COEN 169

Basic Concepts

Yi Fang

Department of Computer Engineering

Santa Clara University

Basic Concepts: Outline

Basic concepts in information retrieval/search engines:

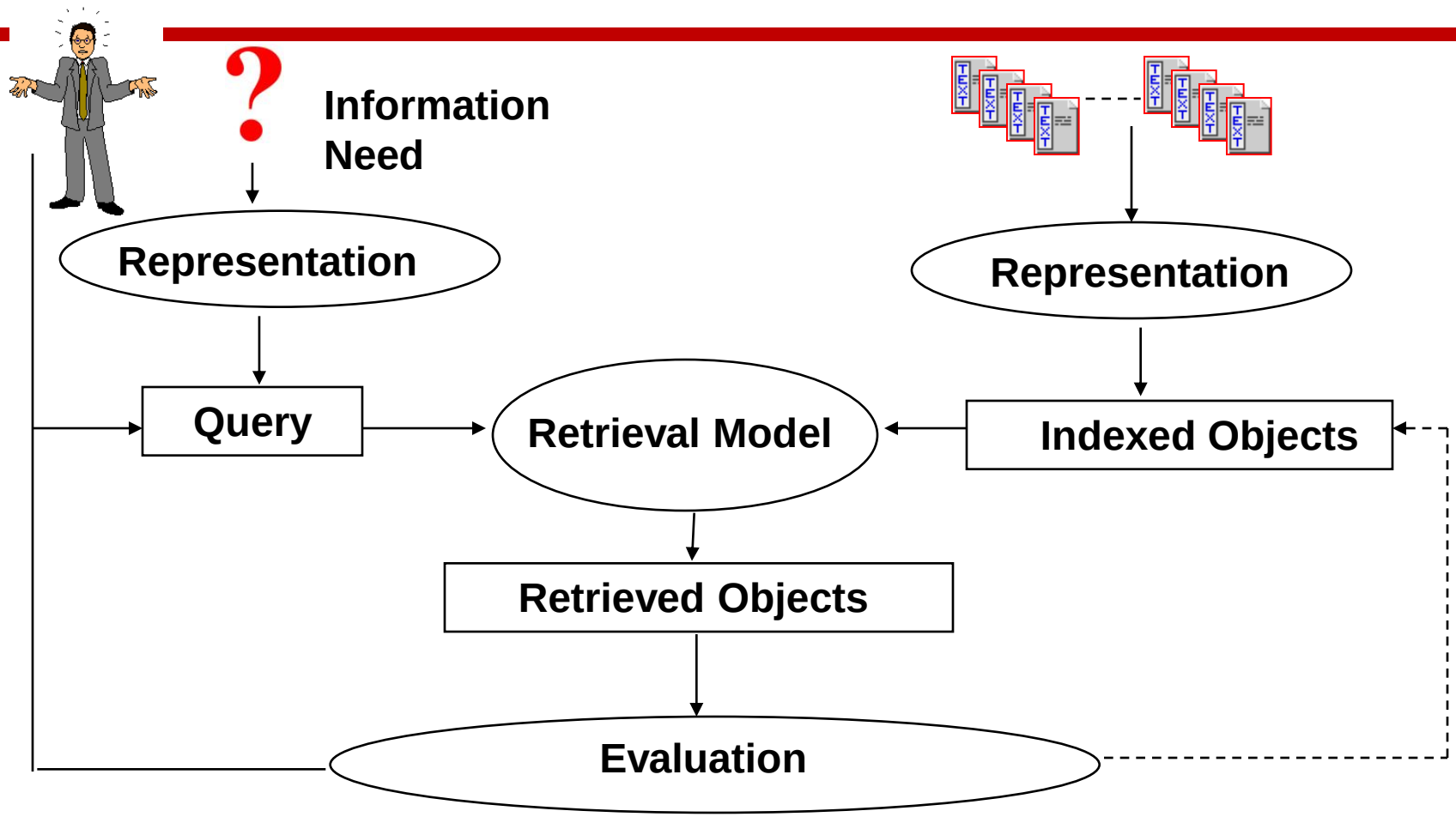
- Task definition
  - Terminologies and concepts
  - Overview of retrieval models
- Text representation
  - Text preprocessing
  - Indexing
- Evaluation
  - Evaluation methodology
  - Evaluation metrics

