Information retrieval with elasticsearch

Alex Lambert (@alambert)
Co-founder & CTO
alex@spindle.com

SPINDLE

About

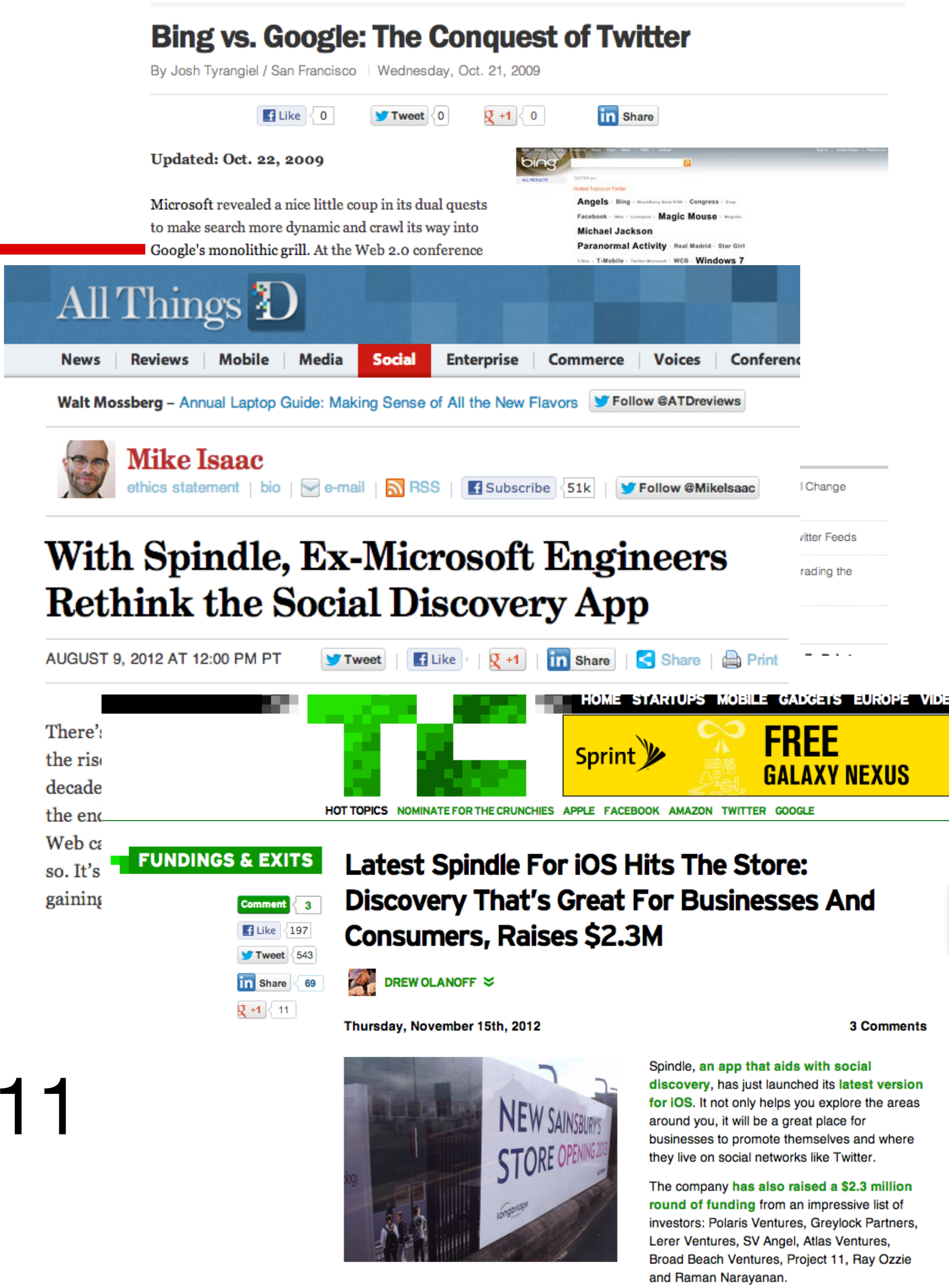
Previously:

- Microsoft Cambridge
- Bing Social Search
- FAST

Now: Spindle

Using ES since June 2011

(Almost) all data in ES



Bing vs. Google: The Conquest of Twitter

By Josh Tyrangiel / San Francisco | Wednesday, Oct. 21, 2009

Updated: Oct. 22, 2009

Microsoft revealed a nice little coup in its dual quests to make search more dynamic and crawl its way into Google's monolithic grill. At the Web 2.0 conference

All Things D

Walt Mossberg – Annual Laptop Guide: Making Sense of All the New Flavors

Mike Isaac

With Spindle, Ex-Microsoft Engineers Rethink the Social Discovery App

AUGUST 9, 2012 AT 12:00 PM PT

There'
the ris
decade
the enc
Web ca
so. It's
gaining

Latest Spindle For iOS Hits The Store: Discovery That's Great For Businesses And Consumers, Raises $2.3M

DREW OLANOFF

Thursday, November 15th, 2012

3 Comments

Spindle, an app that aids with social discovery, has just launched its latest version for iOS. It not only helps you explore the areas around you, it will be a great place for businesses to promote themselves and where they live on social networks like Twitter.

The company has also raised a $2.3 million round of funding from an impressive list of investors: Polaris Ventures, Greylock Partners, Lerer Ventures, SV Angel, Atlas Ventures, Broad Beach Ventures, Project 11, Ray Ozzie and Raman Narayanan.

Like 0
Tweet 0
+1 0
Share
bing
Angels · Bing · Congress
Facebook · Magic Mouse
Michael Jackson
Paranormal Activity · Real Madrid · Star Girl
T-Mobile · WCB · Windows 7
News
Reviews
Mobile
Media
Social
Enterprise
Commerce
Voices
Conferenc
Follow @ATDreviews
ethics statement | bio | e-mail | RSS | Subscribe 51k | Follow @MikeIsaac
l Change
itter Feeds
rading the
Tweet | Like | +1 | Share | Share | Print
HOME STARTUPS MOBILE GADGETS EUROPE VIDE
TC
Sprint
FREE GALAXY NEXUS
HOT TOPICS NOMINATE FOR THE CRUNCHIES APPLE FACEBOOK AMAZON TWITTER GOOGLE
FUNDINGS & EXITS
Comment 3
Like 197
Tweet 543
Share 69
+1 11
NEW SAINSBURY'S STORE OPENING 2013
longbridge

Why *this* talk?

"Hello world"

Config reference

Why *this* talk?

"Hello world"

The Economist
NOVEMBER 5TH–11TH 2011
Economist.com
The euro zone's Greek bombshell
Brazil's oil bonanza
Dealing with Japan's nuclear mess
India's maturing tech titans
The end of ageing?
The missing middle
The woeful gap in America's politics
ELECTION 2012

Config reference

Why *this* talk?

- To use elasticsearch effectively, you must understand the concepts of *information retrieval*
  - "Why can't I just do a regular expression search over my document content?"
  - "Why can't I find the phrase 'to be or not to be'?"
  - "Why was this document returned in my search results? It doesn't have the words from my query!"
  - "Why was this document scored higher than that document?"
- Content based on *Introduction to Information Retrieval* by Manning et al.

