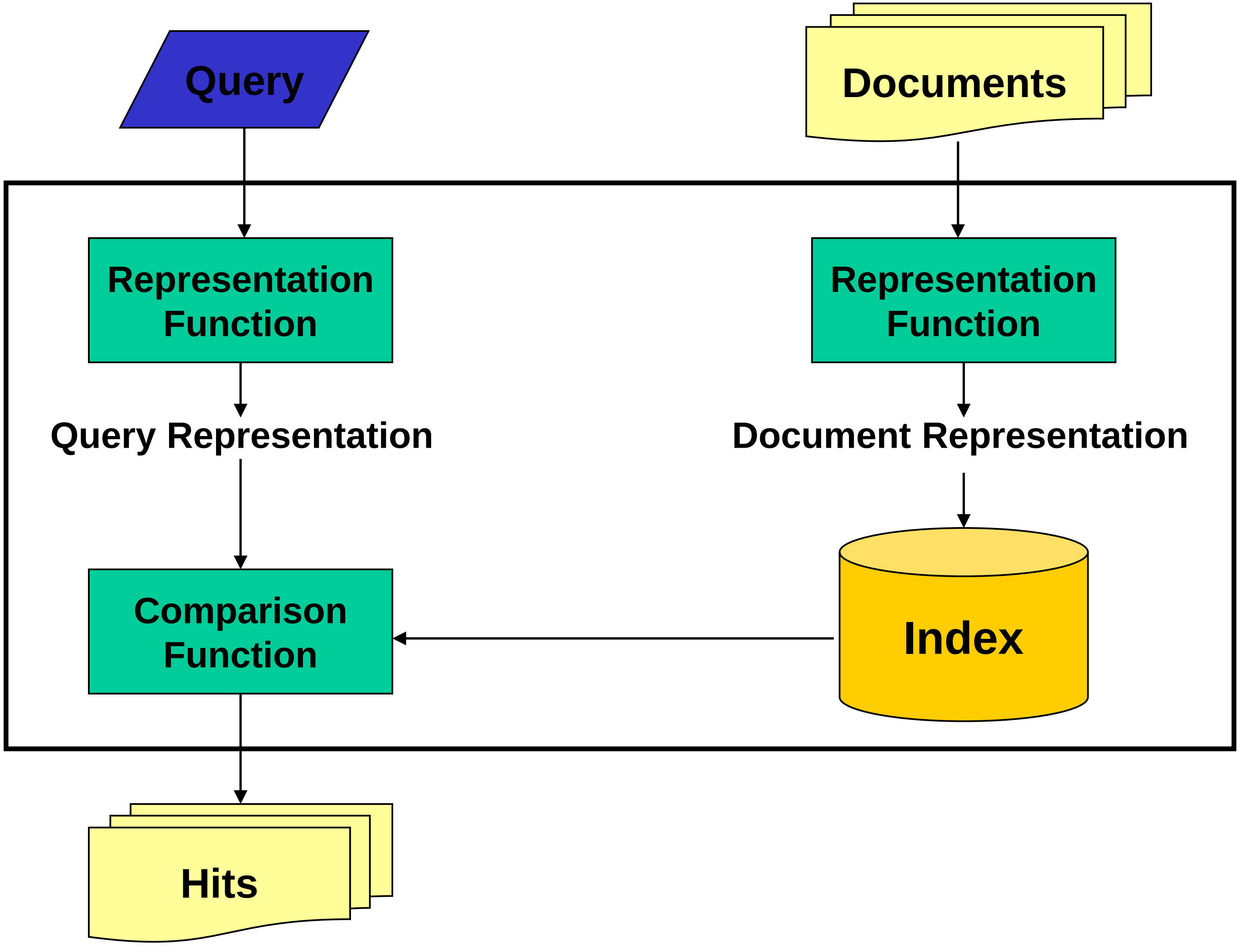
Evidence from Content

LBSC 796/INFM 718R

Session 2

February 9, 2011

Where Representation Fits



Agenda

- ➢ Character sets
- Terms as units of meaning
- Building an index
- MapReduce
- Project Overview

The character ‘A’

- ASCII encoding: 7 bits used per character

  0 1 0 0 0 0 0 1 = 65 (decimal)

  0 1 0 0 0 0 0 1 = 41 (hexadecimal)

  0 1 0 0 0 0 0 1 = 101 (octal)

- Number of representable character codes:

  $2^7 = 128$

- Some codes are used as “control characters”

  e.g. 7 (decimal) rings a “bell” (these days, a beep) (“^G”)

ASCII

- Widely used in the U.S.
  - American Standard Code for Information Interchange
  - ANSI X3.4-1968

0	NUL	32	SPACE	64	@	96	`
1	SOH	33	!	65	A	97	a
2	STX	34	"	66	B	98	b
3	ETX	35	#	67	C	99	c
4	EOT	36	$	68	D	100	d
5	ENQ	37	%	69	E	101	e
6	ACK	38	&	70	F	102	f
7	BEL	39	'	71	G	103	g
8	BS	40	(	72	H	104	h
9	HT	41	)	73	I	105	i
10	LF	42	*	74	J	106	j
11	VT	43	+	75	K	107	k
12	FF	44	,	76	L	108	l
13	CR	45	-	77	M	109	m
14	SO	46	.	78	N	110	n
15	SI	47	/	79	O	111	o
16	DLE	48	0	80	P	112	p
17	DC1	49	1	81	Q	113	q
18	DC2	50	2	82	R	114	r
19	DC3	51	3	83	S	115	s
20	DC4	52	4	84	T	116	t
21	NAK	53	5	85	U	117	u
22	SYN	54	6	86	V	118	v
23	ETB	55	7	87	W	119	w
24	CAN	56	8	88	X	120	x
25	EM	57	9	89	Y	121	y
26	SUB	58	:	90	Z	122	z
27	ESC	59	;	91	[	123	{
28	FS	60	<	92	\	124	|
29	GS	61	=	93	]	125	}
30	RS	62	>	94	^	126	~
31	US	64	?	95	_	127	DEL

Geeky Joke for the Day

- Why do computer geeks confuse Halloween and Christmas?

- ***Because 31 OCT = 25 DEC!***

- *031 OCT* $= 0*8^2 + 3*8^1 + 1*8^0$ *octal*

  $= 0*10^2 + 2*10^1 + 5*10^0$ *decimal*

The Latin-1 Character Set

- ISO 8859-1 8-bit characters for Western Europe
  - French, Spanish, Catalan, Galician, Basque, Portuguese, Italian, Albanian, Afrikaans, Dutch, German, Danish, Swedish, Norwegian, Finnish, Faroese, Icelandic, Irish, Scottish, and English

Printable Characters, 7-bit ASCII

	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	

Additional Defined Characters, ISO 8859-1

	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

Other ISO-8859 Character Sets

-2

	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	-	Ž	Ż
°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

-3

	Ą	˘	Ł	¤	Ľ	Ś	§	¨	Š	Ş	Ť	Ź	-	Ž	Ż
°	ą	˛	ł	´	ľ	ś	ˇ	¸	š	ş	ť	ź	˝	ž	ż
Ŕ	Á	Â	Ă	Ä	Ĺ	Ć	Ç	Č	É	Ę	Ë	Ě	Í	Î	Ď
Đ	Ń	Ň	Ó	Ô	Ő	Ö	×	Ř	Ů	Ú	Ű	Ü	Ý	Ţ	ß
ŕ	á	â	ă	ä	ĺ	ć	ç	č	é	ę	ë	ě	í	î	ď
đ	ń	ň	ó	ô	ő	ö	÷	ř	ů	ú	ű	ü	ý	ţ	˙

-4

	Ħ	˘	£	¤		Ĥ	§	¨	İ	Ş	Ğ	Ĵ	-		Ż
°	ħ	²	³	´	µ	ĥ	·	¸	ı	ş	ğ	ĵ	½		ż
À	Á	Â		Ä	Ċ	Ĉ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
	Ñ	Ò	Ó	Ô	Ġ	Ö	×	Ĝ	Ù	Ú	Û	Ü	Ŭ	Ŝ	ß
à	á	â		ä	ċ	ĉ	ç	è	é	ê	ë	ì	í	î	ï
	ñ	ò	ó	ô	ġ	ö	÷	ĝ	ù	ú	û	ü	ŭ	ŝ	˙

-5

	Ё	Ђ	Ѓ	Є	Ѕ	І	Ї	Ј	Љ	Њ	Ћ	Ќ	-	Ў	Џ
А	Б	В	Г	Д	Е	Ж	З	И	Й	К	Л	М	Н	О	П
Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я
а	б	в	г	д	е	ж	з	и	й	к	л	м	н	о	п
р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я
№	ё	ђ	ѓ	є	ѕ	і	ї	ј	љ	њ	ћ	ќ	§	ў	џ

-6

				¤								،	-		
											؛				؟
	ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د
ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ					
ـ	ف	ق	ك	ل	م	ن	ه	و	ى	ي	ً	ٌ	ٍ	َ	ُ
ِ	ّ	ْ													

-7

	‘	’	£			¦	§	¨	©		«	¬	-		―
°	±	²	³	΄	΅	Ά	·	Έ	Ή	Ί	»	Ό	½	Ύ	Ώ
ΐ	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
Π	Ρ		Σ	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	ά	έ	ή	ί
ΰ	α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
π	ρ	ς	σ	τ	υ	φ	χ	ψ	ω	ϊ	ϋ	ό	ύ	ώ	

-8

		¢	£	¤	¥	¦	§	¨	©	×	«	¬	-	®	‾
°	±	²	³	´	µ	¶	·	¸	¹	÷	»	¼	½	¾	
															‗
א	ב	ג	ד	ה	ו	ז	ח	ט	י	ך	כ	ל	ם	מ	ן
נ	ס	ע	ף	פ	ץ	צ	ק	ר	ש	ת					

-9

	¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
Ğ	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	İ	Ş	ß
à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
ğ	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ı	ş	ÿ

East Asian Character Sets

- More than 256 characters are needed
  - Two-byte encoding schemes (e.g., EUC) are used
- Several countries have unique character sets
  - GB in Peoples Republic of China, BIG5 in Taiwan, JIS in Japan, KS in Korea, TCVN in Vietnam
- Many characters appear in several languages
  - Research Libraries Group developed EACC
    - Unified “CJK” character set for USMARC records

Unicode

- Single code for all the world’s characters
  - ISO Standard 10646
- Separates “code space” from “encoding”
  - Code space extends Latin-1
    - The first 256 positions are identical
  - UTF-7 encoding will pass through email
    - Uses only the 64 printable ASCII characters
  - UTF-8 encoding is designed for disk file systems

Limitations of Unicode

- Produces larger files than Latin-1
- Fonts may be hard to obtain for some characters
- Some characters have multiple representations
  - e.g., accents can be part of a character or separate
- Some characters look identical when printed
  - But they come from unrelated languages
- Encoding does not define the “sort order”

Drawing it Together

- Key concepts
  - Character, Encoding, Font, Sort order
- Discussion question
  - How do you know what character set a document is written in?
  - What if a mixture of character sets was used?

