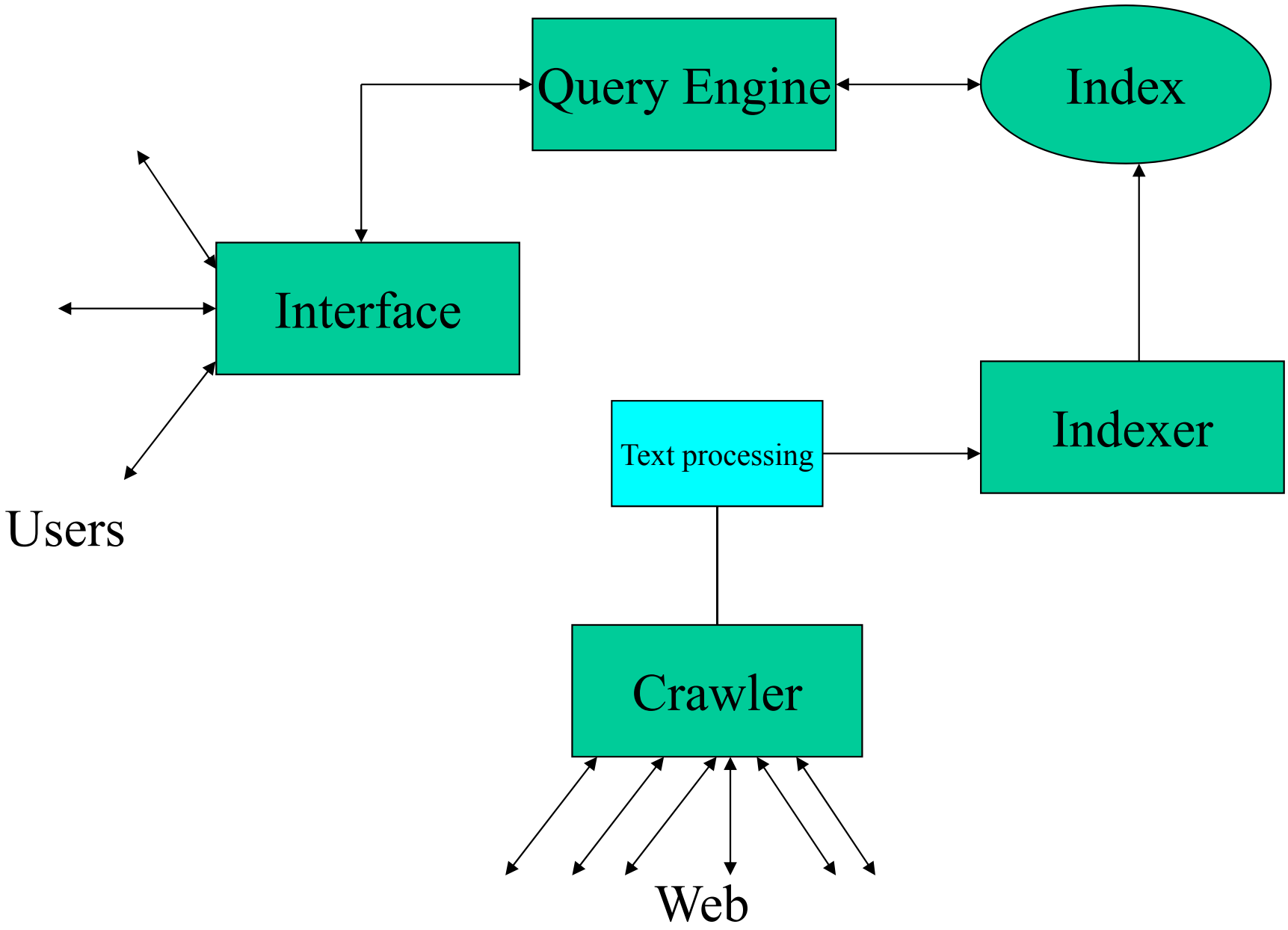
Text Processing & Characteristics

Kron

Text

- Text parsing
  - Tokenization, terms
  - A bit of linguistics
- Text characteristics
  - Zipfs law



A Typical Web Search Engine

Focus on documents

Decide what is an individual document

Can vary depending on problem

- Documents are ***basic units*** consisting of a sequence of tokens or terms and are to be indexed.
- Terms (derived from tokens) are words or roots of words, semantic units or phrases which are the atoms of indexing
- Repositories (databases) and corpora are collections of documents.
- Query is a request for documents on a query-related topic.

Building an index

- Collect documents to be indexed
  - Create your corpora
- Tokenize the text
- Linguistic processing
- Build the inverted index from terms

What is a Document?

- A **document** is a digital object
  - Indexable
    - Can be queried and potentially retrieved.
- Many types of documents
  - Text
  - Image
  - Audio
  - Video
  - Data
  - Email
  - Others?

What is Text?

- Text is so common that we often ignore its importance
- What is text?
  - Strings of characters (alphabets, ideograms, ascii, unicode, etc.)
    - Words
    - . , : ; - ( ) _
      Σψμβολσ
    - 1 2 3, 3.1415, 10^{10}
    - f = ma, H_20
    - Tables
    - Figures
  - Anything that is not an image, etc.
  - Why is text important?
    - Text is language capture
      - an instantiation of language, culture, science, etc.

Collection of text

- Corpora: collection of texts
  - especially if complete and self contained; *the corpus of Anglo-Saxon verse*
  - Special collection
- In linguistics and lexicography, a body of texts, utterances or other specimens considered more or less representative of a language and usually stored as an electronic database (The Oxford Companion to the English Language 1992)
- A collection of naturally occurring language text chosen to characterize a state or variety of a language (John Sinclair Corpus Concordance Collocation OUP 1991)

- Types:
  - Written vs Spoken
  - General vs Specialized
  - Monolingual vs Multilingual
    - e.g. Parallel, Comparable
  - Synchronic (at a particular pt in time) vs Diachronic (over time)
  - Annotated vs Unannotated
  - Indexed vs unindexed
  - Static vs dynamic

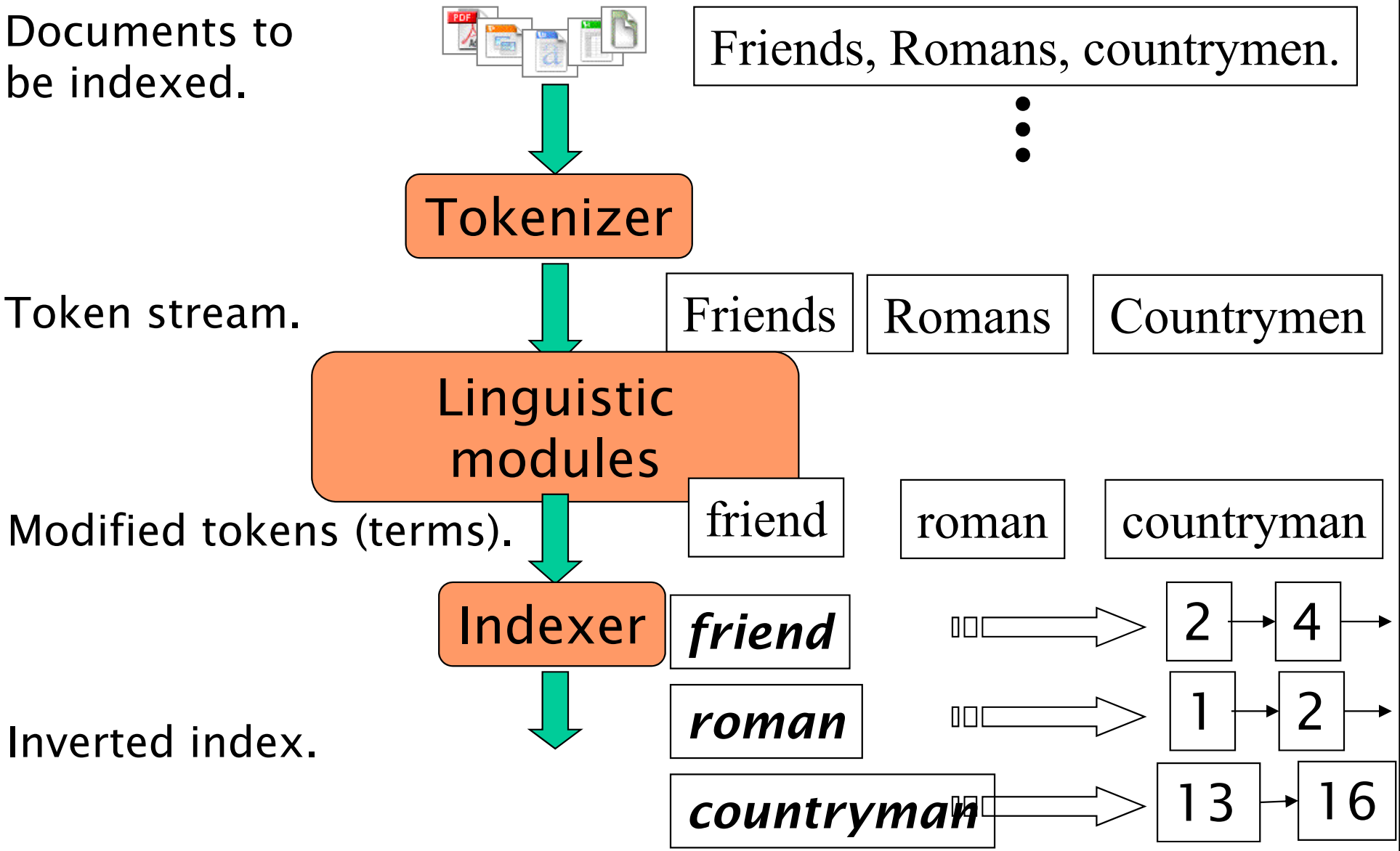
Written corpora

	Brown	LOB
Time of compilation	1960s	1970s
Compiled at	Brown University (US)	Lancaster, Oslo, Bergen
Language variety	Written American English	Written British English
Size	1 million words (500 texts of 2000 words each)	
Design	Balanced corpora; 15 genr es of text, incl. press reportage, editorials, reviews, religion, government documents, reports, biographies, scientific writing, fiction	

Text Processing

- Standard Steps:
  - Recognize document structure
    - titles, sections, paragraphs, etc.
  - Break into tokens – type of markup
    - Tokens are delimited text
      - Hello, how are you.
      - _hello_,_how_are_you_._
    - usually space and punctuation delineated
    - special issues with Asian languages
  - Stemming/morphological analysis
  - What is left are terms
  - Store in inverted index
- Lexical analysis is the process of converting a sequence of characters into a sequence of tokens.
  - A program or function which performs lexical analysis is called a lexical analyzer, lexer or scanner.