Terminologies

Terminologies:

- Query
  - Representative data of user's information need: text (default) and other media
- Document
  - Data candidate to satisfy user's information need: text (default) and other media
- Corpus
  - A set of documents
  - For web search, it is the whole Web
  - Valuable corpora from TREC (Text Retrieval Evaluation Conference)

Basic Process



Overview of Retrieval Models

Retrieval Models

- Boolean
- Vector space
  - Basic vector space — LUCENE
  - Extended Boolean
- Probabilistic models
  - Statistical language models — Lemur
  - Two Poisson model — Okapi
  - Bayesian inference networks — Indri
- Citation/Link analysis models
  - PageRank — Google
  - Hub & authorities — IBM

Overview of Retrieval Models

Retrieval Model

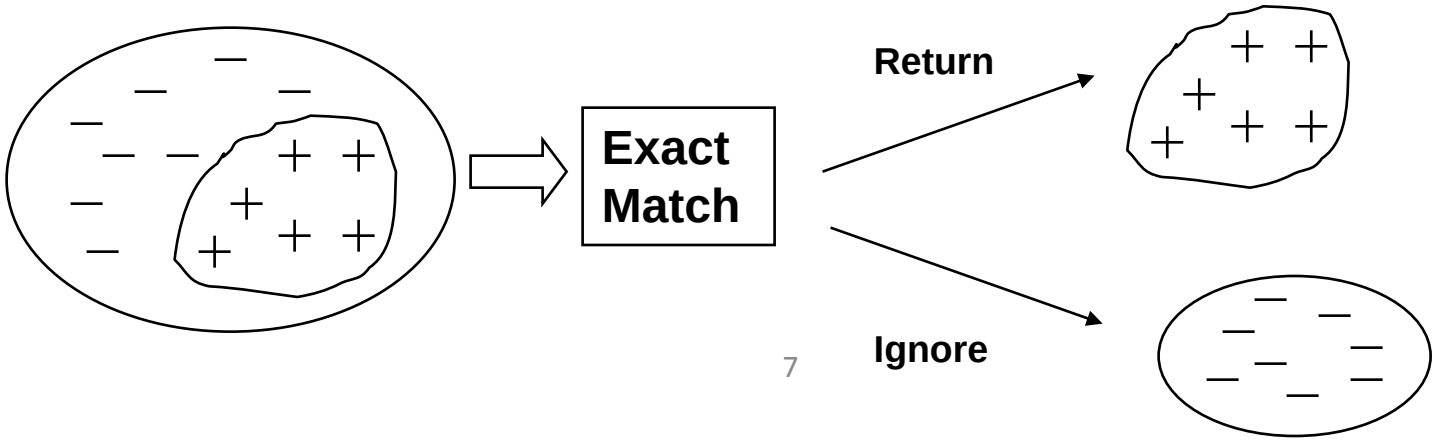
Determine whether a document is relevant to query

- Relevance is difficult to define
  - Varies by judgers
  - Varies by context (i.e., jointly by a set of documents and queries)
- Different retrieval methods estimate relevance differently
  - Word occurrence of document and query
  - In probabilistic framework, P(query|document) or P(Relevance|query,document)
  - Estimate semantic consistency between query and document

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression
- Boolean queries use *AND, OR* and *NOT* to join query terms

  e.g., *apple* AND *orange*
- Relevance is a binary variable; a document is either relevant (i.e., match query) or irrelevant (i.e., mismatch)