Agenda

- Character sets
- ➢ Terms as units of meaning
- Building an index
- MapReduce
- Project overview

Strings and Segments

- Retrieval is (often) a search for concepts
  - But what we actually search are character strings

- What strings best represent concepts?
  - In English, words are often a good choice
    - Well-chosen phrases might also be helpful
  - In German, compounds may need to be split
    - Otherwise queries using constituent words would fail
  - In Chinese, word boundaries are not marked
    - Thissegmentationproblemissimilartothatofspeech

Tokenization

- Words (from linguistics):
  - Morphemes are the units of meaning
  - Combined to make words
    - Anti (disestablishmentarian) ism

- Tokens (from Computer Science)
  - Doug ’s running late !

Morphology

- Inflectional morphology
  - Preserves part of speech
  - Destructions = Destruction+PLURAL
  - Destroyed = Destroy+PAST

- Derivational morphology
  - Relates parts of speech
  - Destructor = AGENTIVE(destroy)

Stemming

- Conflates words, usually preserving meaning
  - Rule-based suffix-stripping helps for English
    - {destroy, destroyed, destruction}: *destr*
  - Prefix-stripping is needed in some languages
    - Arabic: {alselam}: *selam* [Root: SLM (peace)]

- Imperfect: goal is to <u>usually</u> be helpful
  - Overstemming
    - {centennial,century,center}: *cent*
  - Understamming:
    - {acquire,acquiring,acquired}: *acquir*
    - {acquisition}: *acquis*

Longest Substring Segmentation

- Greedy algorithm based on a lexicon
- Start with a list of every possible term
- For each unsegmented string
  - Remove the longest single substring in the list
  - Repeat until no substrings are found in the list
- Can be extended to explore alternatives

Longest Substring Example

- Possible German compound term:
  - washington
- List of German words:
  - ach, hin, hing, sei, ton, was, wasch
- Longest substring segmentation
  - was-hing-ton
  - Roughly translates as “What tone is attached?”

Probabilistic Segmentation

- For an input word $c_1 c_2 c_3 \ldots c_n$
- Try all possible partitions into $w_1 w_2 w_3 \ldots$
  - $c_1 \quad c_2 c_3 \ldots c_n$
  - $c_1 \quad c_2 c_3 \quad c_3 \ldots c_n$
  - $c_1 c_2 \quad c_3 \ldots c_n$

  etc.
- Choose the highest probability partition
  - E.g., compute $\Pr(w_1 w_2 w_3)$ using a language model
- Challenges: search, probability estimation

Non-Segmentation: N-gram Indexing

- Consider a Chinese document $c_1\, c_2\, c_3 \ldots c_n$

- Don’t segment (you could be wrong!)

- Instead, treat *every* character bigram as a term

  $c_1\, c_2\,,\ \ c_2\, c_3\,,\ \ c_3\, c_4\,, \ldots\ ,\ c_{n-1}\, c_n$

- Break up queries the same way

Relating Words and Concepts

- Homonymy: *bank* (river) vs. *bank* (financial)
  - **Different** words are written the same way
  - We'd like to work with word **senses** rather than words

- Polysemy: *fly* (pilot) vs. *fly* (passenger)
  - A word can have different "shades of meaning"
  - Not bad for IR: often helps more than it hurts

- Synonymy: *class* vs. *course*
  - Causes search failures … well address this next week!

Word Sense Disambiguation

- Context provides clues to word meaning
  - “The doctor removed the appendix.”
- For each occurrence, note surrounding words
  - e.g., +/- 5 non-stopwords
- Group similar contexts into clusters
  - Based on overlaps in the words that they contain
- Separate clusters represent different senses

Disambiguation Example

- Consider four example sentences
  - The doctor removed the appendix
  - The appendix was incomprehensible
  - The doctor examined the appendix
  - The appendix was removed
- What clues can you find from nearby words?
  - Can you find enough word senses this way?
  - Might you find too many word senses?
  - What will you do when you aren’t sure?

Why Disambiguation Hurts

- Disambiguation tries to reduce incorrect matches
  - But errors can also reduce correct matches
- Ranked retrieval techniques already disambiguate
  - When more query terms are present, documents rank higher
  - Essentially, queries give each term a context

Phrases

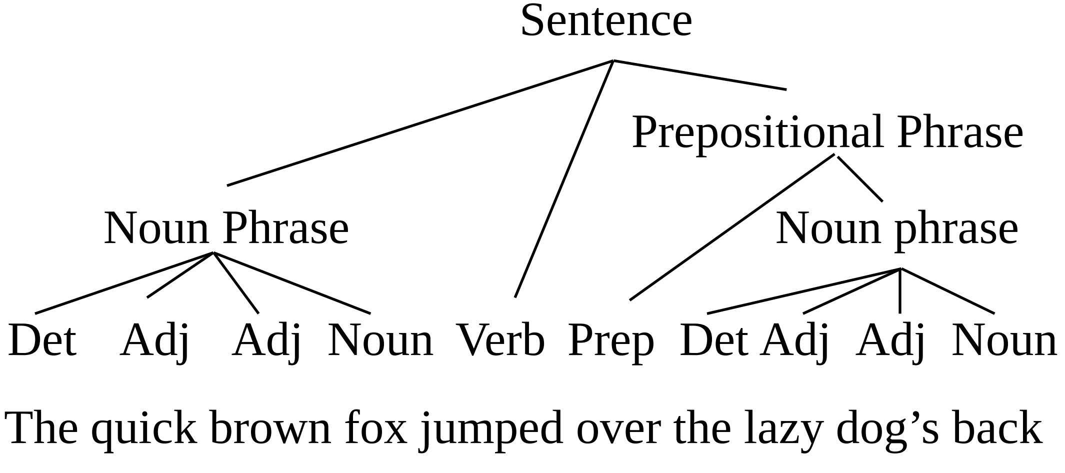
- Phrases can yield more precise queries
  - “University of Maryland”, “solar eclipse”
- Automated phrase detection can be harmful
  - Infelicitous choices result in missed matches
  - Therefore, never index only phrases
    - Better to index phrases <u>and</u> their constituent words
  - IR systems are good at evidence combination
    - Better evidence combination $\Rightarrow$ less help from phrases
- Parsing is still relatively slow and brittle
  - But Powerset is now trying to parse the entire Web

Lexical Phrases

- Same idea as longest substring match
  - But look for word (not character) sequences
- Compile a term list that includes phrases
  - Technical terminology can be very helpful
- Index any phrase that occurs in the list
- Most effective in a limited domain
  - Otherwise hard to capture most useful phrases

Syntactic Phrases

- Automatically construct "sentence diagrams"
  - Fairly good parsers are available
- Index the noun phrases
  - Might work for queries that focus on objects



Syntactic Variations

- The “paraphrase problem”
  - Prof. Douglas Oard studies information access patterns.
  - Doug studies patterns of user access to different kinds of information.
- Transformational variants (Jacquemin)
  - Coordinations
    - lung and breast cancer ⇒ lung cancer
  - Substitutions
    - inflammatory sinonasal disease ⇒ inflammatory disease
  - Permutations
    - addition of calcium ⇒ calcium addition

“Named Entity” Tagging

- Automatically assign “types” to words or phrases
  - Person, organization, location, date, money, …
- More rapid and robust than parsing
- Best algorithms use “supervised learning”
  - Annotate a corpus identifying entities and types
  - Train a probabilistic model
  - Apply the model to new text

Example: Predictive Annotation for Question Answering

In reality, at the time of Edison's 1879 patent, the light bulb
PERSON **TIME**

had been in existence for some five decades ….