Basic indexing pipeline



Parsing a document (lexical analysis)

- What format is it in?
  - pdf/word/excel/html?
- What language is it in?
- What character set is in use?

Each of these is a classification problem which can be solved using heuristics or Machine Learning methods.
But there are complications …

Format/language stripping

- Documents being indexed can include docs from many different languages
  - A single index may have to contain terms of several languages.
- Sometimes a document or its components can contain multiple languages/formats
  - French email with a Portuguese pdf attachment.
- What is a unit document?
  - An email?
  - With attachments?
  - An email with a zip containing documents?

Document preprocessing

- Convert byte sequences into a linear sequence of characters
- Trivial with ascii, but not so with Unicode or others
  - Use ML classifiers or heuristics.

- *Crucial problem for commercial system!*

Tokenization

- Parsing (chopping up) the document into basic units that are candidates for later indexing
  - What parts of text to use and what not
- Issues with
  - Punctuation
  - Numbers
  - Special characters
  - Equations
  - Formula
  - Languages
  - Normalization (often by stemming)

Tokenization

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

Tokenization

- Issues in tokenization:
  - ***Finland's capital*** $\rightarrow$

    Finland? Finlands? Finland's?
  - ***Hewlett-Packard*** $\rightarrow$ ***Hewlett*** and ***Packard*** as two tokens?
    - ***State-of-the-art***: break up hyphenated sequence.
    - co-education ?
    - the hold-him-back-and-drag-him-away-maneuver ?
  - ***San Francisco***: one token or two? How do you decide it is one token?

Numbers

- ***3/12/91***
- ***Mar. 12, 1991***
- ***55 B.C.***
- ***B-52***
- ***My PGP key is 324a3df234cb23e***
- ***100.2.86.144***
  - Generally, don't index as text.
  - Will often index "meta-data" separately
    - Creation date, format, etc.

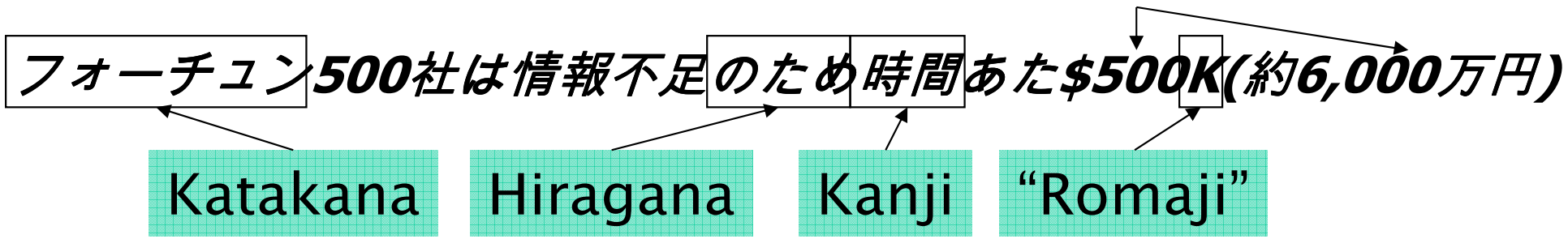
Tokenization: Language issues

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

- German noun compounds are not segmented
  - Lebensversicherungsgesellschaftsangestellter
  - 'life insurance company employee'

Tokenization: language issues

- Chinese and Japanese have no spaces between words:
  - Not always guaranteed a unique tokenization
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats



End-user can express query entirely in hiragana!

Tokenization: language issues

- 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.'
- With Unicode, the surface presentation is complex, but the stored form is straightforward

Normalization

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

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

Normalizing Punctuation

- ***Ne'er vs. never***: use language-specific, handcrafted "locale" to normalize.
  - Which language?
  - Most common: detect/apply language at a pre-determined granularity: doc/paragraph.
- ***U.S.A.*** vs. ***USA*** – remove all periods or use locale.
- ***a.out***

Thesauri and soundex

- Handle synonyms and homonyms
  - Hand-constructed equivalence classes
    - e.g., ***car*** = ***automobile***
    - ***color*** = ***colour***
- Rewrite to form equivalence classes
- Index such equivalences
  - When the document contains ***automobile***, index it under ***car*** as well (usually, also vice-versa)
- Or expand query?
  - When the query contains ***automobile***, look under ***car*** as well

Soundex

- Traditional class of heuristics to expand a query into phonetic equivalents
  - Language specific – mainly for names
  - E.g., ***chebyshev*** → ***tchebycheff***

Stemming and Morphological Analysis

- Goal: “normalize” similar words
- Morphology (“form” of words)
  - Inflectional Morphology
    - E.g,. inflect verb endings and noun number
    - Never change grammatical class
      - *dog, dogs*
  - Derivational Morphology
    - Derive one word from another,
    - Often change grammatical class
      - *build, building; health, healthy*

Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*
- Lemmatization implies doing "proper" reduction to dictionary headword form

Stemming

Morphological variants of a word (morphemes). Similar terms derived from a common stem:

engineer, engineered, engineering
use, user, users, used, using

Stemming in Information Retrieval. Grouping words with a common stem together.

For example, a search on *reads*, also finds *read*, *reading*, and *readable*

Stemming consists of removing **suffixes** and **conflating** the resulting morphemes. Occasionally, **prefixes** are also removed.

Stemming

- Reduce terms to their "roots" before indexing
- "Stemming" suggest crude affix 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 exampl compress and compress ar both accept as equival to compress

Porter's algorithm

- Commonest algorithm for stemming English
  - Results suggest at least as good as other stemming options
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*

Typical rules in Porter

- *sses* → *ss*
- *ies* → *i*
- *ational* → *ate*
- *tional* → *tion*

- Weight of word sensitive rules
- *(m>1) EMENT* →
  - *replacement* → *replac*
  - *cement* → *cement*

Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
  http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
  - Single-pass, longest suffix removal (about 250 rules)
  - Motivated by Linguistics as well as IR
- Full morphological analysis – at most modest benefits for retrieval
- Do stemming and other normalizations help?
  - Often very mixed results: really help recall for some queries but harm precision on others

Automated Methods are the norm

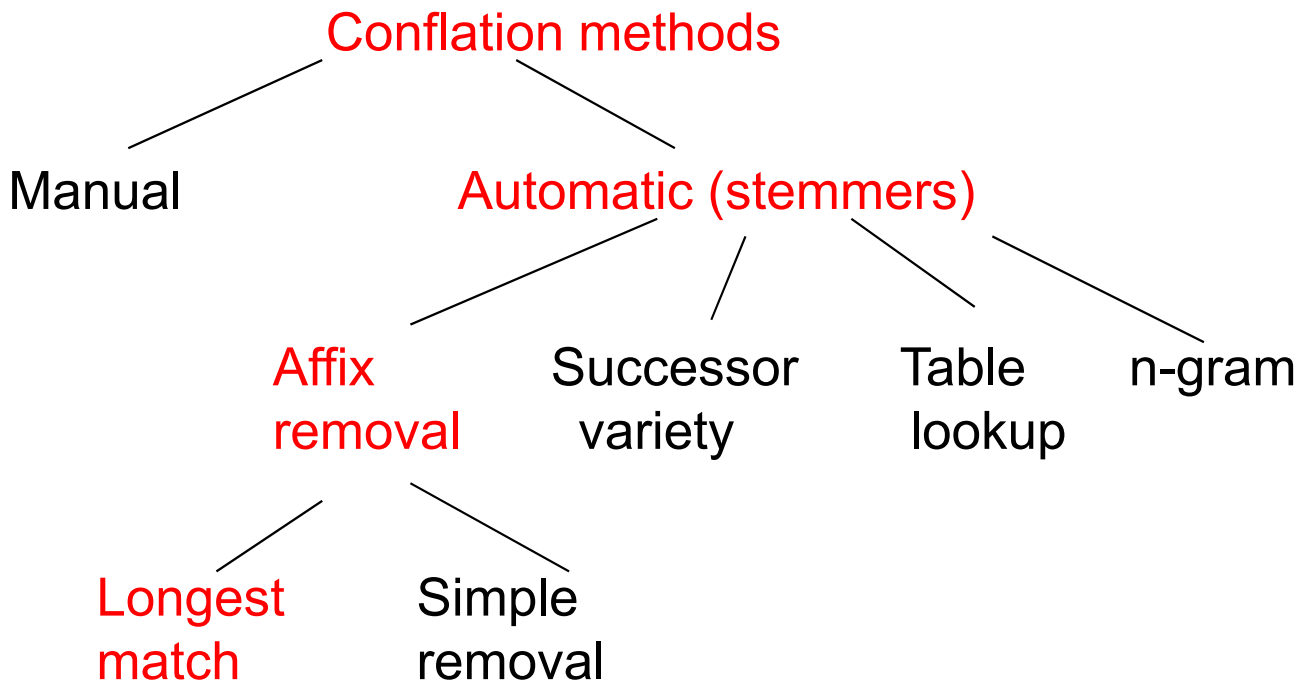
- Powerful multilingual tools exist for morphological analysis
  - PCKimmo, Xerox Lexical technology
  - Require a grammar and dictionary
  - Use “two-level” automata
- Stemmers:
  - Very dumb rules work well (for English)
  - Porter Stemmer: Iteratively remove suffixes
  - Improvement: pass results through a lexicon

Porter's algorithm

- Commonest algorithm for stemming English
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*
- Typical rules
  - *sses* → *ss*
  - *ies* → *i*
  - *ational* → *ate*
  - *tional* → *tion*

Categories of Stemmer

The following diagram illustrate the various categories of stemmer. Porter's algorithm is shown by the red path.