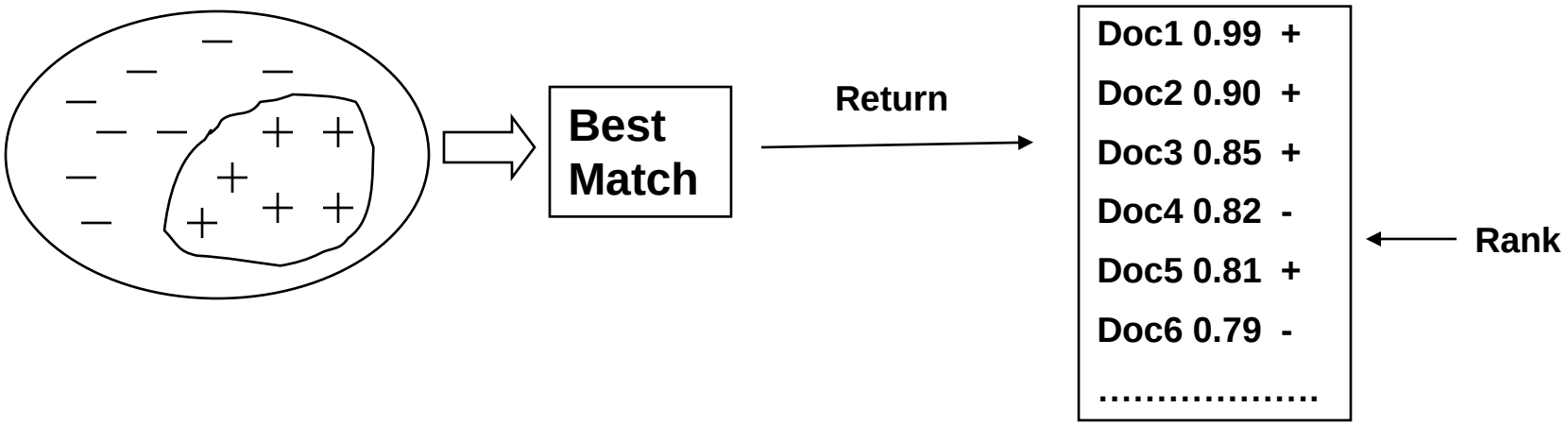


Boolean Retrieval Method

- Perhaps the simplest model to build an IR system
- Primary commercial retrieval tool for 3 decades
- Many search systems you still use are Boolean:
  Email, library catalog, legal retrieval, Mac OS X Spotlight

Types of Retrieval Models

- Best Match (Document Ranking)
  - Example: Most probabilistic models
  - Query describes the desired retrieval criterion
  - Degree of relevance is a continuous/integral variable; each document matches query to some degree
  - Result in a ranked list (top ones match better)
    - ✓ Often return a partial list (e.g., rank threshold)



Types of Retrieval Models

Exact Match (Selection) vs. Best Match (Ranking)

- Best Match is usually more accurate/effective
  - Do not need precise query; representative query generates good results
  - Users have control to explore the rank list: view more if need every piece; view less if need one or two most relevant
- Exact Match
  - Hard to define the precise query; too strict (terms are too specific) or too coarse (terms are too general)
  - Users have no control over the returned results
  - Still prevalent in some markets