Who patented the light bulb? → **patent light bulb PERSON**

When was the light bulb patented? → **patent light bulb TIME**

A “Term” is Whatever You Index

- Word sense
- Token
- Word
- Stem
- Character n-gram
- Phrase

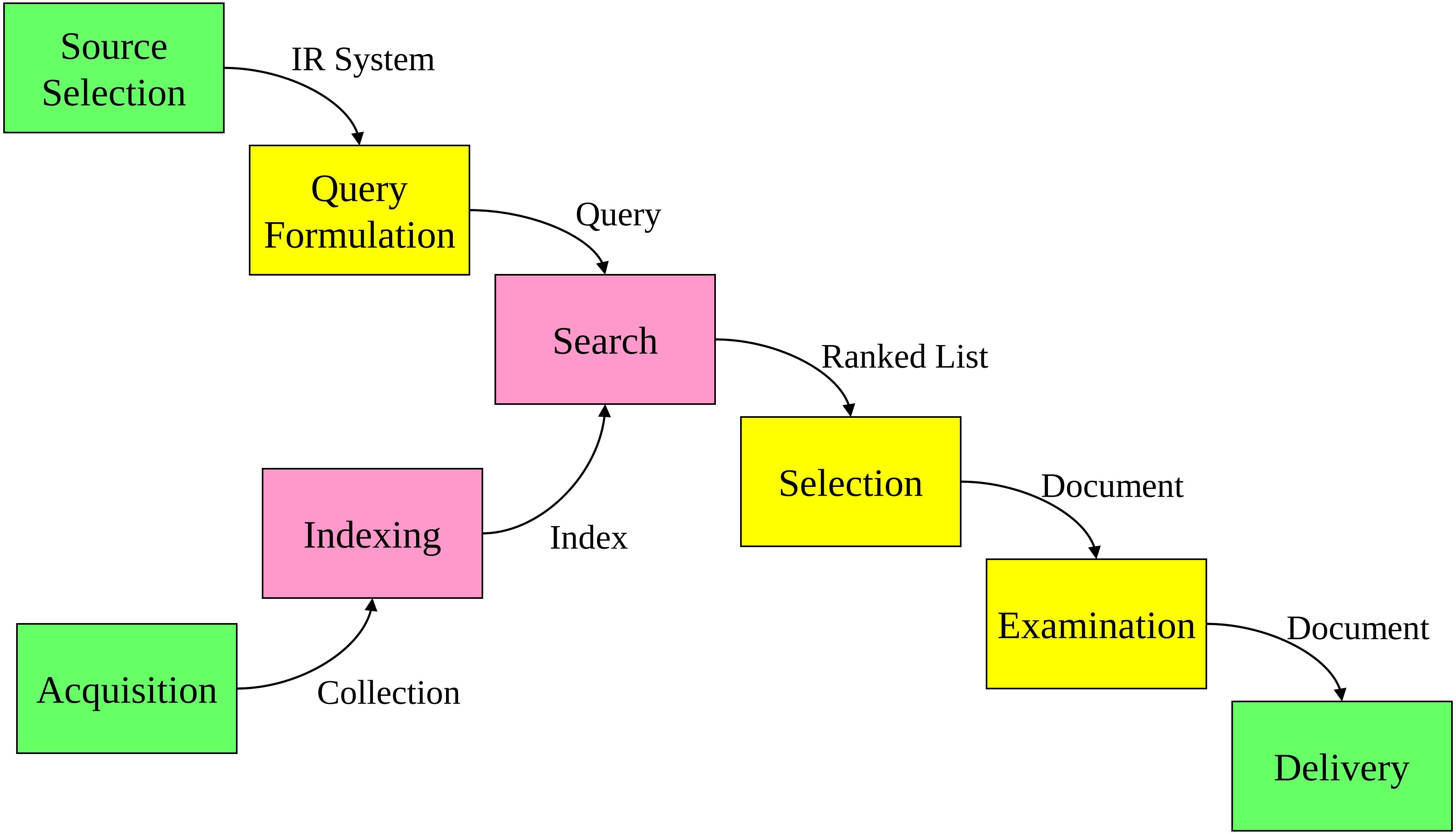
Summary

- The key is to index the right kind of terms
- Start by finding fundamental features
  - So far all we have talked about are character codes
  - Same ideas apply to handwriting, OCR, and speech
- Combine them into easily recognized units
  - Words where possible, character n-grams otherwise
- Apply further processing to optimize the system
  - Stemming is the most commonly used technique
  - Some “good ideas” don’t pan out that way

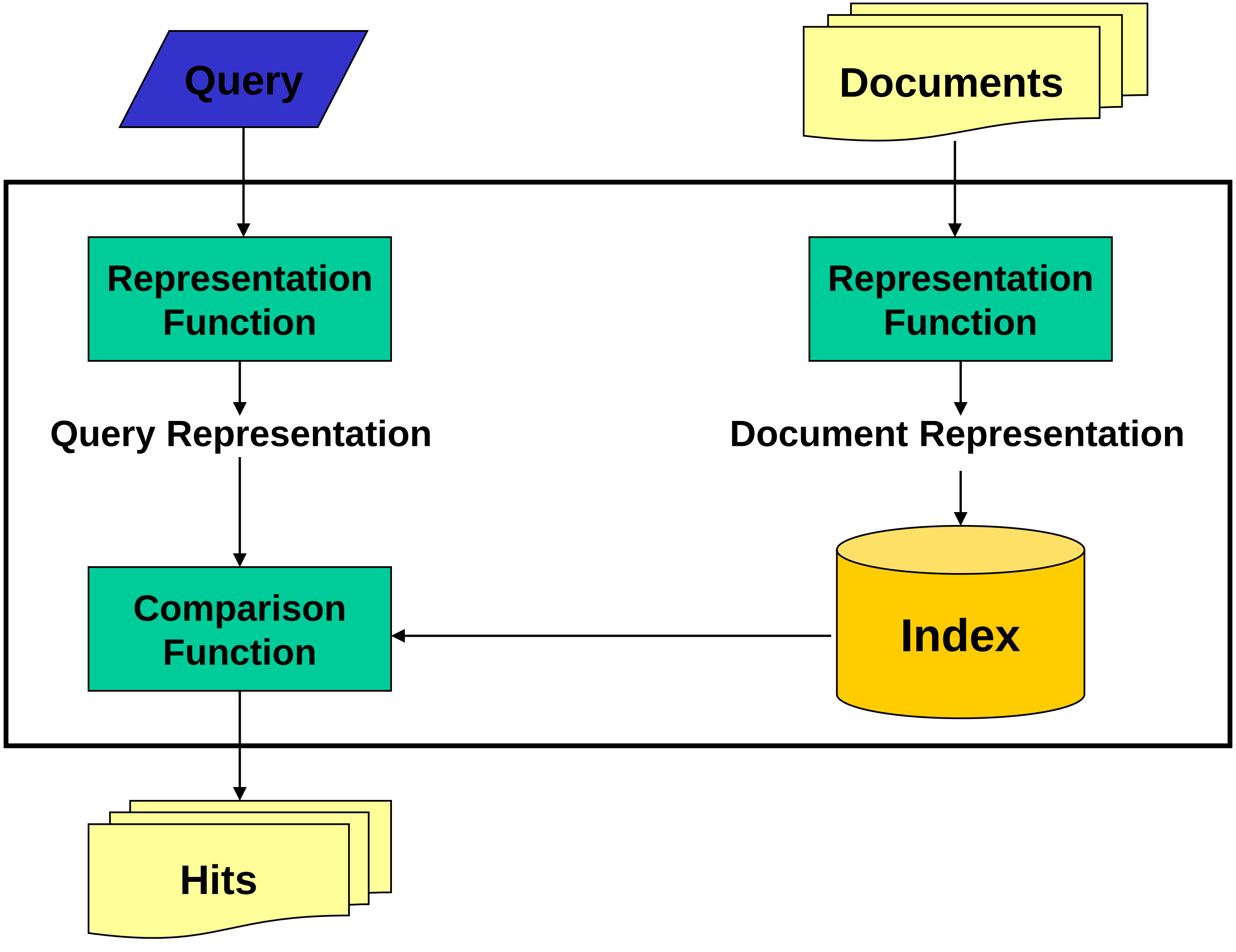
Agenda

- Character sets
- Terms as units of meaning
- ➢ Building an index
- MapReduce
- Project overview

Where Indexing Fits



Where Indexing Fits



A Cautionary Tale

- Windows “Search” scans a hard drive in minutes
  - If it only looks at the file names...
- How long would it take to scan all text on …
  - A 100 GB disk?
  - For the World Wide Web?
- Computers are getting faster, but…
  - How does Google give answers in seconds?

Some Questions for Today

- How long will it take to find a document?
  - Is there any work we can do in advance?
  - If so, how long will that take?
- How big a computer will I need?
  - How much disk space? How much RAM?
- What if more documents arrive?
  - How much of the advance work must be repeated?
  - Will searching become slower?
  - How much more disk space will be needed?

Desirable Index Characteristics

- Very rapid search
  - Less than ~100ms is typically imperceivable
- Reasonable hardware requirements
  - Processor speed, disk size, main memory size
- “Fast enough” creation and updates
  - Every couple of weeks may suffice for the Web
  - Every couple of minutes is needed for news

McDonald's slims down spuds

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

NEW YORK (CNN/Money) - McDonald's Corp. is cutting the amount of "bad" fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won't taste the same? The company says no. "It's a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile," said Mike Roberts, president of McDonald's USA.

But others are not so sure. McDonald's will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald's (MCD: down $0.54 to $23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy's International (WEN: down $0.80 to $34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

…

16 × said

14 × McDonalds

12 × fat

11 × fries

8 × new

6 × company, french, nutrition

5 × food, oil, percent, reduce, taste, Tuesday

…

"Bag of Words"

“Bag of **Terms**” Representation

- Bag = a “set” that can contain duplicates
  - “The quick brown fox jumped over the lazy dog’s back” →

    {back, brown, dog, fox, jump, lazy, over, quick, the, the}

- Vector = values recorded in any consistent order
  - *{back, brown, dog, fox, jump, lazy, over, quick, the, the}* → [1 1 1 1 1 1 1 1 2]

Why Does “Bag of Terms” Work?

- Words alone tell us a lot about content

  Random: beating takes points falling another Dow 355

  Alphabetical: 355 another beating Dow falling points

  Actual: Dow takes another beating, falling 355 points

- It is relatively easy to come up with words that describe an information need

Bag of Terms Example

Document 1

The quick brown fox jumped over the lazy dog's back.

Document 2

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

Term	Document 1	Document 2
aid	0	1
all	0	1
back	1	0
brown	1	0
come	0	1
dog	1	0
fox	1	0
good	0	1
jump	1	0
lazy	1	0
men	0	1
now	0	1
over	1	0
party	0	1
quick	1	0
their	0	1
time	0	1

