22. Text Processing and Boolean Models

INFO 202 - 12 November 2008

Bob Glushko

Plan for Today's Class

Recall and Precision

Models for Information Retrieval

Text Processing Operations and Challenges

The Boolean Model

Overview of Remainder of Semester

We begin with introductory conceptual and technical foundations for IR

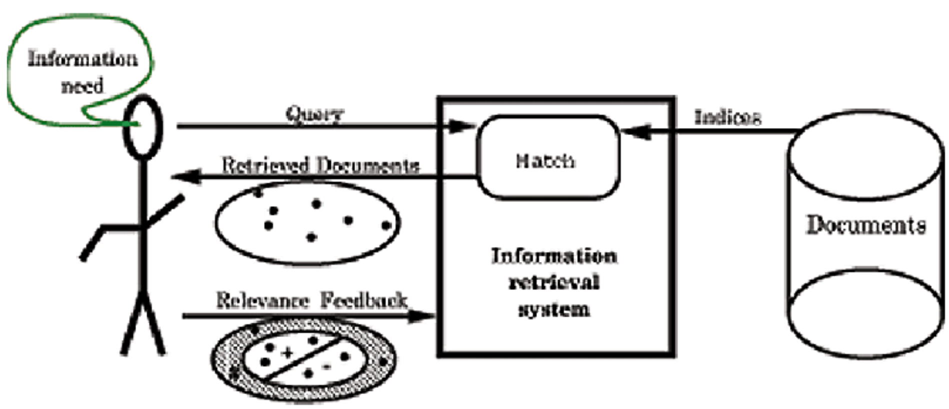
The issues and models get progressively more complicated for the next 5 lectures

On the Monday after Thanksgiving we have a lecture on multimedia retrieval (with lots of demos), followed by a lecture on applications of IR and natural language processing

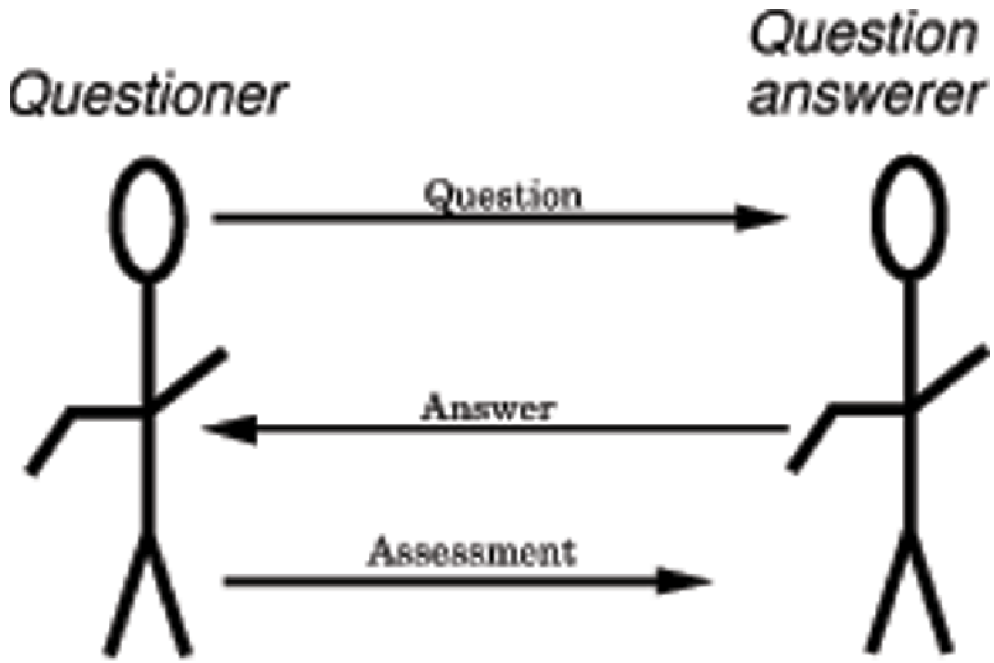
Last new material is "alumni day" (December 8) when some former students talk about their jobs, which emphasize IO and IR

The last class meeting of the semester (December 10) is a course review to prepare you for a three-hour final exam on December 15