Combining Various Retrieval Models

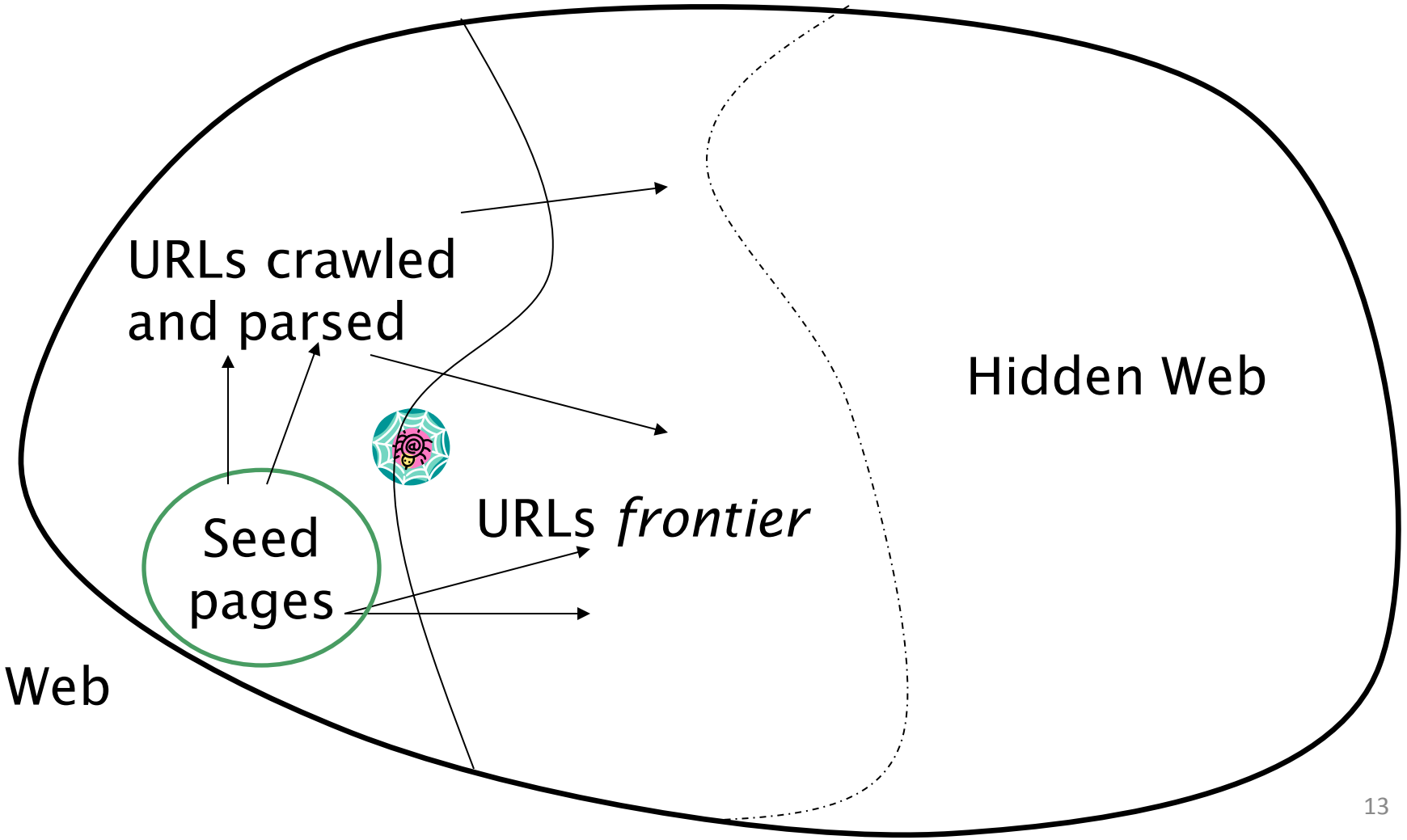
- In modern search engines, we often combine a large number of ranking signals
- Manually assigning weights to different models is usually difficult
- Machine Learning can help (an automatic approach)

Web crawler/spider

Crawler systematically browses the Web and collects the Web pages

- Begin with known “seed” URLs
- Fetch and parse them
  - Extract URLs they point to
  - Place the extracted URLs on a queue
- Fetch each URL on the queue and repeat

Crawling picture



Text Preprocessing

Text Preprocessing: extract representative index terms

- Tokenization
  - For most western languages, words separated by spaces; deal with punctuation, capitalization, hyphenation
  - For Chinese, Japanese: more complex word segmentation…
- Remove stopwords: (remove “the”, “is”,..., existing standard list)
- Morphological analysis (e.g., stemming):
  - Stemming: determine stem form of given inflected forms
- Other: extract phrases

Tokenization

- Analyze text into a sequence of discrete tokens (words)
- Deal with punctuation
- Case folding: reduce all letters in lower case
  - *Is Google doing case folding?*
- Simplest approach: ignore all numbers and punctuation and use only case-insensitive unbroken strings of alphabetic characters as tokens

Stopwords

- It is typical to *exclude* high-frequency words (e.g. function words: “a”, “the”, “in”, “to”; pronouns: “I”, “he”, “she”, “it”)
- Saving Space
- Speeding Searches

Stopwords list

- For efficiency, store strings for stopwords in a hashtable to recognize them in constant time
- Stopwords are language and application dependent