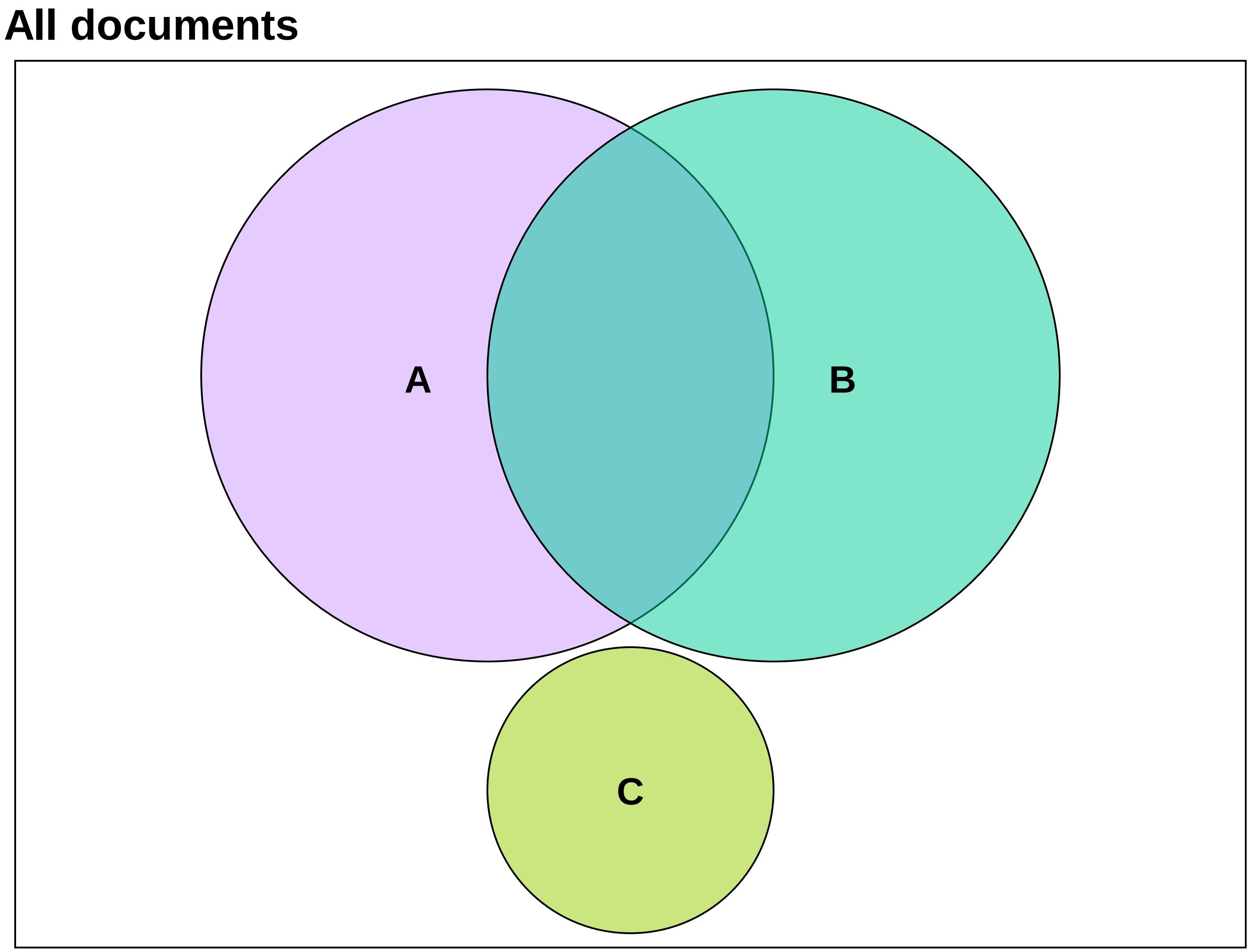
Stopword List

for
is
of
the
to

Boolean “Free Text” Retrieval

- Limit the bag of words to “absent” and “present”
  - “Boolean” values, represented as 0 and 1
- Represent terms as a “bag of documents”
  - Same representation, but <u>rows rather than columns</u>
- Combine the rows using “Boolean operators”
  - AND, OR, NOT
- Result set: every document with a 1 remaining

AND/OR/NOT



Boolean Operators

A OR B

A \ B	0	1
0	0	1
1	1	1

NOT B

B	0	1
	1	0

A AND B

A \ B	0	1
0	0	0
1	0	1

A NOT B
(= A AND NOT B)

A \ B	0	1
0	0	0
1	1	0

Boolean View of a Collection

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
aid	0	0	0	1	0	0	0	1
all	0	1	0	1	0	1	0	0
back	1	0	1	0	0	0	1	0
brown	1	0	1	0	1	0	1	0
come	0	1	0	1	0	1	0	1
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0
good	0	1	0	1	0	1	0	1
jump	0	0	1	0	0	0	0	0
lazy	1	0	1	0	1	0	1	0
men	0	1	0	1	0	0	0	1
now	0	1	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1
party	0	0	0	0	0	1	0	1
quick	1	0	1	0	0	0	0	0
their	1	0	0	0	1	0	1	0
time	0	1	0	1	0	1	0	0

Each column represents the view of a particular document: What terms are contained in this document?

Each row represents the view of a particular term: What documents contain this term?

To execute a query, pick out rows corresponding to query terms and then apply logic table of corresponding Boolean operator

Sample Queries

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8	
dog	0	0	1	0	1	0	0	0	
fox	0	0	1	0	1	0	1	0	
dog $\wedge$ fox	0	0	1	0	1	0	0	0	dog AND fox $\rightarrow$ Doc 3, Doc 5
dog $\vee$ fox	0	0	1	0	1	0	1	0	dog OR fox $\rightarrow$ Doc 3, Doc 5, Doc 7
dog $\neg$ fox	0	0	0	0	0	0	0	0	dog NOT fox $\rightarrow$ empty
fox $\neg$ dog	0	0	0	0	0	0	1	0	fox NOT dog $\rightarrow$ Doc 7

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8	
good	0	1	0	1	0	1	0	1	
party	0	0	0	0	0	1	0	1	
g $\wedge$ p	0	0	0	0	0	1	0	1	good AND party $\rightarrow$ Doc 6, Doc 8
over	1	0	1	0	1	0	1	1	
g $\wedge$ p $\neg$ o	0	0	0	0	0	1	0	0	good AND party NOT over $\rightarrow$ Doc 6

Why Boolean Retrieval Works

- Boolean operators approximate natural language
  - Find documents about a good party that is not over
- AND can discover relationships between concepts
  - good party
- OR can discover alternate terminology
  - excellent party
- NOT can discover alternate meanings
  - Democratic party

Proximity Operators

- More precise versions of AND
  - "NEAR n" allows at most n-1 intervening terms
  - "WITH" requires terms to be adjacent and in order
- Easy to implement, but less efficient
  - Store a list of positions for each word in each doc
    - Warning: stopwords become important!
  - Perform normal Boolean computations
    - Treat WITH and NEAR like AND with an extra constraint

Proximity Operator Example

Term	Doc 1	Doc 2
aid	0	1 (13)
all	0	1 (6)
back	1 (10)	0
brown	1 (3)	0
come	0	1 (9)
dog	1 (9)	0
fox	1 (4)	0
good	0	1 (7)
jump	1 (5)	0
lazy	1 (8)	0
men	0	1 (8)
now	0	1 (1)
over	1 (6)	0
party	0	1 (16)
quick	1 (2)	0
their	0	1 (15)
time	0	1 (4)

- time AND come
  - Doc 2
- time (NEAR 2) come
  - Empty
- quick (NEAR 2) fox
  - Doc 1
- quick WITH fox
  - Empty

Other Extensions

- Ability to search on fields
  - Leverage document structure: title, headings, etc.
- Wildcards
  - lov* = love, loving, loves, loved, etc.
- Special treatment of dates, names, companies, etc.

WESTLAW® Query Examples

- What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM

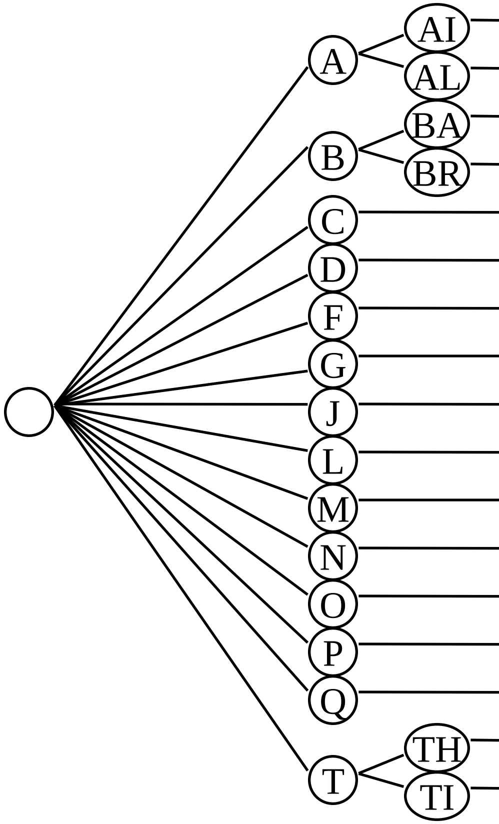
- What factors are important in determining what constitutes a vessel for purposes of determining liability of a vessel owner for injuries to a seaman under the “Jones Act” (46 USC 688)?
  - (741 +3 824) FACTOR ELEMENT STATUS FACT /P VESSEL SHIP BOAT /P (46 +3 688) “JONES ACT” /P INJUR! /S SEAMAN CREWMAN WORKER

- Are there any cases which discuss negligent maintenance or failure to maintain aids to navigation such as lights, buoys, or channel markers?
  - NOT NEGLECT! FAIL! NEGLIG! /5 MAINT! REPAIR! /P NAVIGAT! /5 AID EQUIP! LIGHT BUOY “CHANNEL MARKER”

- What cases have discussed the concept of excusable delay in the application of statutes of limitations or the doctrine of laches involving actions in admiralty or under the “Jones Act” or the “Death on the High Seas Act”?
  - EXCUS! /3 DELAY /P (LIMIT! /3 STATUTE ACTION) LACHES /P “JONES ACT” “DEATH ON THE HIGH SEAS ACT” (46 +3 761)

An “Inverted Index”

Term Index



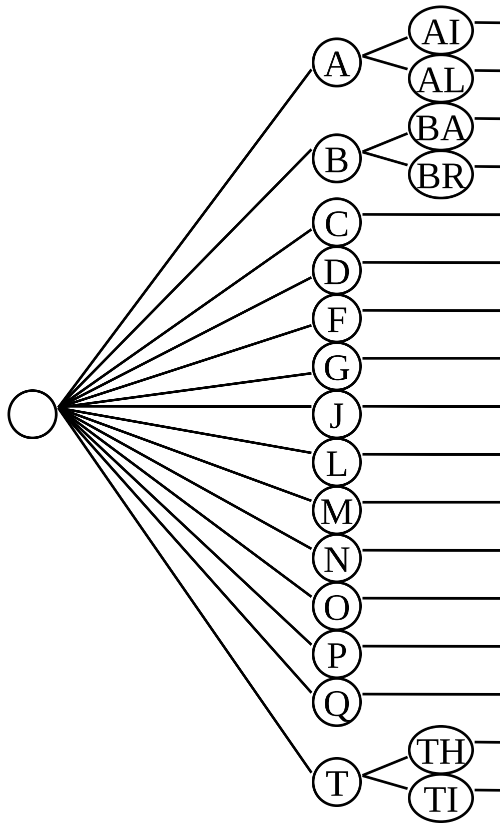
Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8	Postings
aid	0	0	0	1	0	0	0	1	4, 8
all	0	1	0	1	0	1	0	0	2, 4, 6
back	1	0	1	0	0	0	1	0	1, 3, 7
brown	1	0	1	0	1	0	1	0	1, 3, 5, 7
come	0	1	0	1	0	1	0	1	2, 4, 6, 8
dog	0	0	1	0	1	0	0	0	3, 5
fox	0	0	1	0	1	0	1	0	3, 5, 7
good	0	1	0	1	0	1	0	1	2, 4, 6, 8
jump	0	0	1	0	0	0	0	0	3
lazy	1	0	1	0	1	0	1	0	1, 3, 5, 7
men	0	1	0	1	0	0	0	1	2, 4, 8
now	0	1	0	0	0	1	0	1	2, 6, 8
over	1	0	1	0	1	0	1	1	1, 3, 5, 7, 8
party	0	0	0	0	0	1	0	1	6, 8
quick	1	0	1	0	0	0	0	0	1, 3
their	1	0	0	0	1	0	1	0	1, 5, 7
time	0	1	0	1	0	1	0	0	2, 4, 6