Comparison of stemmers

Stemmer	Does	Does not
Simple Stemmer	• simple plural removal e.g. rates--rate; studies--study	• irregulated plural form (eg. women, teeth, etc) • other morphological variations e.g. -ing, -ful, -ness, -able • tense
Porter Stemmer	• plural removal • morphological variations	• irregulated plural form (eg. women, teeth, etc) • doesn't work very well with words end with -y or -able e.g. sleepy---sleepi; doable--doabl • tense
Inflectional Stemmer (Krovetz)	logic: check the online dictionary prior to stemming any word, and if it is found then that is retured as the result. If it is not found , replace 'ing'/'es'/'ed' with 'e', and check the dictionary again. If this fails the entire suffix is remov ed and the dictionary is checked once more. Therefore: • plural removal • some variations	• irregulated plural form (eg. women, teeth, etc) • various morphological variations that exist in the dictionary -ness, -able, e.g., hopefulness --hopefulness, workable ---workable • tense

Stemming in Practice

Evaluation studies have found that stemming can affect retrieval performance, usually for the better, but the results are mixed.

- Effectiveness is dependent on the vocabulary. Fine distinctions may be lost through stemming.
- Automatic stemming is as effective as manual conflation.
- Performance of various algorithms is similar.

Porter's Algorithm is entirely empirical, but has proved to be an effective algorithm for stemming English text with trained users.

Language-specificity

- Many of the above features embody transformations that are
  - Language-specific and
  - Often, application-specific
- These are “plug-in” addenda to the indexing process
- Both open source and commercial plug-ins available for handling these

Normalization: other languages

- Accents: ***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

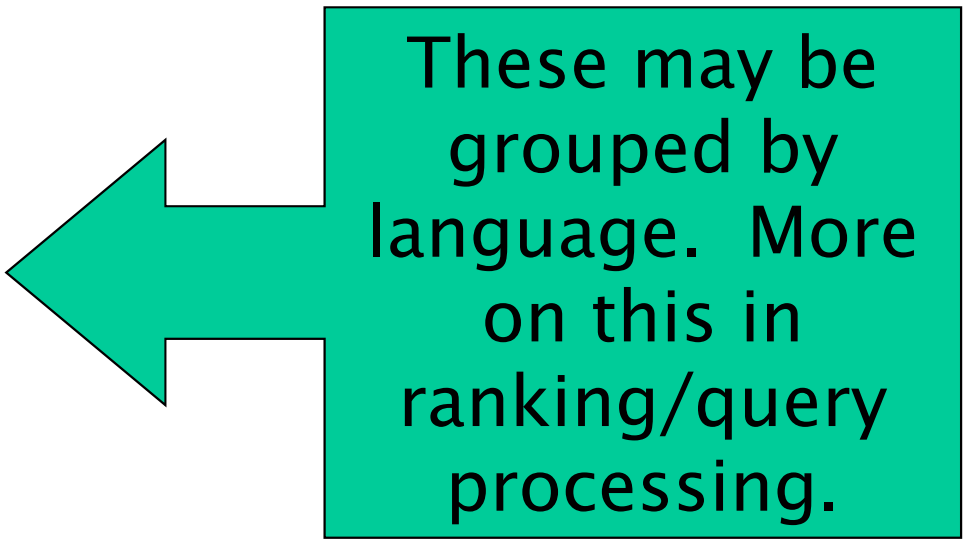
- Character-level alphabet detection and conversion
  - Tokenization not separable from this.
  - Sometimes ambiguous:

Morgen will ich in MIT …

Is this German "mit"?

Dictionary entries – first cut

ensemble.french
時間.japanese
MIT.english
mit.german
guaranteed.english
entries.english
sometimes.english
tokenization.english



Text Documents

A text digital document consists of a sequence of **words** and other symbols, e.g., punctuation.

The individual words and other symbols are known as **tokens** or **terms**.

A textual document can be:

- **Free text**, also known as **unstructured** text, which is a continuous sequence of tokens.
- **Fielded text**, also known as **structured** text, in which the text is broken into sections that are distinguished by tags or other markup.

Example?

Why the focus on text?

- Language is the most powerful query model
- Language can be treated as text
- Others?

Text Based Information Retrieval

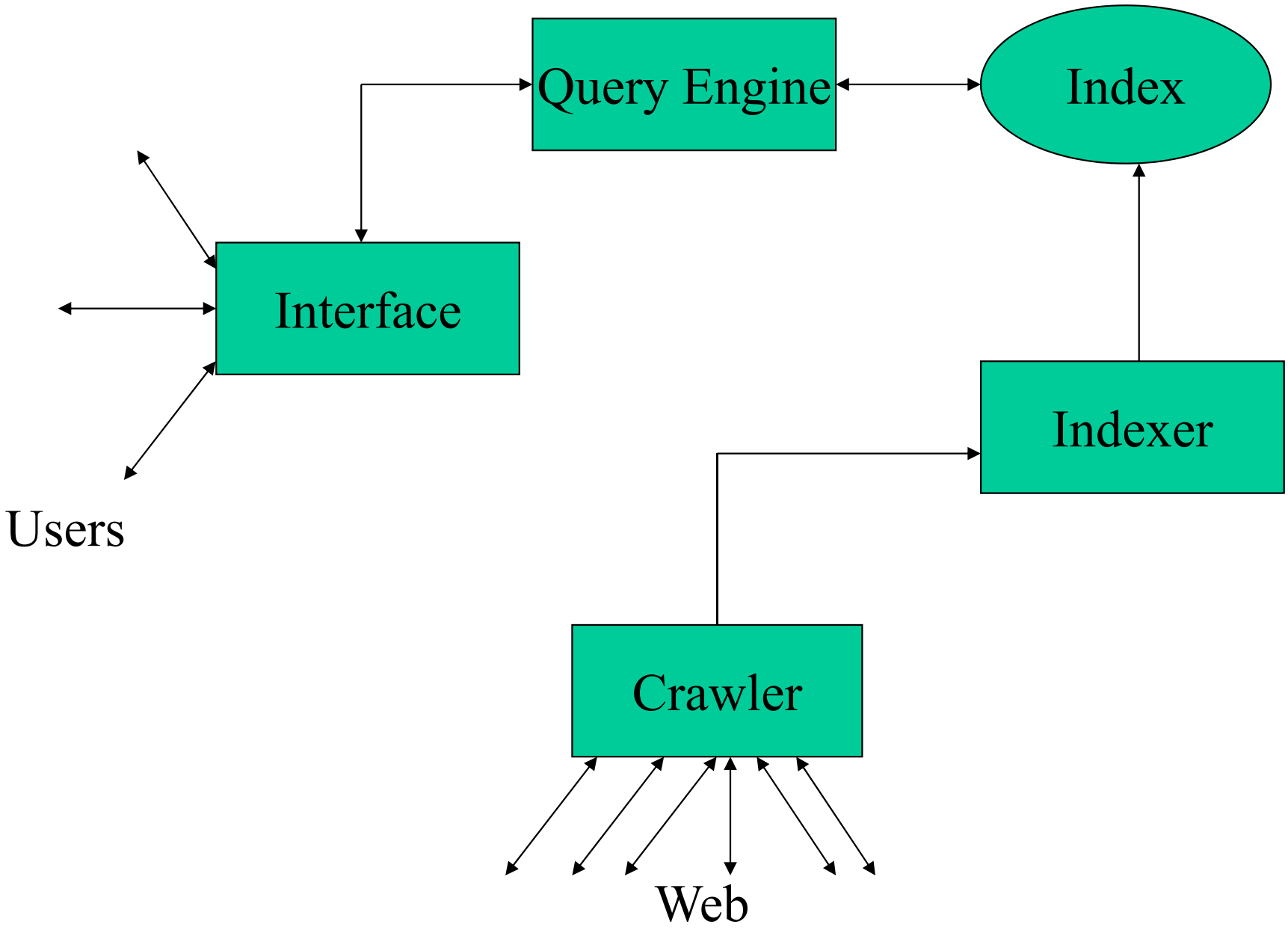
Most **matching** methods are based on **Boolean operators**.

