Cloud Computing Lecture #4

Information Retrieval with MapReduce

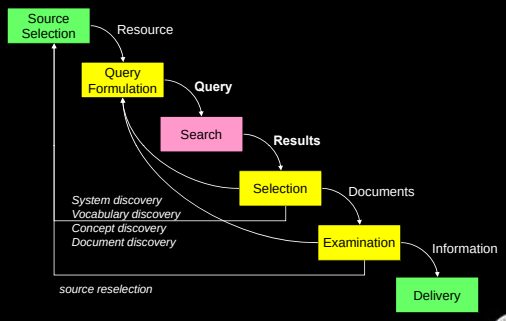
Jimmy Lin
The iSchool
University of Maryland

Wednesday, September 17, 2008

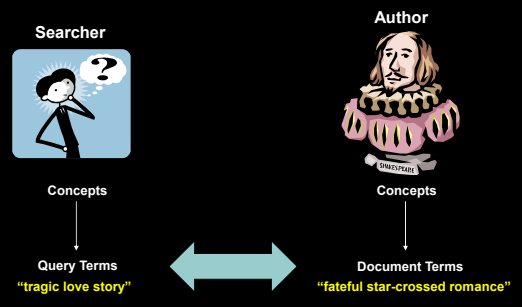
Today's Topics

- Introduction to IR
- Boolean retrieval
- Ranked retrieval
- IR with MapReduce

The Information Retrieval Cycle

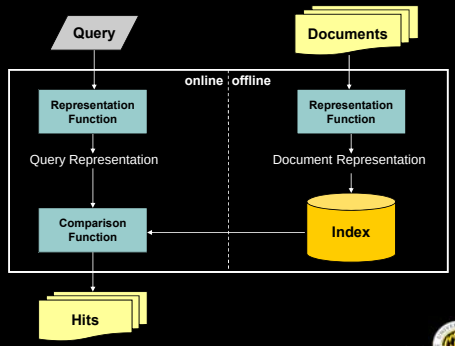


The Central Problem in Search



Do these represent the same concepts?

Architecture of IR Systems



How do we represent text?

- Remember: computers don't "understand" anything!
- "Bag of words"
  - Treat all the words in a document as index terms for that document
  - Assign a "weight" to each term based on "importance"
  - Disregard order, structure, meaning, etc. of the words
  - Simple, yet effective!
- Assumptions
  - Term occurrence is independent
  - Document relevance is independent
  - "Words" are well-defined

What's a word?

天主教教宗若望保祿二世因感冒再度住進醫院。
這是他今年第二度因同樣的病因住院。

وقال مارك ريجيف - الناطق باسم
الخارجية الإسرائيلية - إن شارون قبل
الدعوة وسيقوم للمرة الأولى بزيارة
تونس، التي كانت لفترة طويلة المقر
الرسمي لمنظمة التحرير الفلسطينية بعد خروجها من لبنان عام 1982.

Выступая в Мещанском суде Москвы экс-глава ЮКОСа
заявил не совершал ничего противозаконного, в чем
обвиняет его генпрокуратура России.

भारत सरकार ने वित्तीय वर्ष 2005-06 में सात फ़ीसदी
विकास दर हासिल करने का आकलन किया है और कर सुधार पर ज़ोर दिया है

日米連合で台頭中国に対処…アーミテージ前副長官提言

조지영 기자= 서울시는 25일 이명박 시장이 `행정중심복합도시" 건설안에 대해 `군대라도 동원해 막고 싶은 심정"이라고 말했다는 일부 언론의 보도를 부인했다.

Sample Document

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.

...

"Bag of Words"

- 16 × said
- 14 × McDonalds
- 12 × fat
- 11 × fries
- 8 × new
- 6 × company, french, nutrition
- 5 × food, oil, percent, reduce, taste, Tuesday
- …

Boolean Retrieval

- Users express queries as a Boolean expression
  - AND, OR, NOT
  - Can be arbitrarily nested
- Retrieval is based on the notion of sets
  - Any given query divides the collection into two sets: retrieved, not-retrieved
  - Pure Boolean systems do not define an ordering of the results

Representing Documents

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

Inverted Index

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

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

Boolean Retrieval

- To execute a Boolean query:
  - Build query syntax tree

(fox or dog) and quick → AND (quick, OR (fox, dog))

  - For each clause, look up postings

dog → 3 → 5
fox → 3 → 5 → 7

  - Traverse postings and apply Boolean operator

dog → 3 → 5
fox → 3 → 5 → 7
OR = union → 3 → 5 → 7

- Efficiency analysis
  - Postings traversal is linear (assuming sorted postings)
  - Start with shortest posting first

Extensions

- Implementing proximity operators
  - Store word offset in postings
- Handling term variations
  - Stem words: love, loving, loves … → lov