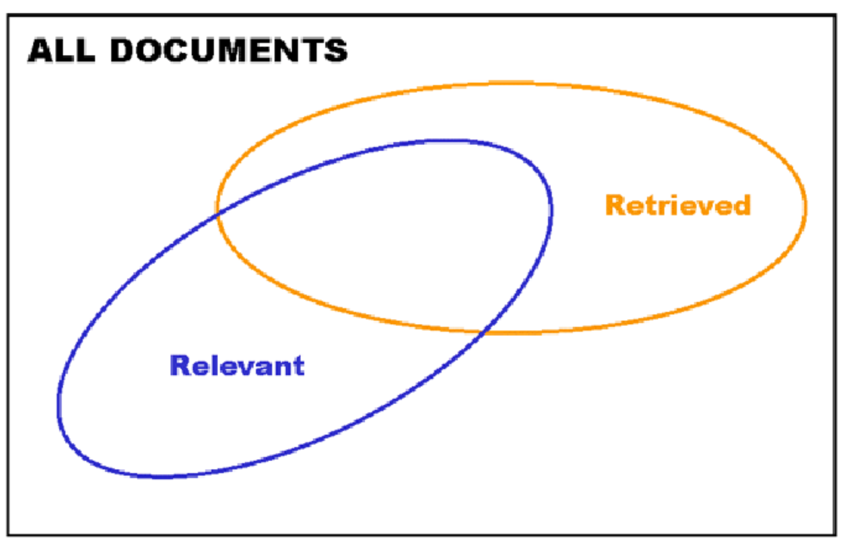
Schematic View of Classical Search



IR Only Approximates "Finding Out About"



Recall and Precision



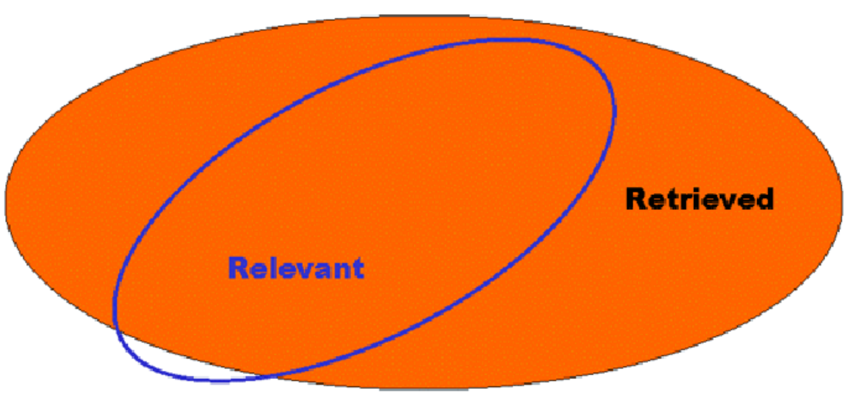
Recall and Precision [2]

RECALL is the proportion of the relevant documents that are retrieved

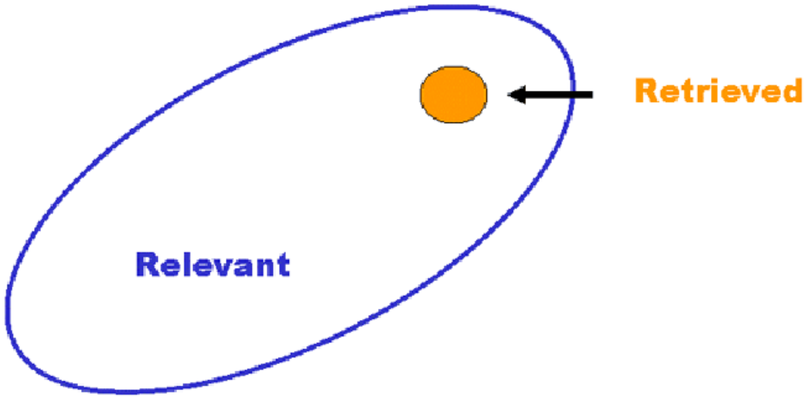
PRECISION is the proportion of the retrieved documents that are relevant

Goal: High recall and precision - Get as much good stuff as possible while getting as little junk as possible