Most **ranking** methods are based on the **vector space model.**

Web search methods combine vector space model with ranking based on **importance** of documents.

Many practical systems combine features of several approaches.

In the basic form, all approaches treat **words** as **separate tokens** with minimal attempt to interpret them linguistically.



A Typical Web Search Engine

Content Analysis of Text

- Automated Transformation of raw text into a form that represent some aspect(s) of its meaning
- Including, but not limited to:
  - Token creation
  - Matrices and Vectorization
  - Phrase Detection
  - Categorization
  - Clustering
  - Summarization

Techniques for Content Analysis

- Statistical / vector
  - Single Document
  - Full Collection
- Linguistic
  - Syntactic
  - Semantic
  - Pragmatic
- Knowledge-Based (Artificial Intelligence)
- Hybrid (Combinations)

Stop Lists

•Very common words, such as *of*, *and*, *the*, are rarely of use in information retrieval.

•A **stop list** is a list of such words that are removed during lexical analysis.

•A long stop list saves space in indexes, speeds processing, and eliminates many false hits.

•However, common words are sometimes significant in information retrieval, which is an argument for a short stop list. (Consider the query, "To be or not to be?")

Suggestions for Including Words in a Stop List

- **Include** the most common words in the English language (perhaps 50 to 250 words).
- **Do not include** words that might be important for retrieval (Among the 200 most frequently occurring words in general literature in English are *time*, *war*, *home*, *life*, *water*, and *world*).
- In addition, **include** words that are very common in context (e.g., *computer*, *information*, *system* in a set of computing documents).

Example: the WAIS stop list (first 84 of 363 multi-letter words)

about	above	according	across	actually	adj
after	afterwards	again	against	all	almost
alone	along	already	also	although	always
among	amongst	an	another	any	anyhow
anyone	anything	anywhere	are	aren't	around
at	be	became	because	become	becomes
becoming	been	before	beforehand	begin	beginning
behind	being	below	beside	besides	between
beyond	billion	both	but	by	can
can't	cannot	caption	co	could	couldn't
did	didn't	do	does	doesn't	don't
down	during	each	eg	eight	eighty
either	else	elsewhere	end	ending	enough
etc	even	ever	every	everyone	everything

Stop list policies

How many words should be in the stop list?

- Long list lowers recall

Which words should be in list?

- Some common words may have retrieval importance:
  -- *war, home, life, water, world*
- In certain domains, some words are very common:
  -- *computer, program, source, machine, language*

There is very little systematic evidence to use in selecting a stop list.

Stop Lists in Practice

The modern tendency is:

(a) have very short stop lists for broad-ranging or multi-lingual document collections, especially when the users are not trained.

(b) have longer stop lists for document collections in well-defined fields, especially when the users are trained professional.

Token generation - stemming

- What are tokens for documents?
  - Words (things between spaces)
- Some words equivalent
- Stemming finds equivalences among words
- Removal of grammatical suffixes

Stemming

- Reduce terms to their roots before indexing
  - 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 exampl compres and compres are both accept as equival to compres.

Selection of tokens, weights, stop lists and stemming

Special purpose collections (e.g., law, medicine, monographs)

Best results are obtained by tuning the search engine for the characteristics of the collections and the expected queries.

It is valuable to use a **training set** of queries, with lists of relevant documents, to tune the system for each application.

General purpose collections (e.g., web search)

The modern practice is to use a basic weighting scheme (e.g., *tf.idf*), a simple definition of token, a short stop list and no stemming except for plurals, with minimal conflation.

Web searching combine similarity ranking with ranking based on document importance.

Analyser for Lucene

- Tokenization: Create an Analyser
  - Options
    - WhitespaceAnalyzer
      - divides text at whitespace
    - SimpleAnalyzer
      - divides text at non-letters
      - convert to lower case
    - StopAnalyzer
      - SimpleAnalyzer
      - removes stop words
    - StandardAnalyzer
      - good for most European Languages
      - removes stop words
      - convert to lower case

org.apache.lucene.analysis

Class Analyzer

java.lang.Object
  └ **org.apache.lucene.analysis.Analyzer**

Direct Known Subclasses:
BrazilianAnalyzer, ChineseAnalyzer, CJKAnalyzer, CzechAnalyzer, DutchAnalyzer, FrenchAnalyzer, GermanAnalyzer, GreekAnalyzer, KeywordAnalyzer, PatternAnalyzer, PerFieldAnalyzerWrapper, RussianAnalyzer, SimpleAnalyzer, SnowballAnalyzer, StandardAnalyzer, StopAnalyzer, ThaiAnalyzer, WhitespaceAnalyzer

---

public abstract class **Analyzer**
extends Object

An Analyzer builds TokenStreams, which analyze text. It thus represents a policy for extracting index terms from text.

Typical implementations first build a Tokenizer, which breaks the stream of characters from the Reader into raw Tokens. One or more TokenFilters may then be applied to the output of the Tokenizer.

WARNING: You must override one of the methods defined by this class in your subclass or the Analyzer will enter an infinite loop.

Constructor Summary

Analyzer()

Method Summary

int	**getPositionIncrementGap**(String fieldName) Invoked before indexing a Fieldable instance if terms have already been added to that field.
abstract TokenStream	**tokenStream**(String fieldName, Reader reader) Creates a TokenStream which tokenizes all the text in the provided Reader.

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

Analyzer

public **Analyzer**()

Example of analyzing a document

```
Analzying "Lee Giles teaches IST441 his email address is giles@ist.psu.edu"
      org.apache.lucene.analysis.WhitespaceAnalyzer:
            [Lee] [Giles] [teaches] [IST441] [his] [email] [address] [is] [giles@ist.psu.edu]

      org.apache.lucene.analysis.SimpleAnalyzer:
            [lee] [giles] [teaches] [ist] [his] [email] [address] [is] [giles] [ist] [psu] [edu]

      org.apache.lucene.analysis.StopAnalyzer:
            [lee] [giles] [teaches] [ist] [his] [email] [address] [giles] [ist] [psu] [edu]

      org.apache.lucene.analysis.standard.StandardAnalyzer:
            [lee] [giles] [teaches] [ist441] [his] [email] [address] [giles@ist.psu.edu]
```

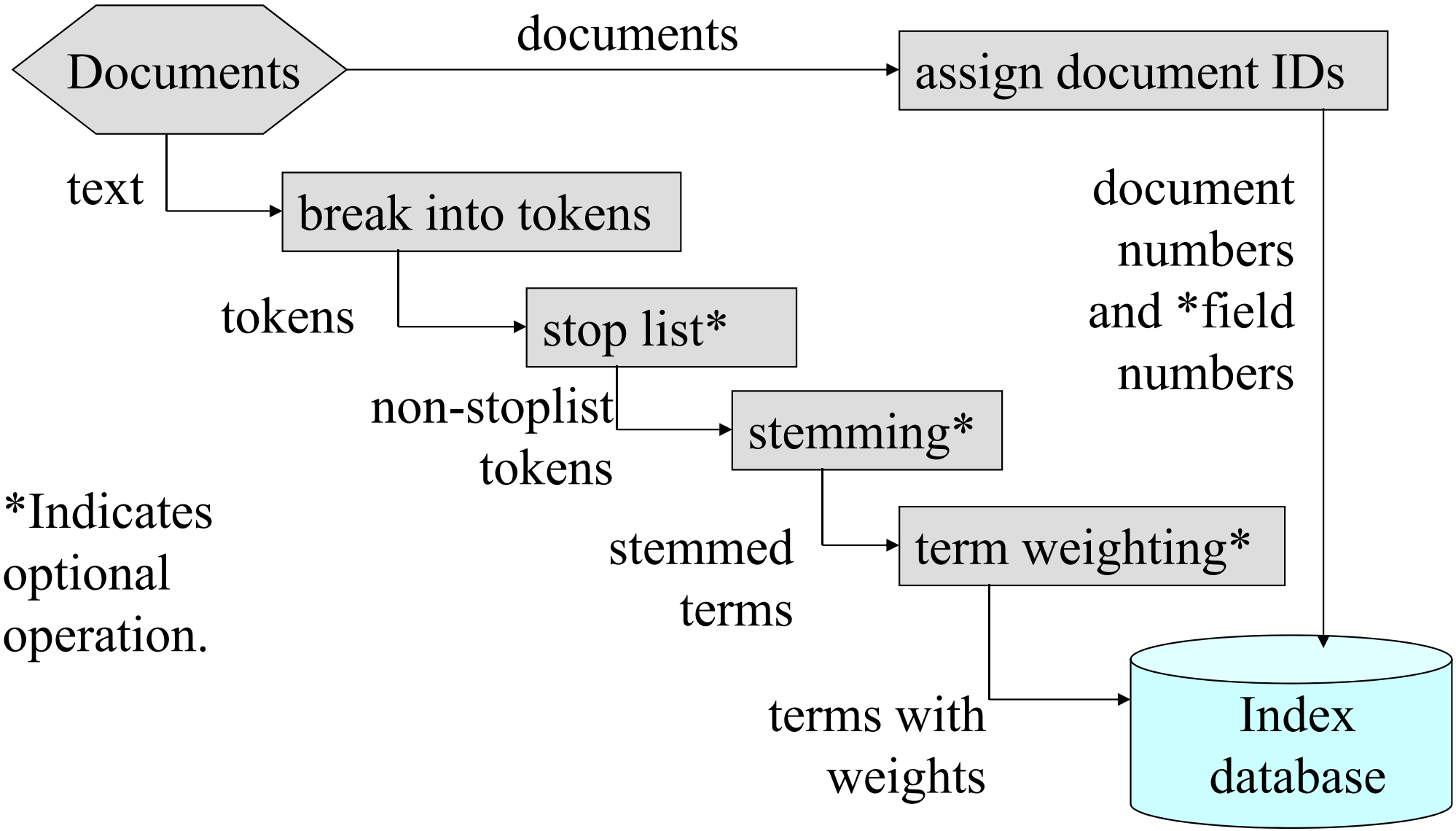
Other Analyzers

- Also available
  - GermanAnalyzer
  - RussianAnalyzer
  - (Lucene Sandbox)
    - BrazilianAnaylzer
    - ChineseAnalyzer (UTF-8)
    - CzechAnalyzer
    - DutchAnalyzer
    - FrenchAnalyzer
    - GreekAnalyzer
    - KoreanAnalyzer
    - JapaneseAnalyzer