Databases, by question

- Relational databases
  - Adept at answering "What are the names of all employees in the finance department earning over $40,000 per year?"
  - Implementations include Oracle, MySQL
- Key-value stores
  - "What is user 123's profile image?"
  - Cassandra, Riak, Dynamo
- Graph databases
  - "Which friends of friends do Steve and Alex have in common?"
  - Neo4j, FlockDB

Information retrieval **engines**

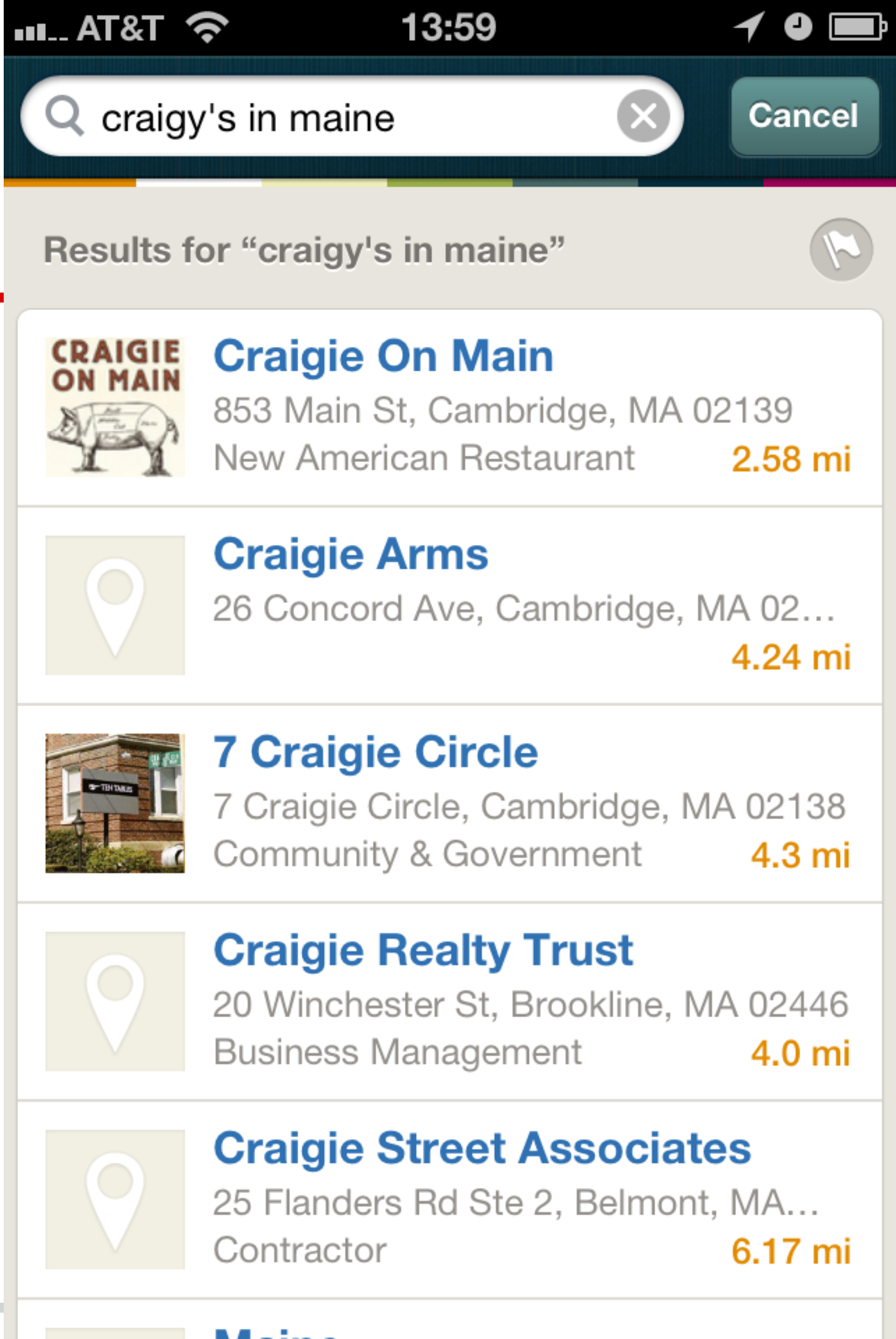
What is Google adept at answering?

> *Information retrieval (IR) is finding documents of an unstructured nature that satisfy an information need from within a large collection.* (Manning)

Implementations: Lucene (elasticsearch, Solr), FAST ESP, Endeca, Sphinx...