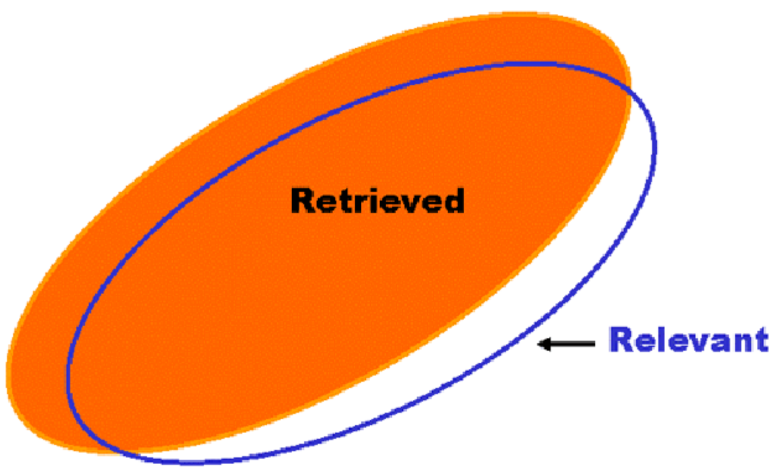
High Recall but Low Precision



Low Recall but High Precision



High Recall and High Precision



Models of Information Retrieval [1]

The core problems of information retrieval are finding relevant documents and ordering the found documents according to relevance

The IR model explains how these problems are solved:

- ...By specifying the representations of queries and documents in the collection being searched
- ...And the information used, and the calculations performed, that order the retrieved documents by relevance
- (And optionally, the model provides mechanisms for using relevance feedback to improve precision and results ordering)

Models of Information Retrieval [2]

BOOLEAN model -- representations are sets of index terms, set theory operations with Boolean algebra calculate relevance as binary

VECTOR models -- representations are vectors with non-binary weighted index terms, linear algebra operations yield continuous measure of relevance

Models of Information Retrieval [3]

STRUCTURE models -- combine representations of terms with information about structures within documents (i.e., hierarchical organization) and between documents (i.e. hypertext links and other explicit relationships) to determine which parts of documents and which documents are most important and relevant

PROBABILISTIC models -- documents are represented by index terms, and the key assumption is that the terms are distributed differently in relevant and non relevant documents.

What is a "Document" in Information Retrieval?

A document is any individually retrievable item in the "pile of text" that makes up the COLLECTION

Sometimes the boundaries that define documents are obvious or conventional (web search returns a web page), but sometimes they aren't

"Carving up" or "chunking" large documents into smaller text passages may be required for some collections or some user interfaces

A collection might contain any number of documents; web search engines index billions of pages

What is a Query?

A query is the expression of a user’s information needs and can take many forms:

A natural language description of the need

An artificial and restricted language

- Restrictions on the vocabulary limit the words that can be used in queries
- Restrictions on syntax limit the ways words can be combined in logical expressions
- These restrictions mean that queries may be unable to express the information need completely or accurately

The user interface(s) to the IR system influence the kinds of queries that the user can express (or express easily)

Text Processing: Motivation

Not all words are equally useful indicators of what a document is about

Nouns and noun groups carry more "aboutness" than adjectives, adverbs, and verbs

Very frequent words that occur in all or most documents add NOISE because they cannot discriminate between documents

So it is worthwhile to pre-process the text of documents to select a smaller set of terms that better represent them; these are called the INDEX terms

Text Processing: Operational Overview

1. DECODING -- extracting the text to be processed from its stored representation
2. FILTERING -- creating a stream of characters by removing formatting or non-semantic markup
3. TOKENIZATION -- segmenting the character stream into linguistic units
4. STOPWORD ELIMINATION -- remove words that poorly discriminate between documents
5. STEMMING -- removing affixes and suffixes to allow the retrieval of syntactic and morphological variations of query terms
6. SELECTING INDEX TERMS -- choosing word/stems (or groups of them) as indexing elements
7. CREATING AUXILIARY STRUCTURES -- like a THESAURUS

Decoding

The sequence of characters in a stored document might be represented in any number of single- or multi-byte encoding schemes

Determining this encoding can be easy (file extensions or metadata) -- but not always

Text encoding specs are well-documented
but "commercial products can easily live or die by the range of encodings they support"

Guess That Encoding [1]

Guess That Encoding [2]

\par {\listtext\pard\plain\hich\af0\dbch\af0\loch\f0 1.\tab}}\pard \ql \fi-360\li720\ri0\widctlpar\jclisttab\tx720\aspalpha\aspnum\faauto\ls4\adjustright\rin0\lin720\itap0 {Degree products. The university currently offers a relatively fixed set of degree \'93 products.\'94 If the university adopted Dell\rquote s \'93build to order\'94 strategy how might its product offerings change? How might this new strategy affect the university\rquote s \'93market share\'94 or \'93profitability\'94 compared with its current strategy?