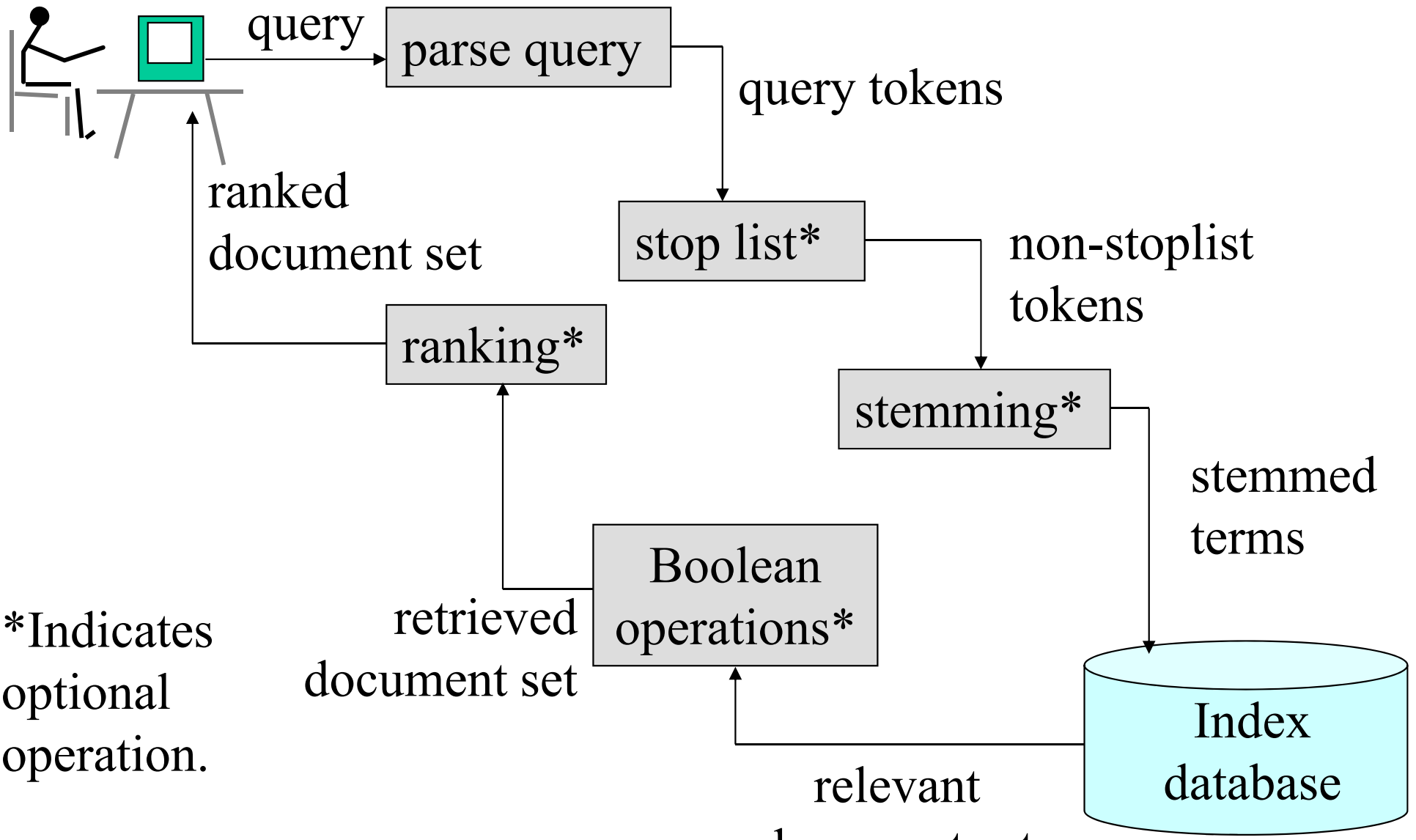
Summary of Text

- Text is reduced to tokens
- Stop words can be removed
- Stemmers widely used for token generation
  - Porter stemmer most common

Indexing Subsystem



Search Subsystem



Bibliographic Entry	Result (w/surrounding text)	Standardized Result
Robert McCrum, William Cran, & Robert MacNeil. *The Story of English*. New York: Penguin, 1992: 1	"The statistics of English are astonishing. Of all the world's languages (which now number some 2,700), it is arguably the richest in vocabulary. The compendious *Oxford English Dictionary* lists about 500,000 words; and a further half-million technical and scientific terms remain uncatalogued. According to traditional estimates, neighboring German has a vocabulary of about 185,000 and French fewer than 100,000, including such Franglais as *le snacque-barre* and *le hit-parade*."	500,000 words 1,000,000 words (including scientific words)
Encyclopedia Americana, Volume 10. Grolier, 1999.	"The vocabulary has grown from the 50,000 to 60,000 words in Old English to the tremendous number of entries -- 650,000 to 750,000 -- in an unabridged dictionary of today."	650 - 750,000 words
Oxford English Dictionary, Second Edition, Volume 1. Oxford University Press, 1989.	"In addition to the headwords of main entries, the Dictionary contains 157,000 combinations and derivatives in bold type, and 169,000 phrases and combinations in bold italic type, making a total of 616,500 word-forms."	616,500 words
Webster's Third New International Dictionary. G&C Merriam Co., 1971.	"This dictionary has a vocabulary of over 450,000 words."	> 450,000 words
Wilton, David. How Many Words Are There In The English Language? Wilton's Word & Phrase Origins. 7 February 2001.	"The OED2, the largest English-language dictionary, contains some 290,000 entries with some 616,500 word forms."	616,500 words

Have you ever encountered a person who just keeps on rambling on and on with no end in sight. If you have, you might have wondered if he/she would ever run out of words to say. Unfortunately, that will remain a dream for all of us.

As we enter the Twenty First Century, English is the most widely spoken and written language on Earth. English was first spoken in Britain by Germanic tribes in Fifth Century AD also known as the Old English (Anglo-Saxon) period. During the Middle English period (1150-1500 AD), a lot of the Old English word endings were replaced by prepositions like by, with, and from. We are now in the Modern English period which started in the Sixteenth Century.

The number of words in English has grown from 50,000 to 60,000 words in Old English to about a million today. There are a number of ways in which the English vocabulary increases. The principal way in which it grows is by borrowing words from other languages. About 80% of the entries in any English dictionary are borrowed, mainly from Latin. Another way is by combining words into one word such as housewife, greenhouse, and overdue. The addition of prefixes and suffixes to words also increases the immense vocabulary of the English language.

Today, more than 750 million people use the English language. An average educated person knows about 20,000 words and uses about 2,000 words in a week. Despite its widespread use, there are only about 350 million people who use it as their mother tongue. It is the official language of the Olympics. More than half of the world's technical and scientific periodicals as well three quarters of the world's mail, and its telexes and cables are in English. About 80% of the information stored in the world's computers (such as this text) are also in English. English is also transmitted to more than 100 million people everyday by 5 of the largest broadcasting companies (CBS, NBC, ABC, BBC, CBC). It seems like English will remain the most widely used language for some time.

Statistical Properties of Text

- Token occurrences in text are not uniformly distributed
- They are also not normally distributed
- They do exhibit a Zipf distribution

A More Standard Collection

Government documents, 157734 tokens, 32259 unique

8164 the	969 on	1 ABC
4771 of	915 FT	1 ABFT
4005 to	883 Mr	1 ABOUT
2834 a	860 was	1 ACFT
2827 and	855 be	1 ACI
2802 in	849 Pounds	1 ACQUI
1592 The	798 TEXT	1 ACQUISITIONS
1370 for	798 PUB	1 ACSIS
1326 is	798 PROFILE	1 ADFT
1324 s	798 PAGE	1 ADVISERS
1194 that	798 HEADLINE	1 AE
973 by	798 DOCNO	