Trends in Stopword Removal

- The earliest systems used stopword lists of 200-300 terms
- To improve efficiency and effectiveness
- Very frequent terms were problematic for early retrieval models (e.g, OR operations in boolean retrieval)
- Web search engines generally do not remove stopwords
- The latest trend is to index stopwords and (possibly) ignore them at query-time if they seem unimportant
- Newer retrieval models are better at handling very frequent terms

Text Representation: Text Preprocessing

4 the	1 at	1 different	1 may	1 step
3 and	1 basal	1 exchange	1 nontarget	1 substance
3 by	1 be	1 exogenous	1 not	1 suggests
3 steroids	1 been	1 fluorescent	1 may	1 target
2 centrioles	1 bodies	1 from	1 of	1 technique
2 in	1 can	1 growth	1 precise	1 two
1 affect	1 at	1 has	1 receptor	1 unexpected
1 already	1 cell	1 identity	1 regularly	1 vitally
1 Although	1 cells	1 level	1 reveal	1 way
1 antibodies	1 cilia-bearing	1 localization	1 Specific	1 with

24 stopwords out of total 61 words

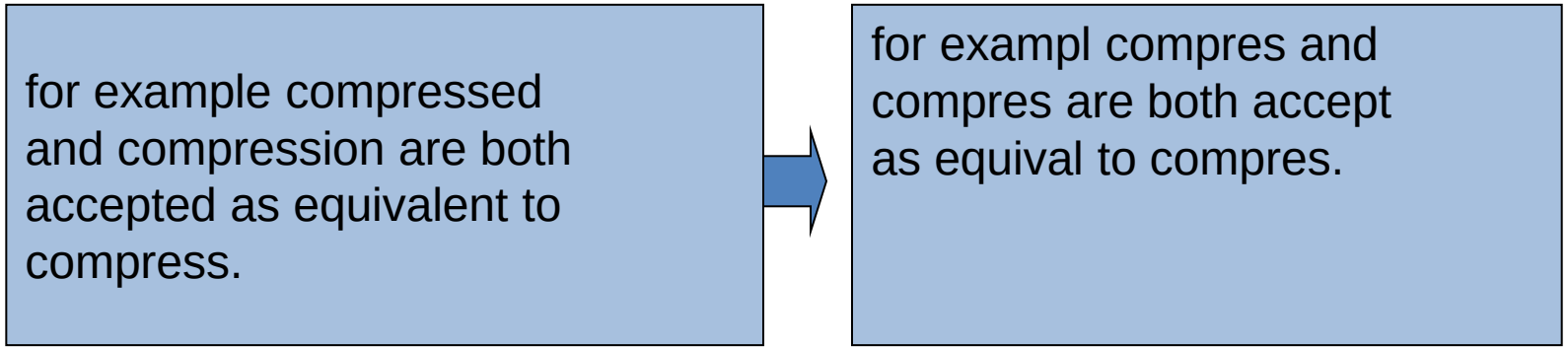
Lemmatization

- Reduce inflectional/variant forms to base form
- Direct impact on VOCABULARY size
- 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*

- How to do this?
  - Need a list of grammatical rules + a list of irregular words
  - Children → child, spoken → speak …

Stemming

- Correct morphological analysis is language specific and can be complex.
- Reduce tokens to “root” form of words to recognize morphological variation.
  - “computer”, “computational”, “computation” all reduced to same token “comput”
- Stemming “blindly” strips off known affixes (prefixes and suffixes) in an iterative fashion.



Porter Stemmer

- Simple procedure for removing known affixes in English without using a dictionary and complex grammar rules.
- Can produce unusual stems that are not English words:
  - “computer”, “computational”, “computation” all reduced to same token “comput”
- May conflate (reduce to the same token) words that are actually distinct.
- Not recognize all morphological derivations.