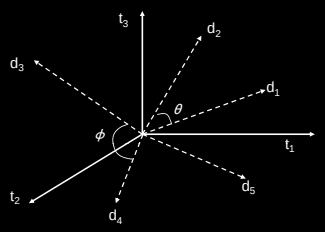
Strengths and Weaknesses

- Strengths
  - Precise, if you know the right strategies
  - Precise, if you have an idea of what you're looking for
  - Implementations are fast and efficient
- Weaknesses
  - Users must learn Boolean logic
  - Boolean logic insufficient to capture the richness of language
  - No control over size of result set: either too many hits or none
  - When do you stop reading? All documents in the result set are considered "equally good"
  - What about partial matches? Documents that "don't quite match" the query may be useful also

Ranked Retrieval

- Order documents by how likely they are to be relevant to the information need
  - Estimate relevance(q, d_i)
  - Sort documents by relevance
  - Display sorted results
- User model
  - Present hits one screen at a time, best results first
  - At any point, users can decide to stop looking
- How do we estimate relevance?
  - Assume document is relevant if it has a lot of query terms
  - Replace relevance (q, d_i) with sim(q, d_i)
  - Compute similarity of vector representations

Vector Space Model



Assumption: Documents that are "close together" in vector space "talk about" the same things

Therefore, retrieve documents based on how close the document is to the query (i.e., similarity ~ "closeness")

Similarity Metric

- How about $|d_1 - d_2|$?
- Instead of Euclidean distance, use "angle" between the vectors
  - It all boils down to the inner product (dot product) of vectors

Term Weighting

- Term weights consist of two components
  - Local: how important is the term in this document?
  - Global: how important is the term in the collection?
- Here's the intuition:
  - Terms that appear often in a document should get high weights
  - Terms that appear in many documents should get low weights
- How do we capture this mathematically?
  - Term frequency (local)
  - Inverse document frequency (global)

TF.IDF Term Weighting

$$w_{i,j} = \mathrm{tf}_{i,j} \cdot \log \frac{N}{n_i}$$

- $w_{i,j}$ **weight assigned to term *i* in document *j***
- $\mathrm{tf}_{i,j}$ **number of occurrence of term *i* in document *j***
- N **number of documents in entire collection**
- n_i **number of documents with term *i***

TF.IDF Example

	tf 1	2	3	4	idf
complicated			5	2	0.301
contaminated	4	1	3		0.125
fallout	5		4	3	0.125
information	6	3	3	2	0.000
interesting		1			0.602
nuclear	3		7		0.301
retrieval		6	1	4	0.125
siberia	2				0.602

complicated	0.301	3,5 4,2
contaminated	0.125	1,4 2,1 3,3
fallout	0.125	1,5 3,4 4,3
information	0.000	1,6 2,3 3,3 4,2
interesting	0.602	2,1
nuclear	0.301	1,3 3,7
retrieval	0.125	2,6 3,1 4,4
siberia	0.602	1,2

Sketch: Scoring Algorithm

- Initialize *accumulators* to hold document scores
- For each query term *t* in the user's query
  - Fetch *t*'s postings
  - For each document, $score_{doc}$ += $w_{t,d} \times w_{t,q}$
- Apply length normalization to the scores at end
- Return top *N* documents

MapReduce it?

- The indexing problem
  - Must be relatively fast, but need not be real time
  - For Web, incremental updates are important
- The retrieval problem
  - Must have sub-second response
  - For Web, only need relatively few results

Indexing: Performance Analysis

- Inverted indexing is fundamental to all IR models
- Fundamentally, a large sorting problem
  - Terms usually fit in memory
  - Postings usually don't
- How is it done on a single machine?
- How large is the inverted index?
  - Size of vocabulary
  - Size of postings

Vocabulary Size: Heaps' Law

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

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

Typically, *K* is between 10 and 100, β is between 0.4 and 0.6

When adding new documents, the system is likely to have seen most terms already… but the postings keep growing

Postings Size: Zipf's Law

- George Kingsley Zipf (1902-1950) observed the following relation between frequency and rank

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

f = frequency
r = rank
c = constant

- In other words:
  - A few elements occur very frequently
  - Many elements occur very infrequently
- Zipfian distributions:
  - English words
  - Library book checkout patterns
  - Website popularity (almost anything on the Web)

Word Frequency in English

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		

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

Does it fit Zipf's Law?

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		

The following shows *rf*×*1000*
r is the rank of word *w* in the sample
f is the frequency of word *w* in the sample

MapReduce: Index Construction

- Map over all documents
  - Emit *term* as key, (*docid*, *tf*) as value
  - Emit other information as necessary (e.g., term position)
- Reduce
  - Trivial: each value represents a posting!
  - Might want to sort the postings (e.g., by *docid* or *tf*)
- MapReduce does all the heavy lifting!

Query Execution

- MapReduce is meant for large-data batch processing
  - Not suitable for lots of real time operations requiring low latency
- The solution: "the secret sauce"
  - Most likely involves document partitioning
  - Lots of system engineering: e.g., caching, load balancing, etc.

MapReduce: Query Execution

- High-throughput batch query execution:
  - Instead of sequentially accumulating scores per query term:
  - Have mappers traverse postings in parallel, emitting partial score components
  - Reducers serve as the accumulators, summing contributions for each query term
- MapReduce does all the heavy lifting
  - Replace random access with sequential reads
  - Amortize over lots of queries
  - Examine multiple postings in parallel

Questions?