Finding a place

Information retrieval (IR) is finding documents of an unstructured nature that satisfy an information need from within a large collection.



Results for “craigy's in maine”

Craigie On Main
853 Main St, Cambridge, MA 02139
New American Restaurant 2.58 mi

Craigie Arms
26 Concord Ave, Cambridge, MA 02…
4.24 mi

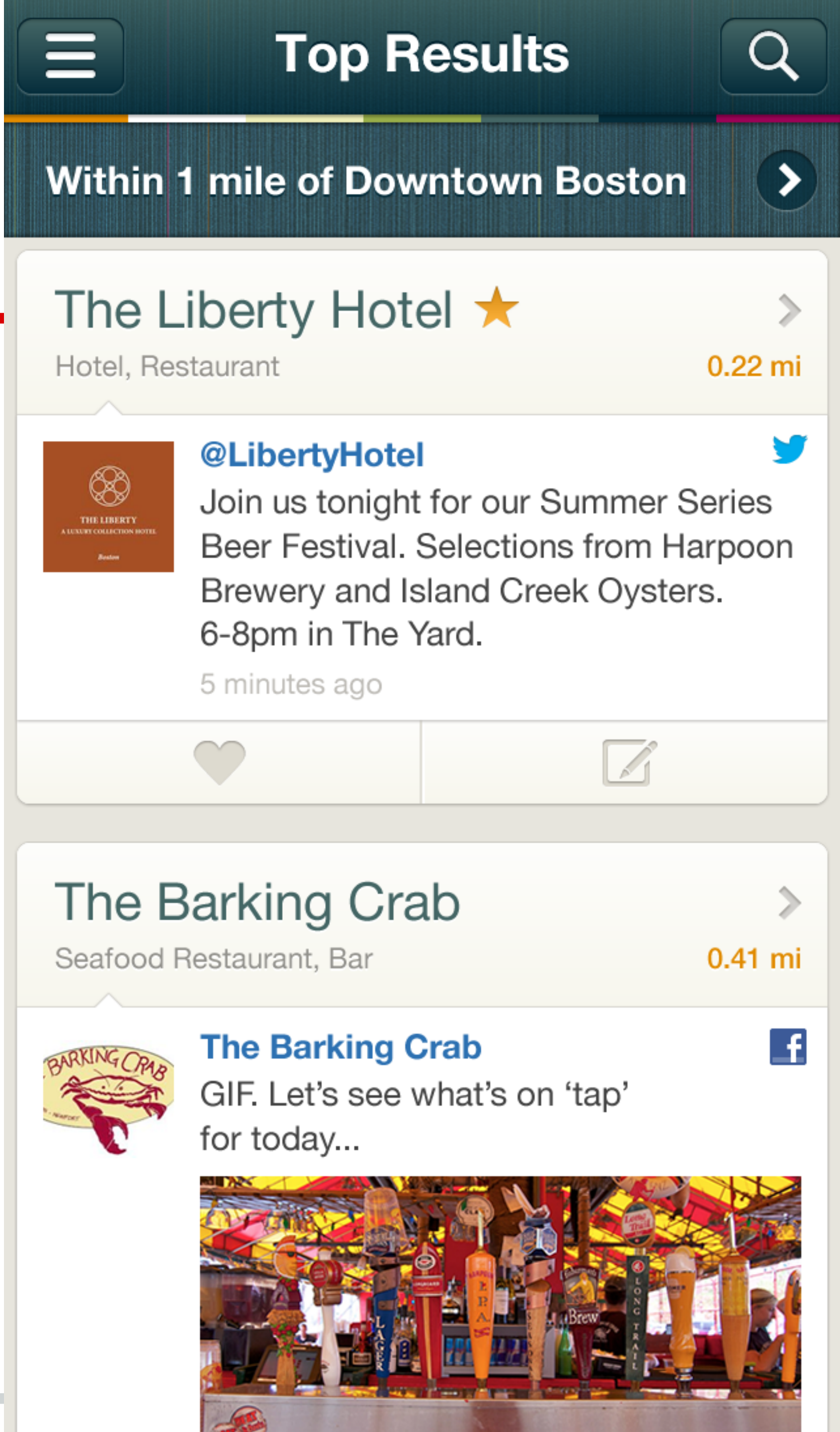
7 Craigie Circle
7 Craigie Circle, Cambridge, MA 02138
Community & Government 4.3 mi