Plotting Word Frequency by Rank

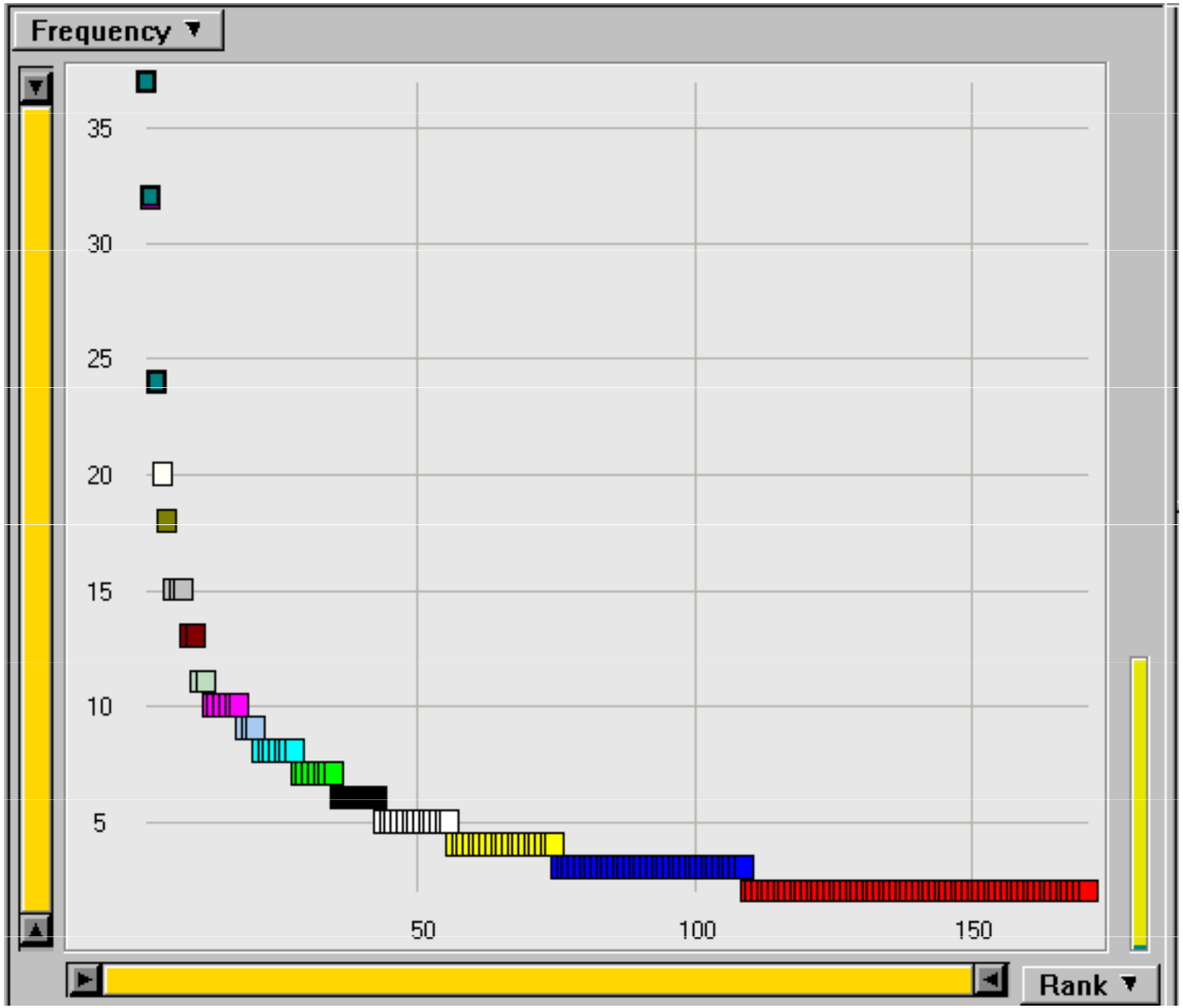
- Main idea: count
  - How many times tokens occur in the text
    - Over all texts in the collection
- Now rank these according to how often they occur. This is called the rank.

Most and Least Frequent Terms

Rank	Freq	Term
1	37	system
2	32	knowledg
3	24	base
4	20	problem
5	18	abstract
6	15	model
7	15	languag
8	15	implem
9	13	reason
10	13	inform
11	11	expert
12	11	analysi
13	10	rule
14	10	program
15	10	oper
16	10	evalu
17	10	comput
18	10	case
19	9	gener
20	9	form
150	2	enhanc
151	2	energi
152	2	emphasi
153	2	detect
154	2	desir
155	2	date
156	2	critic
157	2	content
158	2	consider
159	2	concern
160	2	compon
161	2	compar
162	2	commerci
163	2	clause
164	2	aspect
165	2	area
166	2	aim
167	2	affect

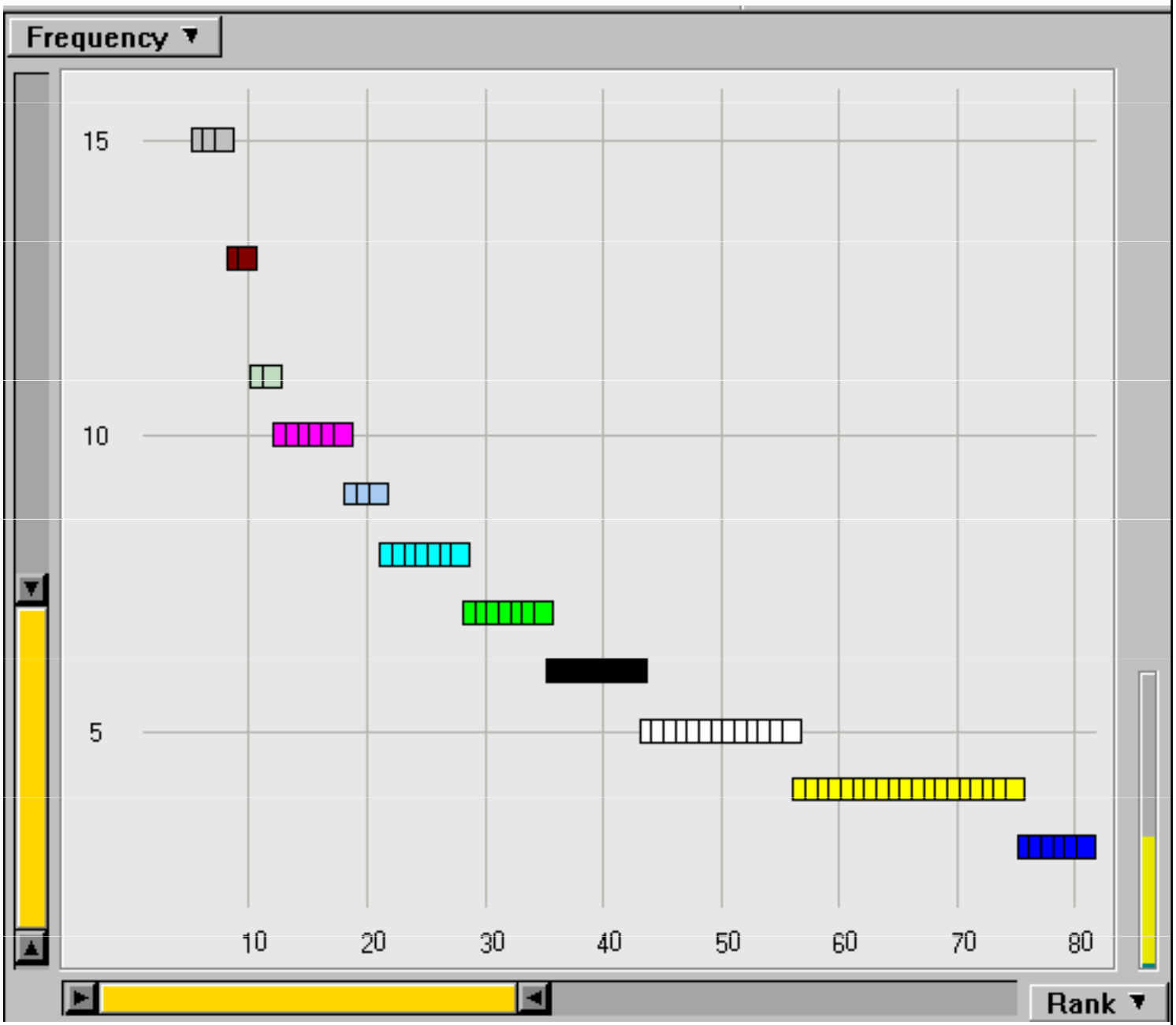
The Corresponding Zipf Curve

Rank	Freq	
1	37	system
2	32	knowledg
3	24	base
4	20	problem
5	18	abstract
6	15	model
7	15	languag
8	15	implem
9	13	reason
10	13	inform
11	11	expert
12	11	analysi
13	10	rule
14	10	program
15	10	oper
16	10	evalu
17	10	comput
18	10	case
19	9	gener
20	9	form



Zoom in on the Knee of the Curve

43	6	approach
44	5	work
45	5	variabl
46	5	theori
47	5	specif
48	5	softwar
49	5	requir
50	5	potenti
51	5	method
52	5	mean
53	5	inher
54	5	data
55	5	commit
56	5	applic
57	4	tool
58	4	technolog
59	4	techniqu



Zipf Distribution

- The Important Points:
  - a few elements occur *very frequently*
  - a medium number of elements have medium frequency
  - many elements occur *very infrequently*
  - Self similarity
    - Same shape for large and small frequency words
  - Long tail
  - Not necessarily obeys central limit theorem

