, cars'* → *car*
    - *walking, walks, walked* → *walk*
  - *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing “proper” reduction to dictionary headword form

Stop words (1)

- 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 total occurrences for top 30 words
- But the trend recently 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”

Stop words (2)



Graph showing relationship between frequency and rank of words

Dictionaries

- The lexicon of the indexed corpus
  - e.g., all words on the web
  - All names, acronyms etc. (including the mis-spellings)
- Language-specific dictionary
  - **Webster's** English Dictionary
  - **WordNet** is a lexical database for the English language. It groups English words into sets of synonyms called synsets
- Domain-specific dictionary
  - The **Unified Medical Language System (UMLS)** is a compendium of many controlled vocabularies in the biomedical sciences
  - **MeSH** is a comprehensive controlled vocabulary for the purpose of indexing journal articles and books in the life sciences
  - **Loughran-McDonald Master Dictionary** collects lexicon used in finance and the sentiment each word evokes in an increasing collection of documents

References

Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze
Introduction to Information Retrieval
Cambridge University Press. 2008

The book is also online for free:

- HTML edition (2009.04.07)
- PDF of the book for online viewing (with nice hyperlink features, 2009.04.01)
- PDF of the book for printing (2009.04.01)