Craigie Realty Trust
20 Winchester St, Brookline, MA 02446
Business Management 4.0 mi

Craigie Street Associates
25 Flanders Rd Ste 2, Belmont, MA…
Contractor 6.17 mi

Finding something to do tonight

Information retrieval (IR) is finding documents of an unstructured nature that satisfy an information need from within a large collection.



The Liberty Hotel ★
Hotel, Restaurant
0.22 mi

@LibertyHotel
Join us tonight for our Summer Series Beer Festival. Selections from Harpoon Brewery and Island Creek Oysters. 6-8pm in The Yard.
5 minutes ago

The Barking Crab
Seafood Restaurant, Bar
0.41 mi

The Barking Crab
GIF. Let's see what's on 'tap' for today...

The IR perspective

Is Harvard a valid result for the query "universities in Boston"?

Traditional: "the user knows *precisely* what he wants and how that's represented; I must do *exactly* what he says"

IR: "the user wants to find out about something and has given me a *hint* about what it is; I must be *helpful*"

The Simplest Query Language That Could Possibly Work, Proceedings of the 2nd INEX Workshop (2003)

What makes IR engines different?

Information need: which of Shakespeare's plays mention Brutus and Caesar but not Calpurnia?

MySQL query	*elasticsearch query*
select name from plays where (text like '%Brutus%') and (text like '%Caesar%') and not (text like '%Calpurnia%')	{"bool": {"must": [{"text": {"text": "Brutus"}}, {"text": {"text": "Caesar"}}],"must_not": [{"text": {"text": "Calpurnia"}}]}}

Why ES over SQL? IR engines provide:

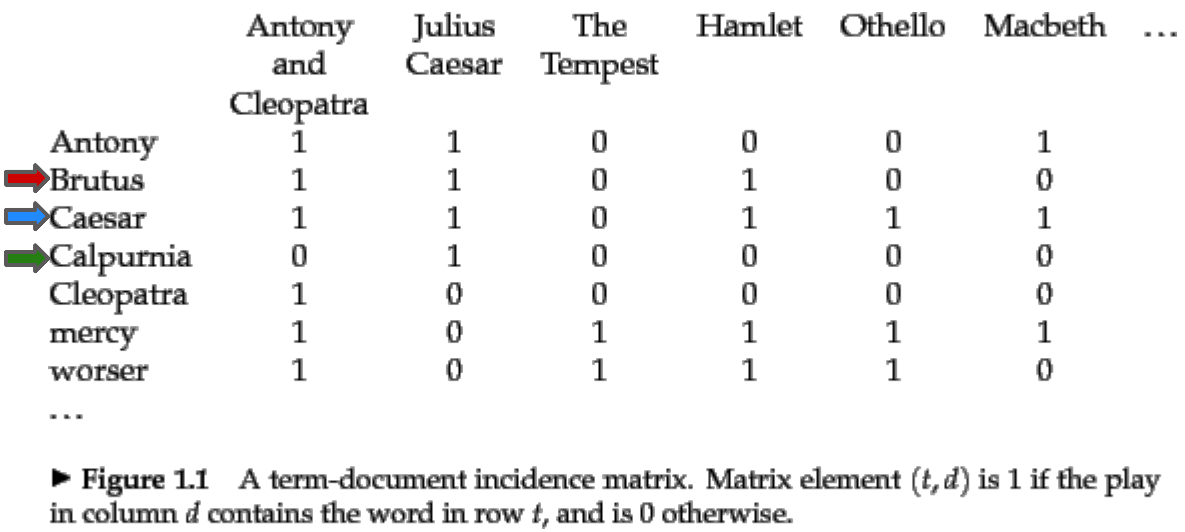
- Efficient access to huge collections: no table scans
- Flexible matching: "Romans" within 5 words of "countrymen"
- Ranked retrieval: best matches first

Efficient access to huge collections

Information need: which of Shakespeare's plays mention Brutus and Caesar but not Calpurnia?