\par }\pard \ql \li0\ri0\widctlpar\aspalpha\aspnum\faauto\adjustright\rin0\lin0\itap0 {

\par Numerous documents are currently involved in the end-to-

Guess That Encoding [3]

```
<Party>
<Name First="Arnold" Last="Schwarzenegger"/>
<Address>
<StreetAddress>Governor's Mansion, 1526 H Street</StreetAddress>
<City>Sacramento</City>
<State>California</State>
<PostalCode Route="1234">95814</PostalCode>
</Address>
<Phone>
<AreaCode>916</AreaCode>
<LocalNumber>323-3047</LocalNumber>
</Phone>
</Party>
```

Filtering

Removing surrounding header or format information from the text to be processed

What you filter depends on the encoding format or document type

- You'd probably discard HTML markup before indexing
- You'd almost certainly save XML tags for indexing
- You'd probably want to use the rich metadata in email mail headers

Sentence Segmentation

Many IR and text processing applications require that the documents be broken into their constituent sentences

Punctuation marks like -- . , ! ? " -- can make this easy; but not always: sometimes you'll say "Dr. Glushko, this is too hard."

But abbreviations (Dr.) break the obvious rule, and even more complex rules like "period-space-capital letter" signals a sentence break still makes a lot of mistakes

Tokenization into "Wordlike" Elements

Another problem that seems trivial -- just use white space, right?

But what about:

- abbreviations (Dr. is a word)
- hyphens (sometimes part of a word, but sometimes a result of formatting)
- case (do we distinguish Bank from bank)

Tokenization Challenges [1]

Character sequences where the tokens include complex alphanumeric structure or punctuation syntax:

- glushko@ischool.berkeley.edu
- 10/26/53
- October 26, 1953
- 55 B.C
- B-52
- 128.32.226.140
- My PGP key is 324a3df234ch23e

Tokenization Challenges [2]

Mr. O'Neill thinks that the boys' stories about Chile's capital aren't amusing.

For *O'Neill*, which of the following is the desired tokenization?

And for *aren't*, is it:

Tokenization Challenges [3]

The language that the characters represent needs to be identified during decoding because it influences the order and nature of tokenization

In languages that are written right-to-left like Arabic and Hebrew, left-to-right text can be interspersed, like numbers and dollar amounts

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

← → ← → ← START

'Algeria achieved its independence in 1962 after 132 years of French occupation.'

In German compound nouns don't have spaces between the tokens

- Lebensversicherungsgesellschaftsangestellter = "life insurance company employee"

Tokenization in "Non-Segmented" Languages

And these problems in "segmented languages" that use white space and punctuation to delimit words seem trivial compared to problems tokenizing Oriental languages that are "non-segmented"

These languages have ideographic characters that can appear as one-character words but they also can combine to create new words.

The analogous problem in English would be the word "TOGETHER" -- do we treat it as one word or is three separate words "TO GET HER"

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。

Stop or Noise Words

Any word that doesn't convey meaning by itself can't help us "find out about" anything so it can be discarded during text processing

In English these STOP or NOISE words include:

- determiners, such as "the" and "a(n)"
- auxiliaries, such as "might," "have," and "be"
- conjunctions, such as "and," "that," and "whether"
- degree adverbs, such as "very" and "too"

These words are always among the most frequent in a collection, but high frequency alone isn't what makes them bad index terms

So stop or noise words are usually not determined by frequency analysis -- text processors usually employ a list of them as a kind of negative dictionary

But Stop Words Should be Kept for Phrase Indexing

"President of the United States" is a more precise query than "President" AND "United States"

"To be or not to be"

"Let it Be"

"Flights to London" and "Flights from London" aren't the same query

"Laser printer toner cartridge" vs "Laser printer, with toner cartridge"

One Minute Morphology

MORPHOLOGY is the part of linguistics concerned with the mechanisms by which natural languages create words and word forms from smaller units