Saving Space

- Can we make this data structure smaller, keeping in mind the need for fast retrieval?
- Observations:
  - The nature of the search problem requires us to quickly find which documents contain a term
  - The term-document matrix is very sparse
  - Some terms are more useful than others

What Actually Gets Stored

Term Index



Term	Postings
aid	4, 8
all	2, 4, 6
back	1, 3, 7
brown	1, 3, 5, 7
come	2, 4, 6, 8
dog	3, 5
fox	3, 5, 7
good	2, 4, 6, 8
jump	3
lazy	1, 3, 5, 7
men	2, 4, 8
now	2, 6, 8
over	1, 3, 5, 7, 8
party	6, 8
quick	1, 3
their	1, 5, 7
time	2, 4, 6

Deconstructing the Inverted Index

The term Index	Postings File
aid	4, 8
all	2, 4, 6
back	1, 3, 7
brown	1, 3, 5, 7
come	2, 4, 6, 8
dog	3, 5
fox	3, 5, 7
good	2, 4, 6, 8
jump	3
lazy	1, 3, 5, 7
men	2, 4, 8
now	2, 6, 8
over	1, 3, 5, 7, 8
party	6, 8
quick	1, 3
their	1, 5, 7
time	2, 4, 6

Term Index Size

- Heap's Law tells us about vocabulary size

$$V = Kn^{\beta}$$

$K \approx 20, \beta \approx 0.6$

V is vocabulary size
n is corpus size (number of documents)
K and β are constants

  - When adding new documents, the system is likely to have seen terms already
  - Usually fits in RAM
- But the postings file keeps growing!

Linear Dictionary Lookup

Suppose we want to find the word “complex”

relaxation
astronomical
zebra
belligerent
subterfuge
daffodil
cadence
wingman
loiter
peace
arcade
respondent
complex
tax
kingdom
jambalaya

Found it!

- How long does this take, in the worst case?
- Running time is proportional to number of entries in the dictionary
- This algorithm is O(*n*) = linear time algorithm

With a Sorted Dictionary

Let's try again, except this time with a sorted dictionary: find "complex"

arcade
astronomical
belligerent
cadence
complex
daffodil
jambalaya
kingdom
loiter
peace
relaxation
respondent
subterfuge
tax
wingman
zebra

Found it!

- How long does this take, in the worst case?

Which is Faster?

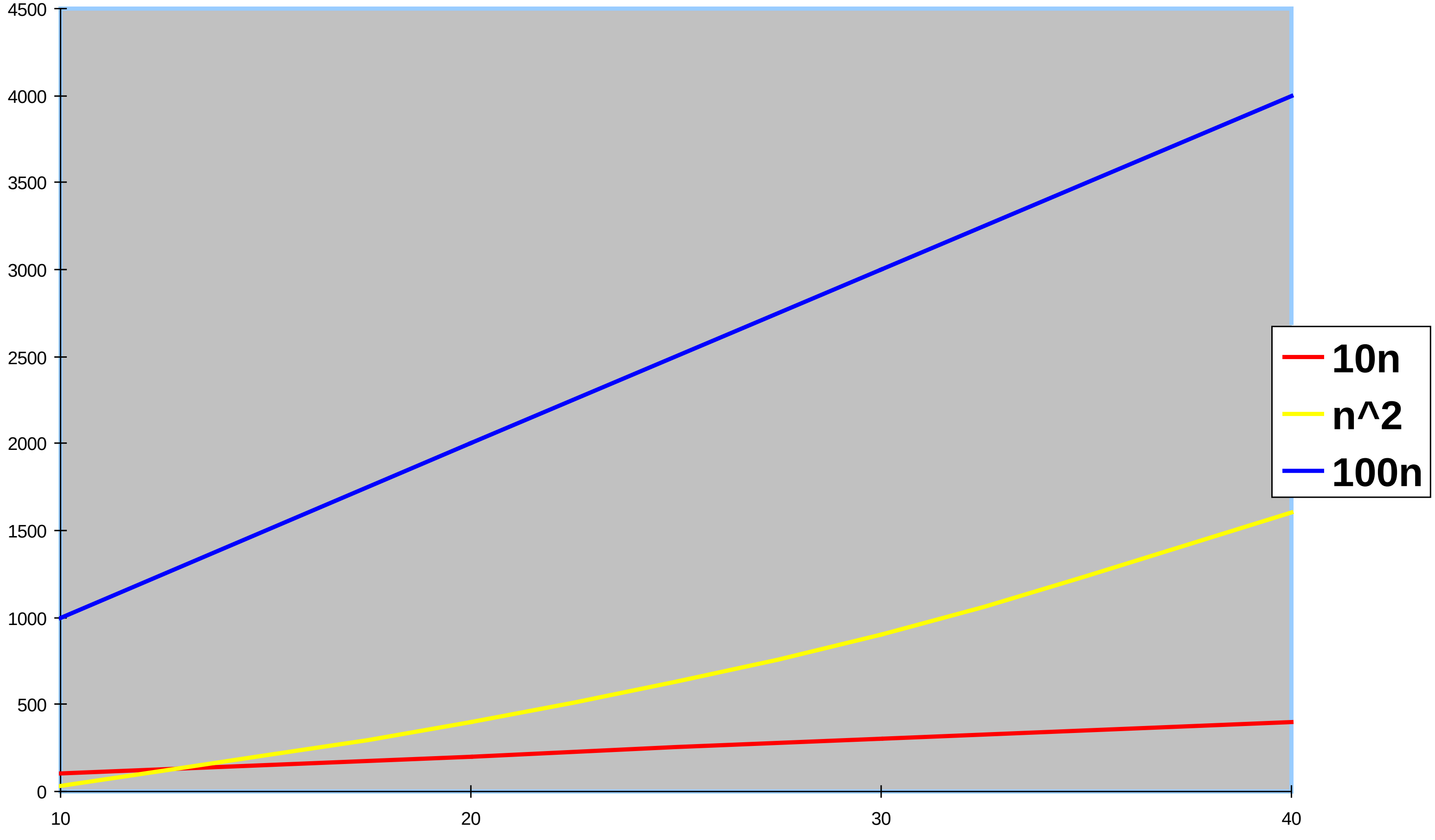
- Two algorithms:
  - O(n): Sequentially “search”
  - O($\log n$): Binary “search”

- Big-O notation
  - Allows us to compare different algorithms on <u>very large</u> collections

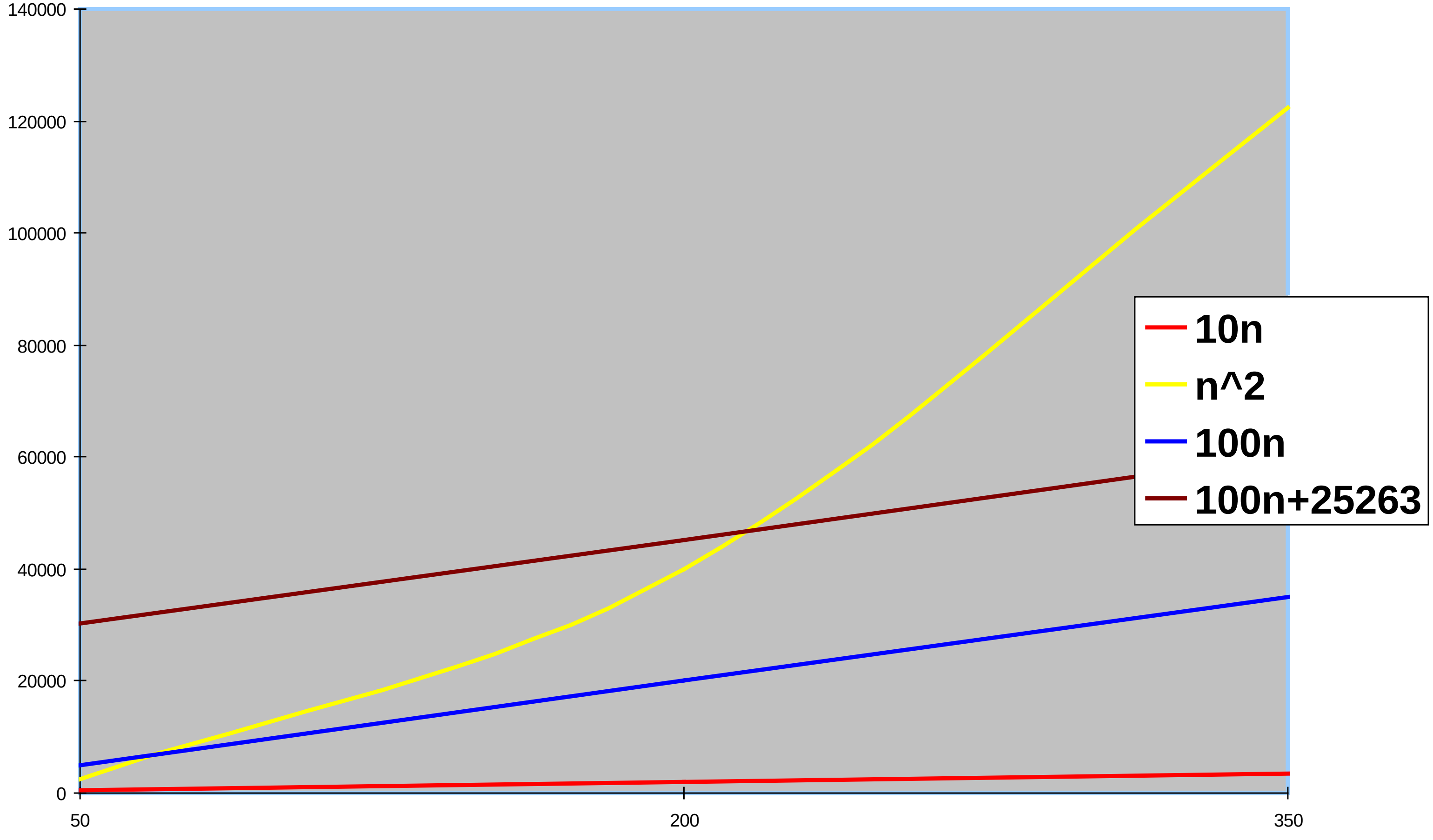
Computational Complexity

- Time complexity: how long will it take …
  - At index-creation time?
  - At query time?
- Space complexity: how much memory is needed …
  - In RAM?
  - On disk?
- Things you need to know to assess complexity:
  - What is the “size” of the input? (“n”)
  - What are the internal data structures?
  - What is the algorithm?