Zipf Distribution

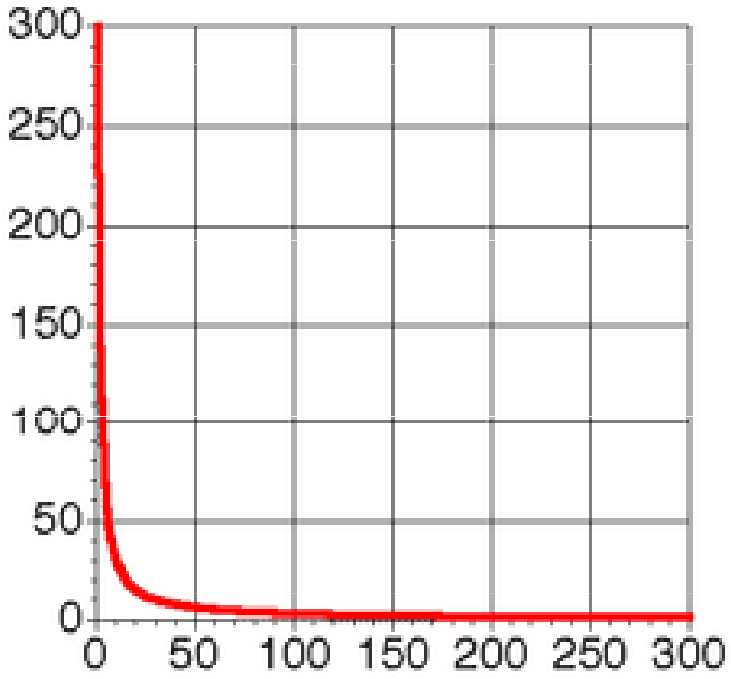
- The product of the frequency of words (f) and their rank (r) is approximately constant
  - Rank = order of words' frequency of occurrence

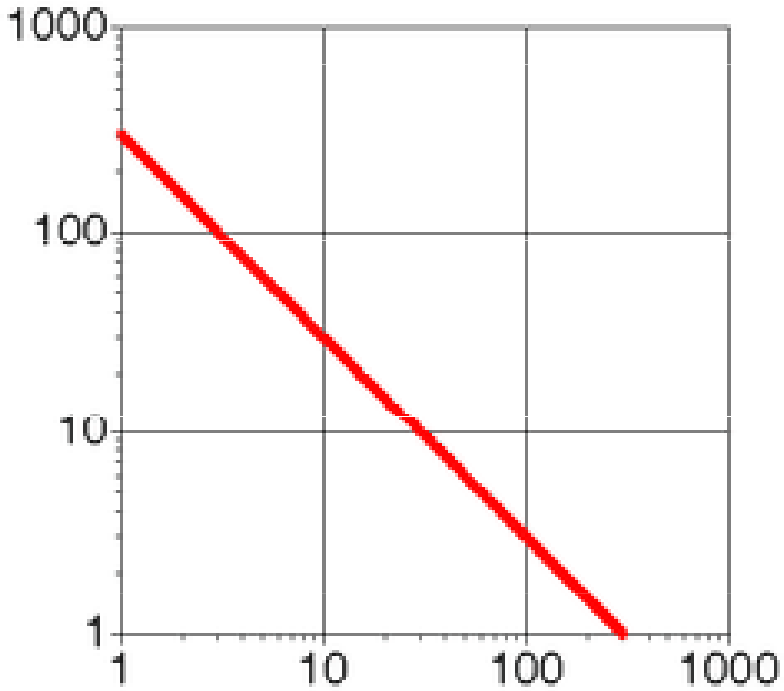
$$f = C * 1/r$$

$$C \cong N/10$$

- Another way to state this is with an approximately correct rule of thumb:
  - Say the most common term occurs C times
  - The second most common occurs C/2 times
  - The third most common occurs C/3 times
  - …

Zipf Distribution (linear and log scale)





What Kinds of Data Exhibit a Zipf Distribution?

- Words in a text collection
  - Virtually any language usage
- Library book checkout patterns
- Incoming Web Page Requests (Nielsen)
- Outgoing Web Page Requests (Cunha & Crovella)
- Document Size on Web (Cunha & Crovella)
- Many sales with certain retailers

Power Laws

Power law distributions on ranked data are often called Zipf distributions, or Pareto-Zipf distributions. In language applications, for instance, the distribution of words ranked by their frequency in a large corpus of text is invariably a power law distribution, also known as Zipf's law. There is a vast literature on power law distributions and the related log-normal distribution (e.g. Mitzenmacher (2002)).

A discrete power law distribution with coefficient $\gamma > 1$ is a distribution of the form

$$P(X = k) = Ck^{-\gamma} \tag{1.28}$$

for $k = 1, 2, \ldots$. The corresponding density in the continuous case is

$$f(x) = Cx^{-\gamma} \tag{1.29}$$

for $x \in [1, +\infty)$. In many real life situations associated with power law distributions, the distribution for small values of k or x may deviate from the expressions in Equations 1.28 and 1.29. Thus, a more flexible definition is to say that Equations 1.28 and 1.29 describe the behavior for sufficiently large values of x or k.

Power Law Statistics - problems with means

In both the discrete and continuous cases, moments of order $m \geq 0$ are finite if and only if $\gamma > m + 1$. In particular, the expectation is finite if and only if $\gamma > 2$, and the variance is finite if and only if $\gamma > 3$. In the discrete case,

$$E[X^m] = \sum_{k=1}^{\infty} Ck^{m-\gamma} = C\zeta(\gamma - m) \tag{1.30}$$

where ζ is Riemann's zeta function ($\zeta(s) = \sum_k 1/k^s$). In the continuous case,

$$E[X^m] = \int_1^{\infty} Cx^{m-\gamma} \mathrm{d}x = \frac{C}{\gamma - m - 1} \tag{1.31}$$

In particular, $C = \gamma - 1$, $E[X] = \gamma - 1/(\gamma - 2)$ and $\mathrm{Var}[X] = \frac{\gamma-1}{\gamma-3} - (\frac{\gamma-1}{\gamma-2})^2$.

A simple consequence is that in a power law distribution the average behavior is not the most frequent or the most typical, in sharp contrast with what is observed with, for instance, a Gaussian distribution. Furthermore, the total mass of the points to the right of the average is greater than the total mass of points to the left. Thus in a random sample, the majority of points are to the right of the average.

Power-law distributions

- The degree distributions of most real-life networks follow a power law

$$p(k) = Ck^{-\alpha}$$

- Right-skewed/Heavy-tail distribution
  - there is a non-negligible fraction of nodes that has very high degree (hubs)
  - scale-free: no characteristic scale, average is not informative

- In stark contrast with the random graph model!
  - Poisson degree distribution, z=np

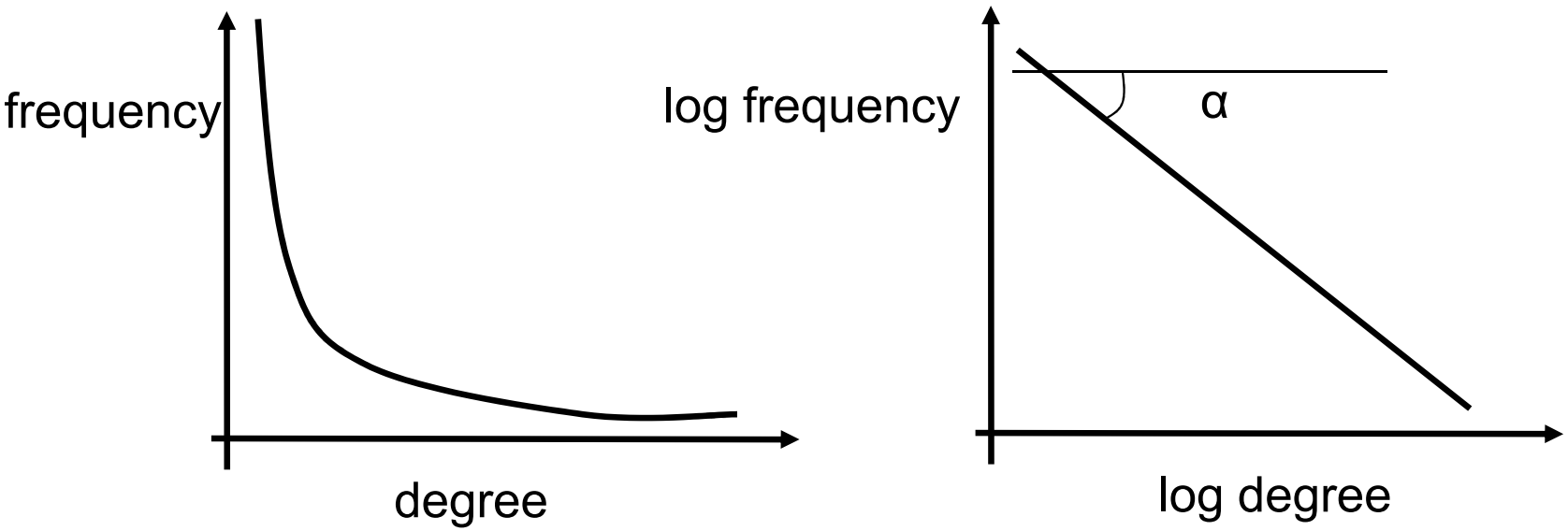
$$p(k) = P(k;z) = \frac{z^k}{k!}e^{-z}$$

  - highly concentrated around the mean
  - the probability of very high degree nodes is exponentially small