These basic building blocks are called MORPHEMES and can express semantic concepts (when they are called ROOTS or abstract features like "pastness" or "plural")

Every natural language contains about 10,000 morphemes and because of how they combine to create words, the number of words is an order of magnitude greater

Inflection and Derivation

INFLECTION is the morphological mechanism that changes the form of a word to handle tense, aspect, agreement, etc. It never changes the part-of-speech (grammatical category)

- dog, dogs
- tengo, tienes, tenemos, tienen

DERIVATION is the mechanism for creating new words, usually of a different part-of-speech category, by adding a BOUND MORPH to a BASE MORPH

- build + ing -> building; health + y -> healthy

Morphological Processing

Morphological analysis of a language is often used in information retrieval and other low-level text processing applications (hyphenation, spelling correction) because solving problems using root forms and rules is more scaleable and robust than solving them using word lists

- Natural languages are generative, with new words continually being invented
- Many misspellings of common words are obscure low frequency words, so adding them to a misspelling list would make it impossible to check spellings for the latter

Stemming

STEMMING is morphological processing to remove prefixes and suffixes to leave the root form of words

Stemming reduces many related words and word forms to a common canonical form

This makes it possible to retrieve documents when they contain the meaning we're looking for even if the form of the search word doesn't exactly match what's in the documents

In English, inflectional morphology is relatively easy to handle and "dumb" stemmers (e.g., iteratively remove suffixes, matching longest sequence in rewrite rule) perform acceptably

Derivational morphology is more difficult

Stemming Mistakes

Stemming affects the recall/precision tradeoff

OVERSTEMMING results when stemming is so aggressive that it reduces words that are not morphologically related to the same root

- Organization, organ
- Policy, police
- Arm, army

UNDERSTEMMING results when stemming is too timid and some morphologically related words are not reduced to the same root

- acquire, acquiring, acquired -> acquir
- acquisition -> acquis

Selecting Index Terms

At this stage in text processing the text collection is represented as a set of stems

But not all of them will help a searcher find what they're looking for because they will retrieve too many or too few documents

We can select better index terms if we analyze the distribution of words / stems in the collection

- We can eliminate some terms entirely
- We will treat some terms as more important than others in indicating what a document is about

The Index -- Logical View

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Anthony	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	
...							

An index is a data structure that records information about the occurrences of terms in documents

This is a term-document matrix -- rows for terms, columns for documents -- one such data structure

The "Inverted" Index

Using a term-document matrix index representation is both infeasible and nonsensical for any substantial collection of documents

So instead we divide the index into two parts

- A DICTIONARY is a list of the terms
- A POSTINGS LIST is the list of documents in which each term occurs (usually with frequency and position information within each document)

Brutus → 1 | 2 | 4 | 11 | 31 | 45 | 173 | 174

Caesar → 1 | 2 | 4 | 5 | 6 | 16 | 57 | ...

Calpurnia → 2 | 31 | 54 | 101

Indexing Step 1 - Term List

Doc 1

Now is the time for all good men to come to the aid of their country

Doc 2

It was a dark and stormy night in the country manor. The time was past midnight

Term	Doc #
now	1
is	1
the	1
time	1
for	1
all	1
good	1
men	1
to	1
come	1
to	1
the	1
aid	1
of	1
their	1
country	1
it	2
was	2
a	2
dark	2
and	2
stormy	2
night	2
in	2
the	2
country	2
manor	2
the	2
time	2
was	2
past	2
midnight	2

Step 2 -- Alphabetize and Merge

Term	Doc #
a	2
aid	1
all	1
and	2
come	1
country	1
country	2
dark	2
for	1
good	1
in	2
is	1
it	2
manor	2
men	1
midnight	2
night	2
now	1
of	1
past	2
stormy	2
the	1
the	1
the	2
the	2
their	1
time	1
time	2
to	1
to	1
was	2
was	2

➔

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

Step 3 -- Separate Dictionary and Postings

Term	Doc #	Freq
a	2	1
aid	1	1
all	1	1
and	2	1
come	1	1
country	1	1
country	2	1
dark	2	1
for	1	1
good	1	1
in	2	1
is	1	1
it	2	1
manor	2	1
men	1	1
midnight	2	1
night	2	1
now	1	1
of	1	1
past	2	1
stormy	2	1
the	1	2
the	2	2
their	1	1
time	1	1
time	2	1
to	1	2
was	2	2