Typical rules in Porter

- *sses* $\rightarrow$ *ss*
- *ies* $\rightarrow$ *i*
- *ational* $\rightarrow$ *ate*
- *tional* $\rightarrow$ *tion*

Totally about 30 rules. Implementations of Porter stemmer:

http://tartarus.org/~martin/PorterStemmer/

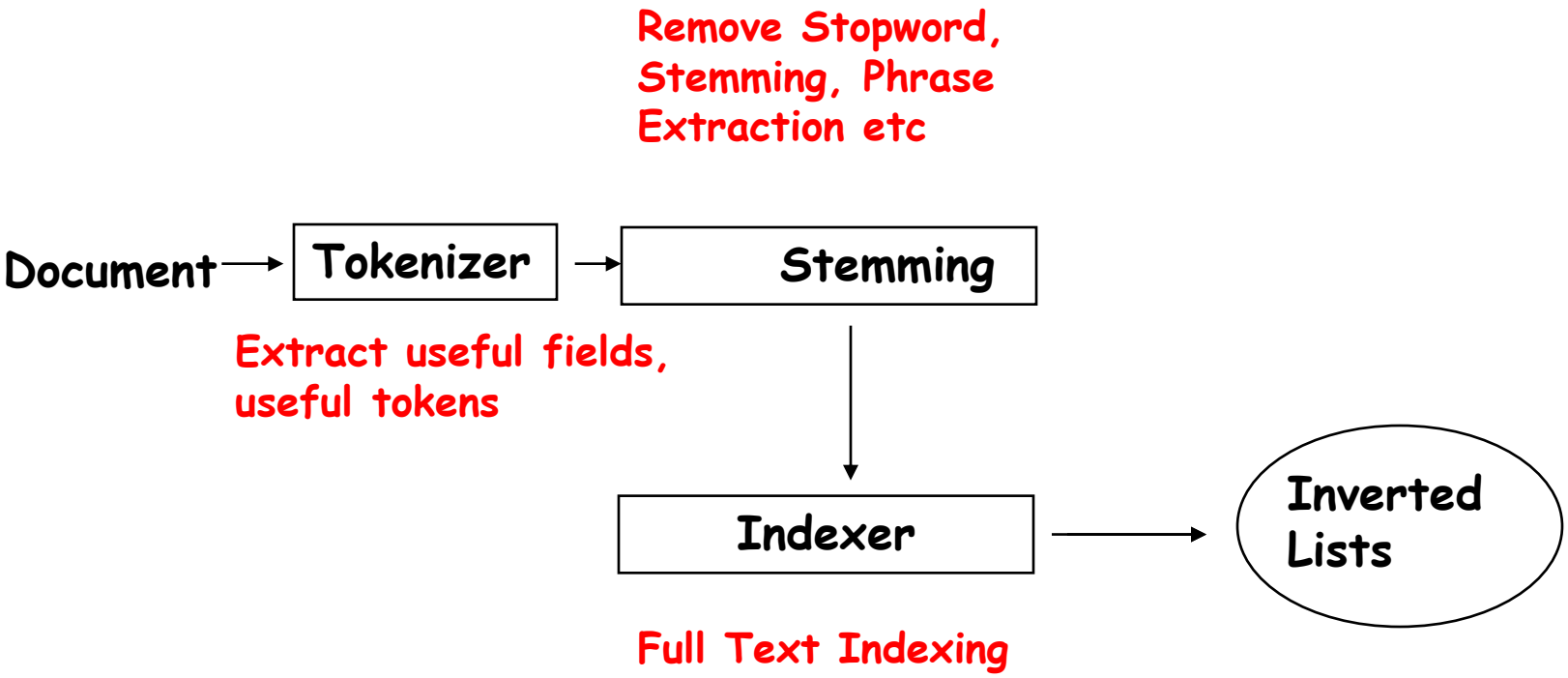
Porter Stemmer Errors

- Errors of “comission”:
  - organization, organ $\rightarrow$ organ
  - police, policy $\rightarrow$ polic
  - arm, army $\rightarrow$ arm
- Errors of “omission”:
  - cylinder, cylindrical
  - create, creation
  - Europe, European

Other stemmers

- Other stemmers exist, e.g., Lovins stemmer
  http://www.comp.lancs.ac.uk/computing/research/stemming/general/lovins.htm
- about 250 rules
- Motivated by Linguistics as well as IR
- Full morphological analysis - modest benefits for retrieval