Complexity for Small n



“Asymptotic” Complexity



Building a Term Index

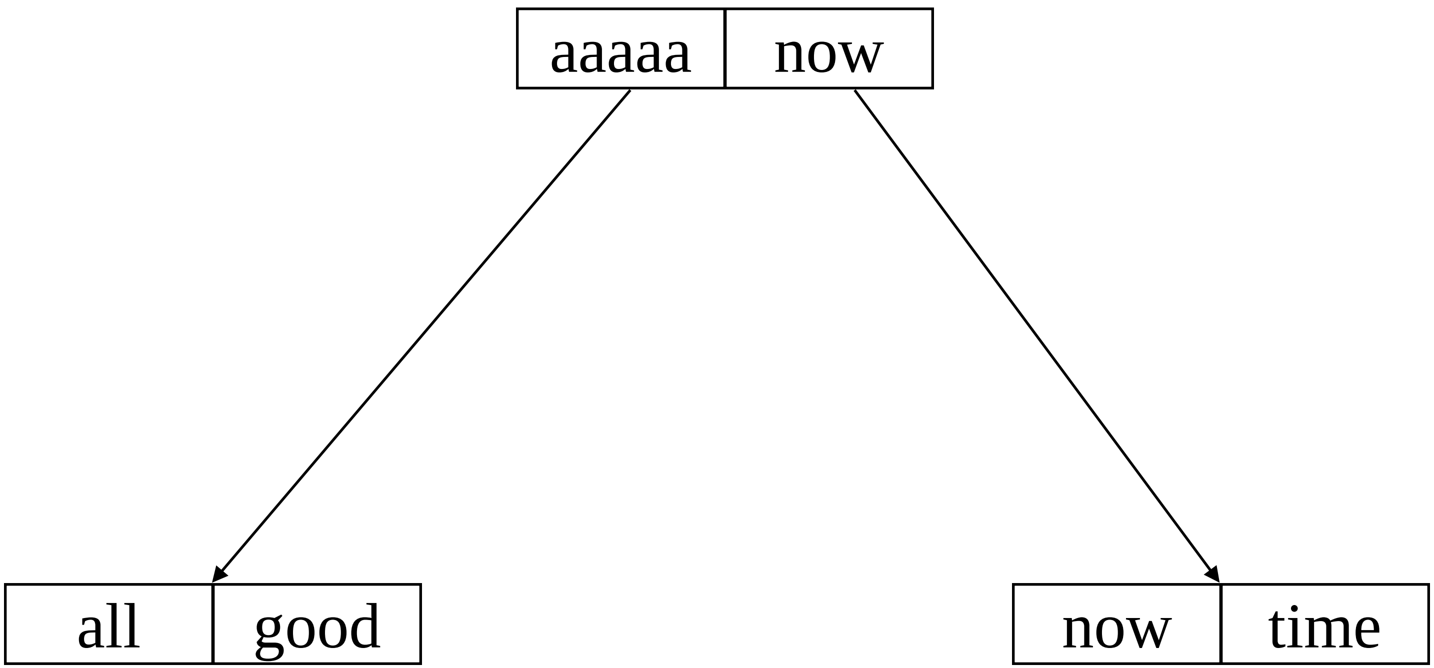
- Simplest solution is a single sorted array
  - Fast lookup using binary search
  - But sorting is expensive [it’s O(n * log n)]
    - And adding one document means starting over

- Tree structures allow easy insertion
  - But the worst case lookup time is O(n)

- Balanced trees provide the best of both
  - Fast lookup [O (log n) and easy insertion [O(log n)]
  - But they require 45% more disk space

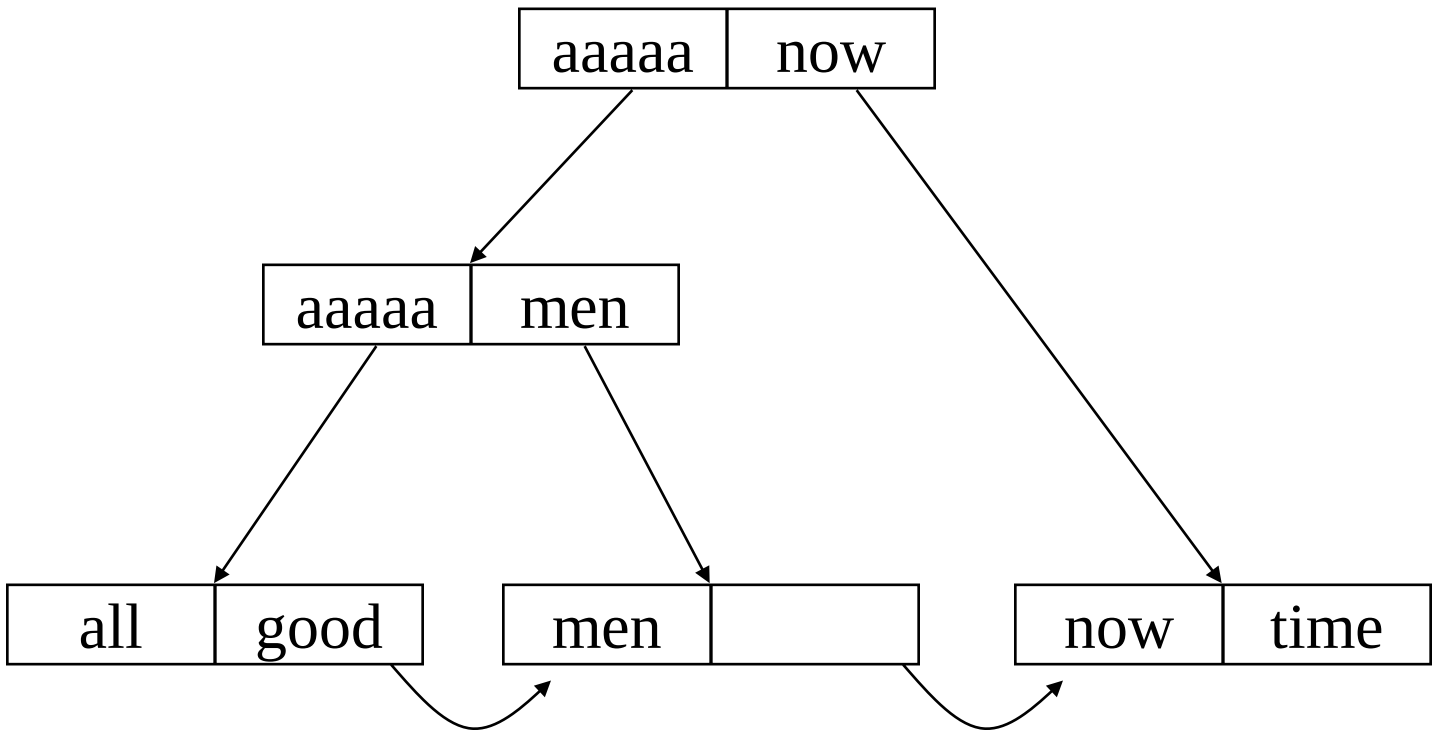
Starting a B+ Tree Term Index

Now is the time for all good …



Adding a New Term

Now is the time for all good men …



What’s in the Postings File?

- Boolean retrieval
  - Just the document number
- Proximity operators
  - Word offsets for each occurrence of the term
    - Example: Doc 3 (t17, t36), Doc 13 (t3, t45)
- Ranked Retrieval
  - Document number and term weight

How Big Is a Raw Postings File?

- Very compact for Boolean retrieval
  - About 10% of the size of the documents
    - If an aggressive stopword list is used!
- Not much larger for ranked retrieval
  - Perhaps 20%
- Enormous for proximity operators
  - Sometimes larger than the documents!

Large Postings Files are Slow

- RAM
  - Typical size: 1 GB
  - Typical access speed: 50 ns
- Hard drive:
  - Typical size: 80 GB (my laptop)
  - Typical access speed: 10 ms
- Hard drive is 200,000x slower than RAM!

- Discussion question:
  - How does stopword removal improve speed?