Power-law signature

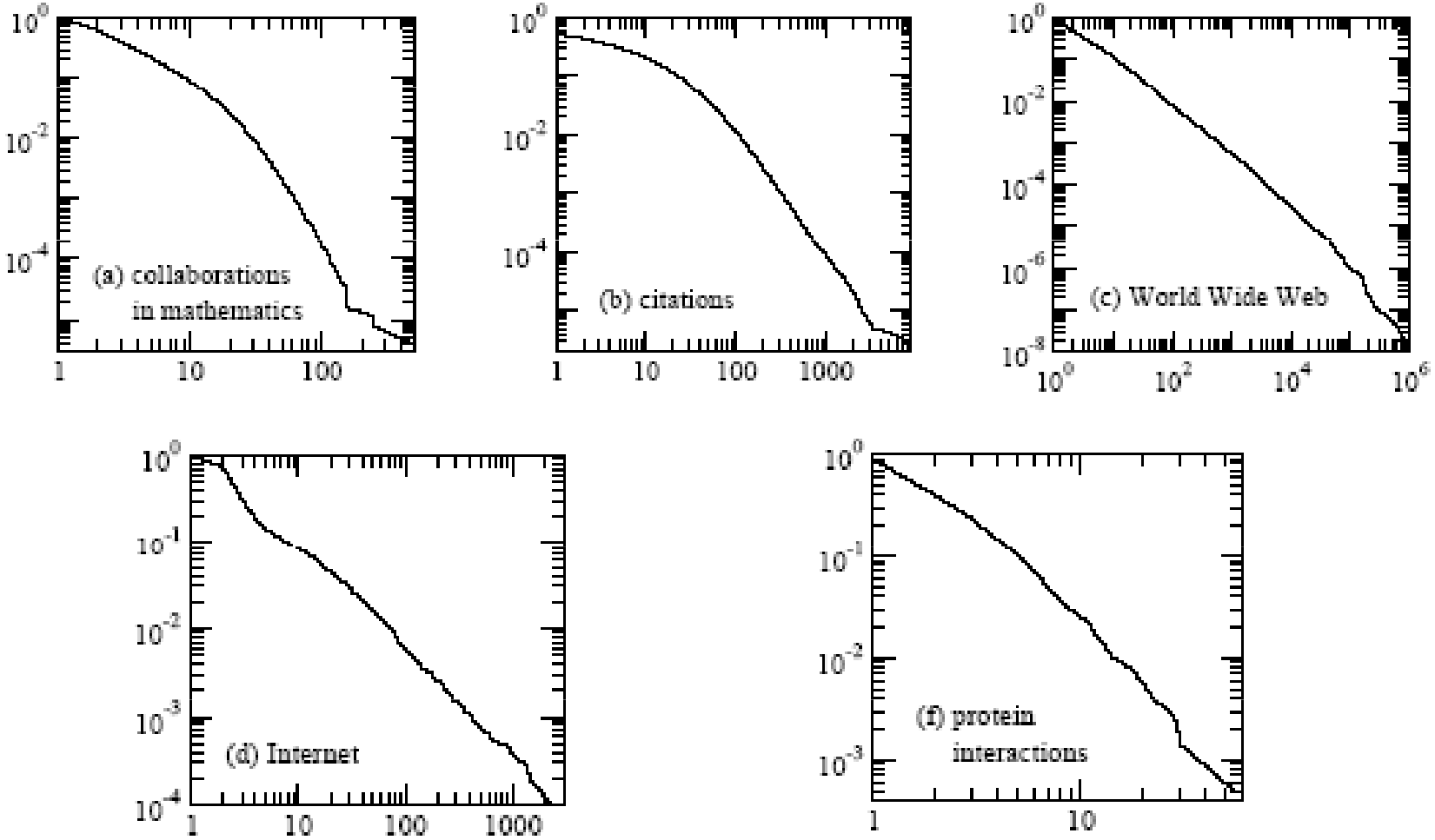
- Power-law distribution gives a line in the log-log plot

$$\log p(k) = -\alpha \log k + \log C$$



α : power-law exponent (typically $2 \le \alpha \le 3$)

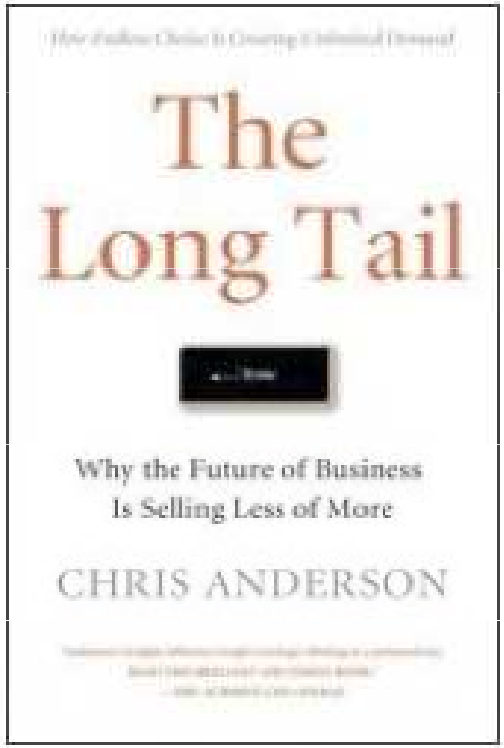
Examples of degree distribution for power laws



Taken from [Newman 2003]

Power Law Statistics - long tails

Power of the long tail:



The phrase The Long Tail, as a proper noun, was first coined by Chris Anderson. The concept drew in part from an influential February 2003 essay by Clay Shirky, "Power Laws, Weblogs and Inequality" that noted that a relative handful of weblogs have many links going into them but "the long tail" of millions of weblogs may have only a handful of links going into them. Beginning in a series of speeches in early 2004 and culminating with the publication of a Wired magazine article in October 2004, Anderson described the effects of the long tail on current and future business models. Anderson later extended it into the book The Long Tail: Why the Future of Business is Selling Less of More (2006).

Anderson argued that products that are in low demand or have low sales volume can collectively make up a market share that rivals or exceeds the relatively few current bestsellers and blockbusters, if the store or distribution channel is large enough. Examples of such mega-stores include the online retailer Amazon.com and the online video rental service Netflix. The Long Tail is a potential market and, as the examples illustrate, the distribution and sales channel opportunities created by the Internet often enable businesses to tap into that market successfully.

Word Frequency vs. Resolving Power

The most frequent words are not the most descriptive.

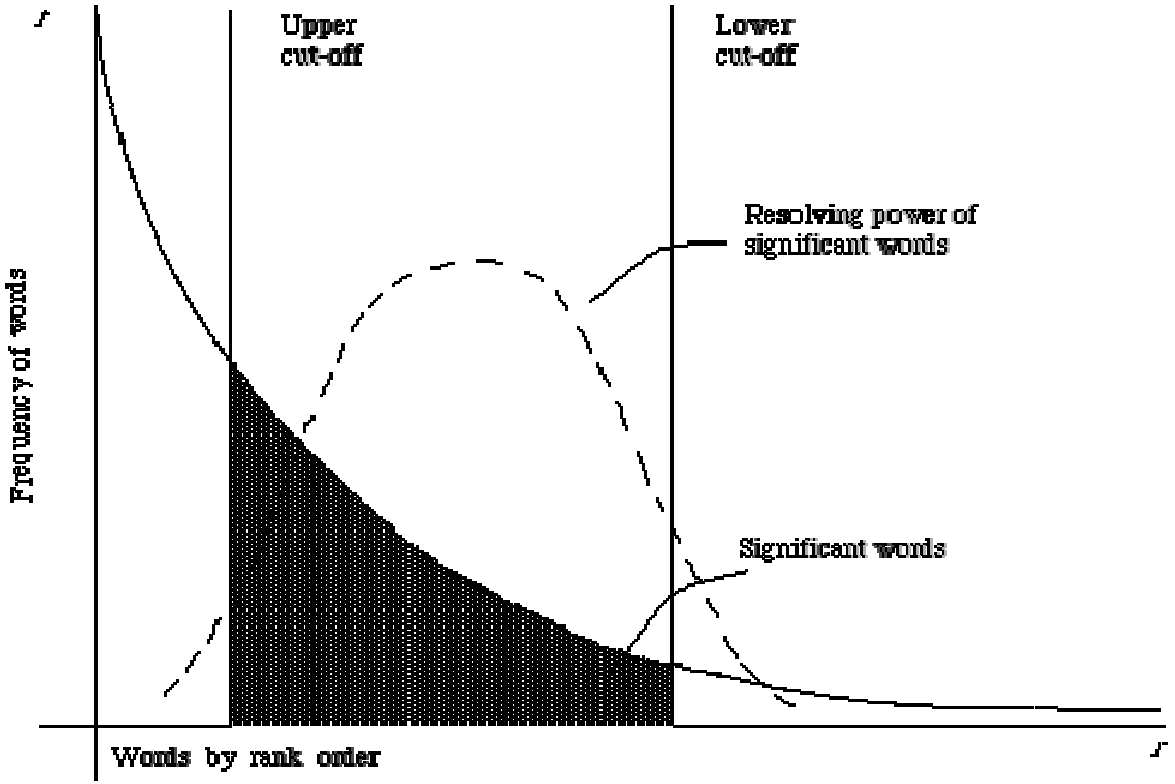


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[49] page 120)

van Rijsbergen 79

Consequences of Zipf for IR

- There are always a few very frequent tokens that are **not** good discriminators.
  - Called "stop words" in IR
  - Usually correspond to linguistic notion of "closed-class" words
    - English examples: *to, from, on, and, the, ...*
    - Grammatical classes that don't take on new members.
- There are always a large number of tokens that occur once and can mess up algorithms.
- Medium frequency words most descriptive

Text

- Perform lexical analysis - processing text into tokens
  - Many issues: normalization, lemmatization
- Stemming reduces the number of tokens
  - Porter stemmer most common
- Stop words removed to improve performance
- What remains are terms to be indexed
- Text has power law distribution
  - Words with resolving power in the middle and tail of the distribution