Text Representation: Process of Indexing



Text Representation: Inverted Lists

Inverted lists are one of the most common indexing techniques

- Source file: collection organized by documents
- Inverted list file: collection organized by term

  one record per term, the lists of documents that contain the specific term

- Possible actions with inverted lists
  - OR: the union of lists
  - And: the intersection of lists

Text Representation: Inverted Lists

Doc ID	Text
1	kids question noting in 1960s
2	young man question everything in 1970s
3	kids question questions in 1980s
4	young man question nothing in 2000s

Documents

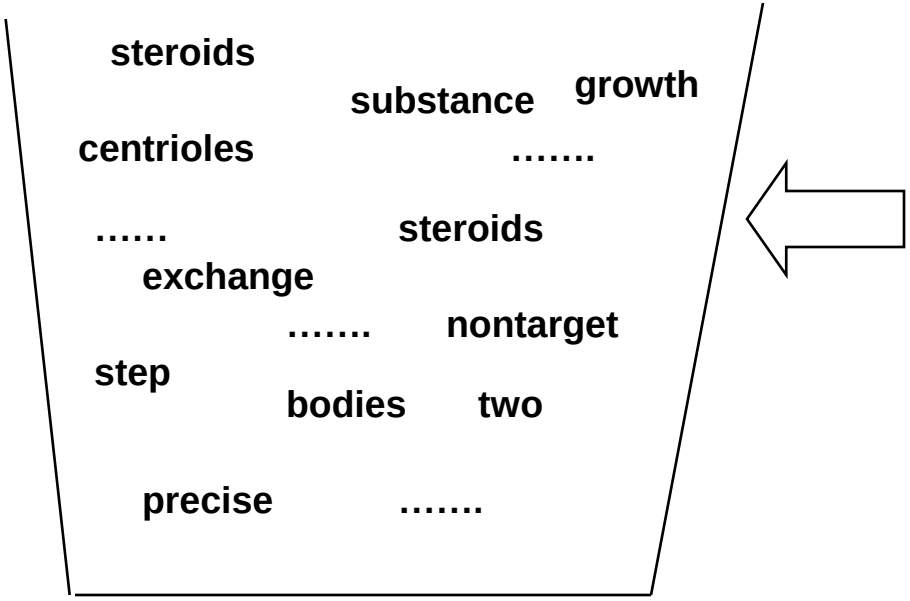
Term ID	Term	Documents
1	kids	1,3
2	question	1,2,3,4
3	nothing	1,4
4	in	1,2,3,4
5	19060s	1
6	young	2,4
7	man	2,4
8	everything	2
9	1970s	2
10	questions	3
11	1980s	3
11	2000s	4

Inverted Lists

Text Representation: Bag of Words

The simplest text representation: “bag of words”

- Query/document: a bag that contains words in it
- Order among words is ignored



3 steroids	1 cilia-bearing	1 precise	1 two
2 centrioles	1 different	1 receptor	1 unexpected
1 affect	1 exchange	1 regularly	1 vitally
1 already	1 exogenous	1 reveal	1 way
1 Although	1 fluorescent	1 Specific	
1 antibodies	1 growth	1 step	
1 basal	1 identity	1 substance	
1 bodies	1 level	1 suggests	
1 cell	1 localization	1 target	
1 cells	1 nontarget	1 technique	

Text Representation: Phrases

- Single word/stem indexing may not be sufficient

  e.g., “santa clara university”

- More complicated indexing includes phrases (thesaurus classes)

- How to automatically identify phrases
  - Dictionary
  - Find the most common N word phrases by corpus statistics
  - Syntactic analysis, noun phrases
  - More sophisticated segmentation algorithm like “Hidden Markov Model”

Stemming

Evaluation results (Hollink et al., 2004)

- English: 0-5% improvements
- Finnish: 30% improvement
- Spanish: 10% improvement

Google is doing stemming

My own suggestions

- Text-preprocessing is about making decisions on what to store in the index
- Three big decisions: tokenization, stopwords, and stemming
- Write the customized Tokenizer for your specific domain/application
- Remove stopwords from documents only if you have to (e.g., don't have enough disk-space)
- Remove stopwords from the query if they are not part of a phrase
- Stemming depending on the importance of recall and the size of the collection