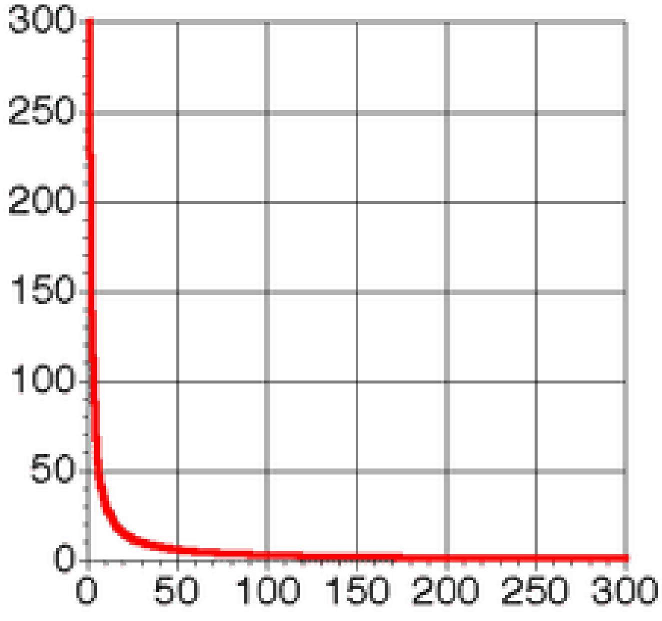
Zipf's Law

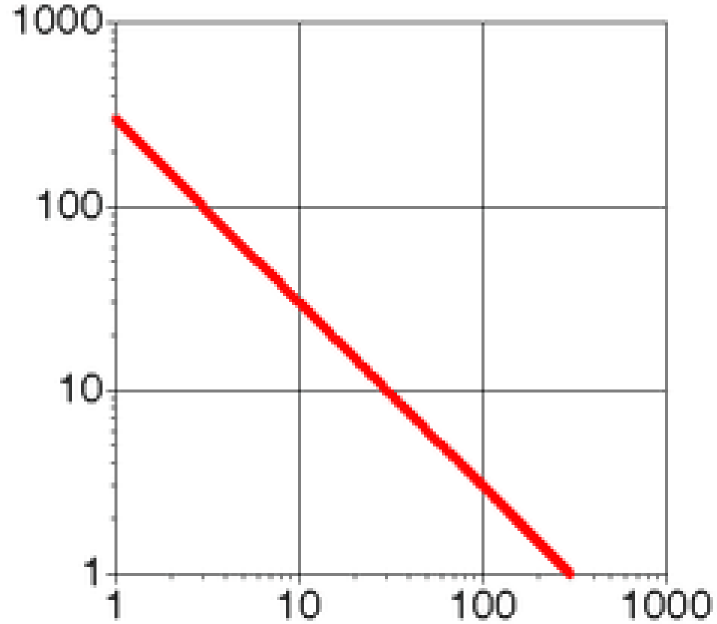
- George Kingsley Zipf (1902-1950) observed that for many frequency distributions, the *n*th most frequent event is related to its frequency in the following manner:

$$f \cdot r = c \qquad \text{or} \qquad f = \frac{c}{r}$$

f = frequency
r = rank
c = constant

Zipfian Distribution: The “Long Tail”





- A few elements occur very frequently
- Many elements occur very infrequently

Some Zipfian Distributions

- Library book checkout patterns
- Website popularity
- Incoming Web page requests
- Outgoing Web page requests
- Document size on Web

Word Frequency in English

Frequency of 50 most common words in English
(sample of 19 million words)

the	1130021	from	96900	or	54958
of	547311	he	94585	about	53713
to	516635	million	93515	market	52110
a	464736	year	90104	they	51359
in	390819	its	86774	this	50933
and	387703	be	85588	would	50828
that	204351	was	83398	you	49281
for	199340	company	83070	which	48273
is	152483	an	76974	bank	47940
said	148302	has	74405	stock	47401
it	134323	are	74097	trade	47310
on	121173	have	73132	his	47116
by	118863	but	71887	more	46244
as	109135	will	71494	who	42142
at	101779	say	66807	one	41635
mr	101679	new	64456	their	40910
with	101210	share	63925		

Demonstrating Zipf's Law

The following shows *rf*1000/n*

r is the rank of word *w* in the sample

f is the frequency of word *w* in the sample

n is the total number of word occurrences in the sample

the	59	from	92	or	101
of	58	he	95	about	102
to	82	million	98	market	101
a	98	year	100	they	103
in	103	its	100	this	105
and	122	be	104	would	107
that	75	was	105	you	106
for	84	company	109	which	107
is	72	an	105	bank	109
said	78	has	106	stock	110
it	78	are	109	trade	112
on	77	have	112	his	114
by	81	but	114	more	114
as	80	will	117	who	106
at	80	say	113	one	107
mr	86	new	112	their	108
with	91	share	114		

Index Compression

- CPU's are much faster than disks
  - A disk can transfer 1,000 bytes in ~20 ms
  - The CPU can do ~10 million instructions in that time
- Compressing the postings file is a big win
  - Trade decompression time for fewer disk reads
- Key idea: reduce redundancy
  - Trick 1: store relative offsets (some will be the same)
  - Trick 2: use an optimal coding scheme

Compression Example

- Postings (one byte each = 7 bytes = 56 bits)
  - 37, 42, 43, 48, 97, 98, 243
- Difference
  - 37, 5, 1, 5, 49, 1, 145
- Optimal (variable length) Huffman Code
  - 0:1, 10:5, 110:37, 1110:49, 1111: 145
- Compressed (17 bits)
  - 11010010111001111

Remember This?

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
dog	0	0	1	0	1	0	0	0
fox	0	0	1	0	1	0	1	0

dog $\wedge$ fox	0	0	1	0	1	0	0	0

dog AND fox → Doc 3, Doc 5

dog $\vee$ fox	0	0	1	0	1	0	1	0

dog OR fox → Doc 3, Doc 5, Doc 7

dog $\neg$ fox	0	0	0	0	0	0	0	0

dog NOT fox → empty

fox $\neg$ dog	0	0	0	0	0	0	1	0

fox NOT dog → Doc 7

Term	Doc 1	Doc 2	Doc 3	Doc 4	Doc 5	Doc 6	Doc 7	Doc 8
good	0	1	0	1	0	1	0	1
party	0	0	0	0	0	1	0	1

g $\wedge$ p	0	0	0	0	0	1	0	1
over	1	0	1	0	1	0	1	1

good AND party → Doc 6, Doc 8

g $\wedge$ p $\neg$ o	0	0	0	0	0	1	0	0

good AND party NOT over → Doc 6

Indexing-Time, Query-Time

- Indexing
  - Walk the term index, splitting if needed
  - Insert into the postings file in sorted order
  - Hours or days for large collections

- Query processing
  - Walk the term index for each query term
  - Read the postings file for that term from disk
  - Compute search results from posting file entries
  - Seconds, even for enormous collections

Summary

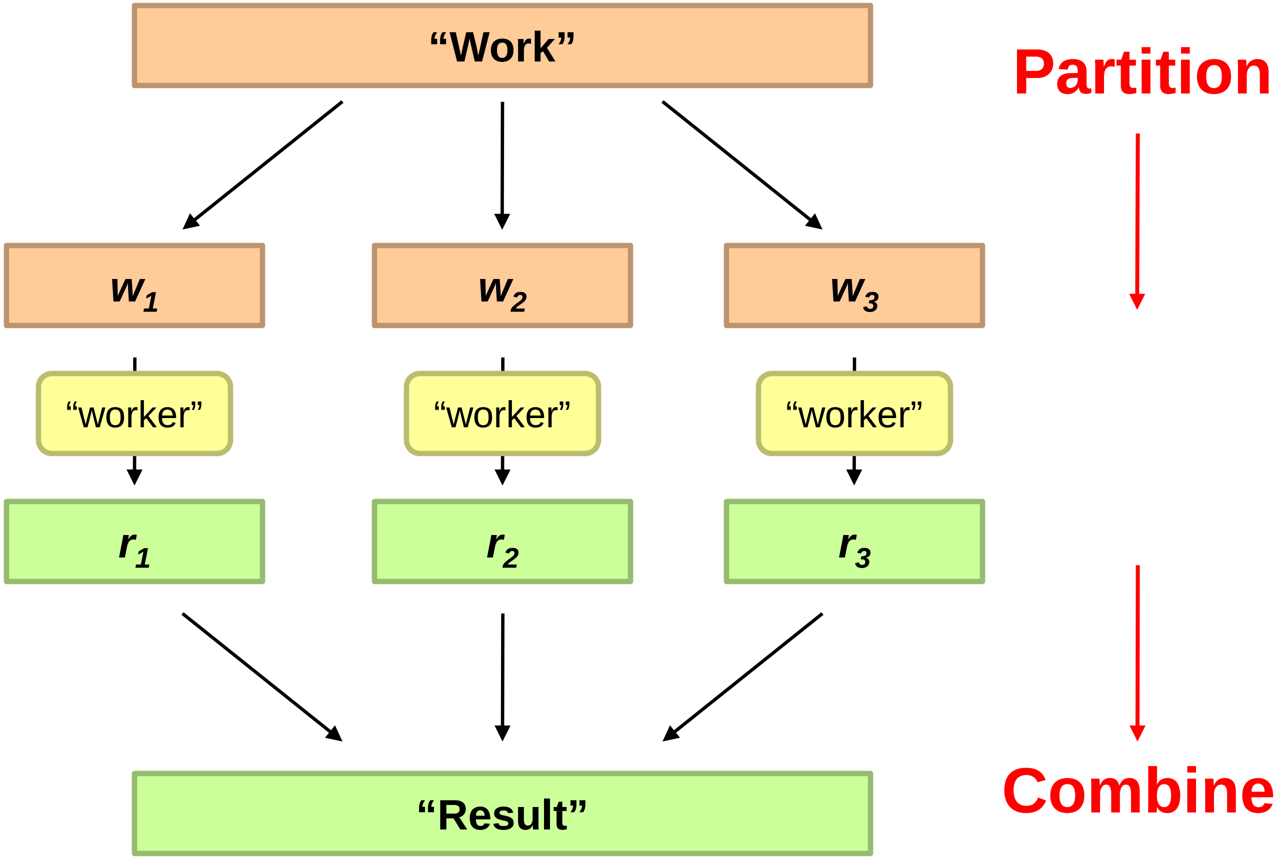
- Slow indexing yields fast query processing
  - Key fact: most terms don't appear in most documents
- We use extra disk space to save query time
  - Index space is in addition to document space
  - Time and space complexity must be balanced
- Disk block reads are the critical resource
  - This makes index compression a big win

Agenda

- Character sets
- Terms as units of meaning
- Building an index
- ➢ MapReduce
- ➢ Project Overview

Source: Wikipedia (IBM Roadrunner)

Divide and Conquer



Parallelization Challenges

- How do we assign work units to workers?
- What if we have more work units than workers?
- What if workers need to share partial results?
- How do we aggregate partial results?
- How do we know all the workers have finished?
- What if workers die?

What is the common theme of all of these problems?

Managing Multiple Workers

- Difficult because
  - We don’t know the order in which workers run
  - We don’t know when workers interrupt each other
  - We don’t know the order in which workers access shared data
- Thus, we need:
  - Semaphores (lock, unlock)
  - Conditional variables (wait, notify, broadcast)
  - Barriers
- Still, lots of problems:
  - Deadlock, livelock, race conditions...
  - Dining philosophers, sleeping barbers, cigarette smokers...
- Moral of the story: be careful!