Dictionary

Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
it	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stormy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

Postings

Doc #	Freq
2	1
1	1
1	1
2	1
1	1
1	1
2	1
2	1
1	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
1	1
2	1
2	1
1	2
2	2
1	1
1	1
2	1
1	2
2	2

Boolean Queries

The simplest query language to implement is a Boolean one because it has a very direct correspondence to the text processing story and indexing story we just told

Boolean queries dominate commercial IR systems (and are implemented but rarely used for web searches)

Boolean queries are expressed as Terms + Operators

- Terms are words or stemmed words
- Operators are AND, OR, NOT

Boolean Expressions

Usually expressed with INFIX operators:

- ((a AND b) OR (c AND b))

NOT is UNARY PREFIX operator:

- ((a AND b) OR (c AND (NOT b)))

AND and OR can be n-ary operators:

- (a AND b AND c AND d)

DeMorgan's Law:

- NOT(a) AND NOT(b) = NOT(a OR b)
- NOT(a) OR NOT(b)= NOT(a AND b)

Sample Boolean Queries

Cat

Cat OR Dog

Cat AND Dog

(Cat AND Dog) OR Collar

(Cat AND Dog) OR (Collar AND Leash)

(Cat OR Dog) AND (Collar OR Leash)

Interpreting Boolean Queries

(Cat OR Dog) AND (Collar OR Leash)

Doc #	1	2	3	4	5	6	7
CAT	X	X	X		X	X	
DOG		X	X	X		X	X
COLLAR	X		X	X		X	X
LEASH		X	X		X		X

Doc #	1	2	3	4	5	6	7
CAT		X		X			
DOG	X			X			
COLLAR			X		X		
LEASH			X			X	

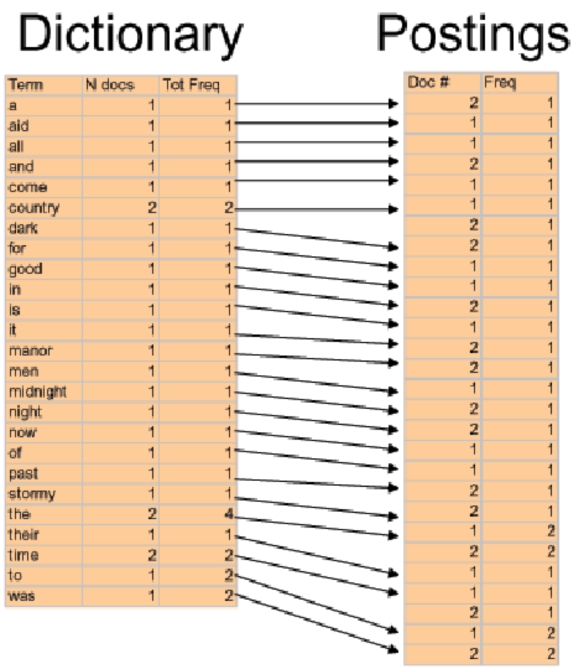
Boolean Search with Inverted Indexes [1]

Permit fast search for individual terms

For each term, you get a list consisting of:

- Document ID
- Frequency of term in doc (optional)
- Position of term in doc (optional)

Boolean Search with Inverted Indexes [2]



Term	N docs	Tot Freq
a	1	1
aid	1	1
all	1	1
and	1	1
come	1	1
country	2	2
dark	1	1
for	1	1
good	1	1
in	1	1
is	1	1
it	1	1
manor	1	1
men	1	1
midnight	1	1
night	1	1
now	1	1
of	1	1
past	1	1
stormy	1	1
the	2	4
their	1	1
time	2	2
to	1	2
was	1	2

Doc #	Freq
2	1
1	1
1	1
2	1
1	1
1	1
2	1
2	1
1	1
1	1
2	1
1	1
2	1
2	1
1	1
2	1
2	1
1	1
1	1
2	1
2	1
1	2
2	2
1	1
1	1
2	1
1	2
2	2

QUERY:
“time AND dark”

2 documents with “time” in dictionary have DOC#1, DOC#2 in posting file

1 document with “dark” in dictionary has DOC#2 in posting file

SOLUTION:
DOC#2 satisfies query

Readings for Lecture #23 (11/17)

Manning: Chapter 7: Vector Space Retrieval