“Big Ideas”

- Scale “out”, not “up”
  - Limits of SMP and large shared-memory machines
- Move processing to the data
  - Cluster have limited bandwidth
- Process data sequentially, avoid random access
  - Seeks are expensive, disk throughput is reasonable
- Seamless scalability
  - From the mythical man-month to the tradable machine-hour

Typical Large-Data Problem

- Iterate over a large number of records
- Extract something of interest from each
- Shuffle and sort intermediate results
- Aggregate intermediate results
- Generate final output

Map

Reduce

Key idea: provide a functional abstraction for these two operations

(Dean and Ghemawat, OSDI 2004)

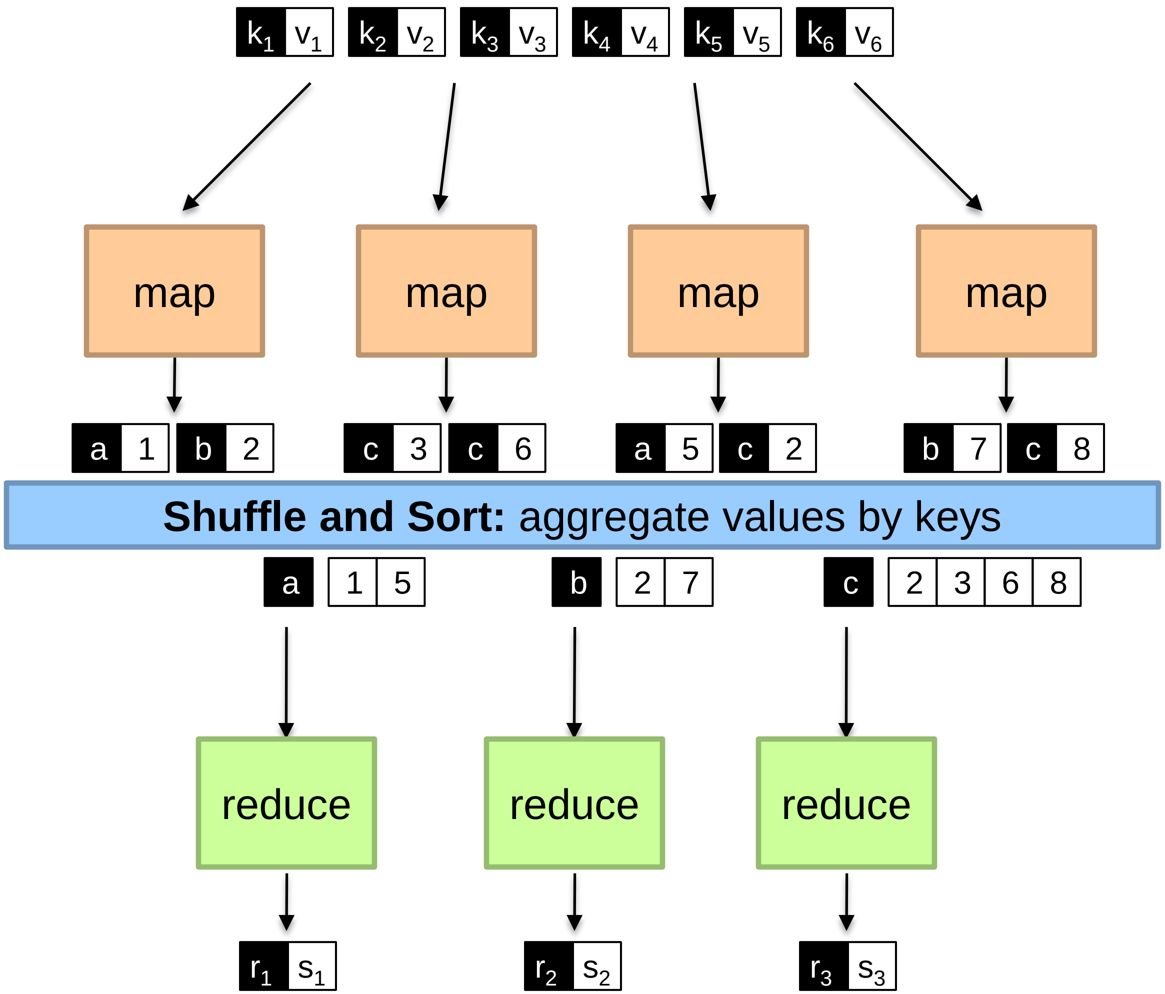
MapReduce

- Programmers specify two functions:

  map (k, v) → <k’, v’>*

  reduce (k’, v’) → <k’, v’>*

  - All values with the same key are sent to the same reducer
- The execution framework handles everything else…



k_1 v_1 k_2 v_2 k_3 v_3 k_4 v_4 k_5 v_5 k_6 v_6
map
map
map
map
a 1 b 2
c 3 c 6
a 5 c 2
b 7 c 8
Shuffle and Sort: aggregate values by keys
a 1 5
b 2 7
c 2 3 6 8
reduce
reduce
reduce
r_1 s_1
r_2 s_2
r_3 s_3

MapReduce

- Programmers specify two functions:

  map (k, v) → <k', v'>*

  reduce (k', v') → <k', v'>*

  - All values with the same key are sent to the same reducer
- The execution framework handles everything else…

What's "everything else"?

MapReduce “Runtime”

- Handles scheduling
  - Assigns workers to map and reduce tasks
- Handles “data distribution”
  - Moves processes to data
- Handles synchronization
  - Gathers, sorts, and shuffles intermediate data
- Handles errors and faults
  - Detects worker failures and restarts
- Everything happens on top of a distributed FS (later)

MapReduce

- Programmers specify two functions:

  map (k, v) → <k’, v’>*

  reduce (k’, v’) → <k’, v’>*

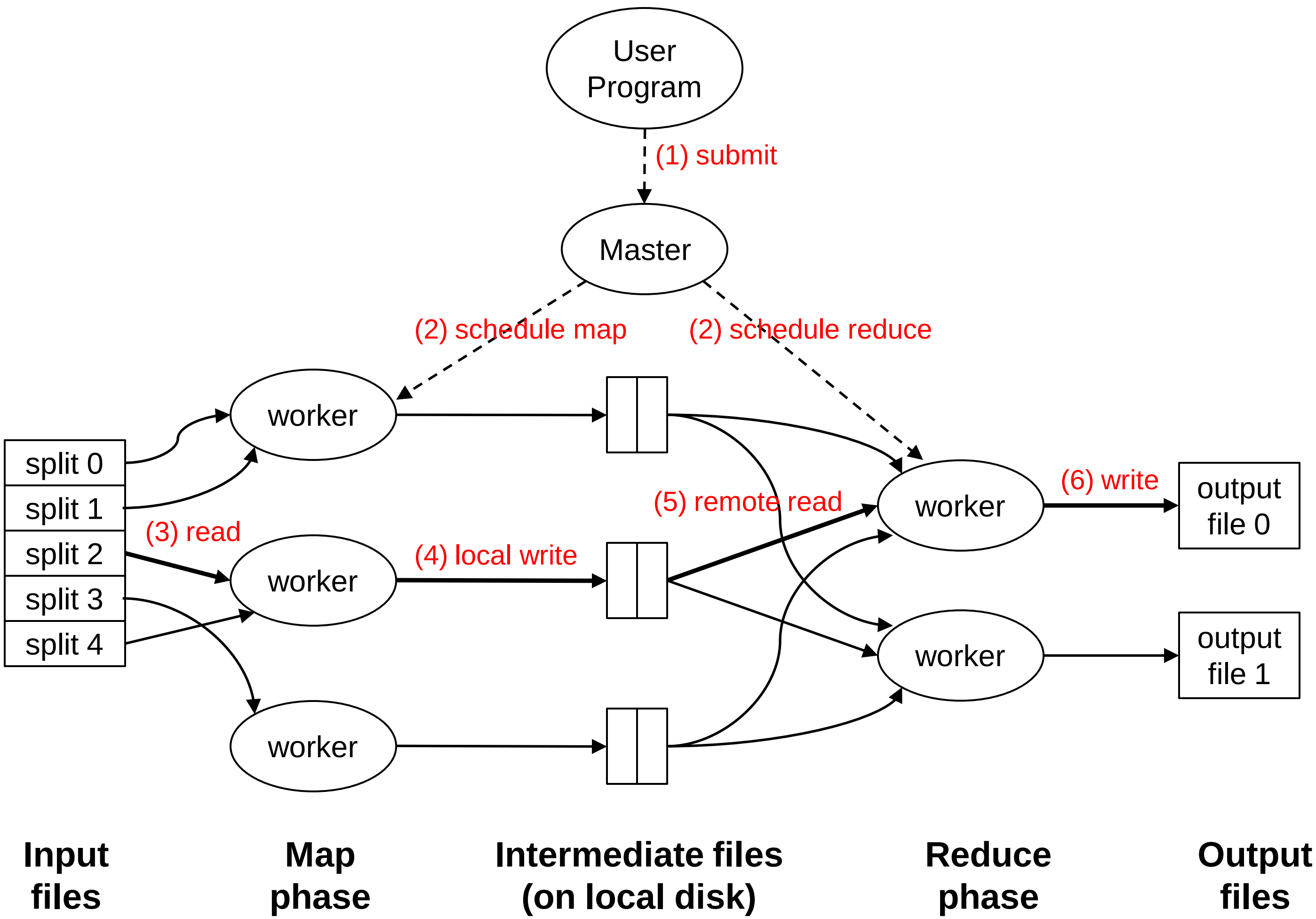
  - All values with the same key are reduced together
- The execution framework handles everything else…
- Not quite…usually, programmers also specify:

  partition (k’, number of partitions) → partition for k’

  - Often a simple hash of the key, e.g., hash(k’) mod n
  - Divides up key space for parallel reduce operations

  combine (k’, v’) → <k’, v’>*

  - Mini-reducers that run in memory after the map phase
  - Used as an optimization to reduce network traffic



Adapted from (Dean and Ghemawat, OSDI 2004)

Project Options

- Instructor-designed project
  - Team of ~6: design, implementation, evaluation
  - Data is in hand, broad goals are outlined
  - Fixed “deliverable” schedule

- Roll-your-own project
  - Individual, or group of any (reasonable) size
  - Pick your own topic and deliverables
  - Requires my approval (start discussion by Feb 16)

Before You Go!

On a sheet of paper, please briefly answer the following question (no names):

What was the muddiest point in today’s lecture?

Don’t forget the homework due next week!