We can *index* the plays. Collect all *terms* (words, for now). For each *document*, record whether it contains each possible *term*.

Query: Brutus AND Caesar AND NOT Calpurnia



	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	…
Antony	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	
…							

▶ **Figure 1.1** A term-document incidence matrix. Matrix element (t, d) is 1 if the play in column d contains the word in row t, and is 0 otherwise.

110100... AND 110111... AND NOT 010000... = 100100...

The Boolean retrieval model

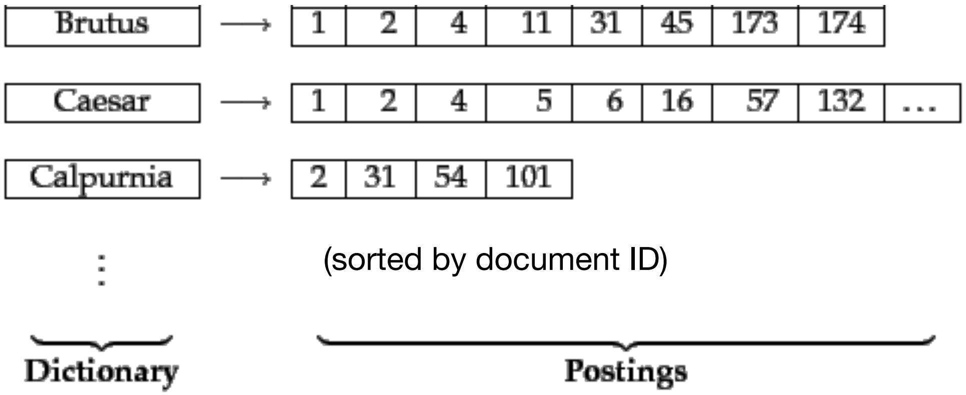
- A *document* is a set of *terms* (words, for now)
- *Query*: a Boolean expression of terms
  - Document *d* matches *t* iff term *t* is in *d*
  - Document *d* matches *t1 AND t2* iff *t1* and *t2* are in *d*
  - Document *d* matches *t1 OR t2* iff *t1* or *t2* are in *d*
  - Document *d* matches *NOT t* iff *d* does not contain *t*
- Match ≥ 2 of 3: *(a AND b) OR (b AND c) OR (a AND c)*

document	*query*	*matched?*
Friends, Romans, countrymen.	Romans AND Americans	no
The quick brown fox jumps over the lazy dog	(quick AND brown) AND (fox OR pig)	yes
Texas with a dollar sign	(texas AND dollar) OR (dollar AND sign) OR (texas AND sign)	yes

Implemented in elasticsearch as *filters* (fast & cacheable!)

Efficient access to huge collections

- Matrix has documentCount * termCount entries, most 0
- Assign each document a numeric ID, and store a *postings list*: for each *term*, store a list of documents that contain the term


Brutus → 1 2 4 11 31 45 173 174
Caesar → 1 2 4 5 6 16 57 132 …
Calpurnia → 2 31 54 101
⋮
Dictionary
Postings

(sorted by term)

(sorted by document ID)

How would we evaluate "Brutus AND Caesar AND NOT Calpurnia"?

"Why can't I just do a regular expression search over my document content?"

Lucene: IndexReader#terms(), IndexReader#termDocs(Term term)

The story so far

> *Information retrieval (IR) is finding documents of an unstructured nature that satisfy an information need from within a large collection.*

Users have *information needs* rendered as *queries*

The *Boolean model* provides simple, unranked matching

We can implement the Boolean model using a *postings list*

IR engines provide:

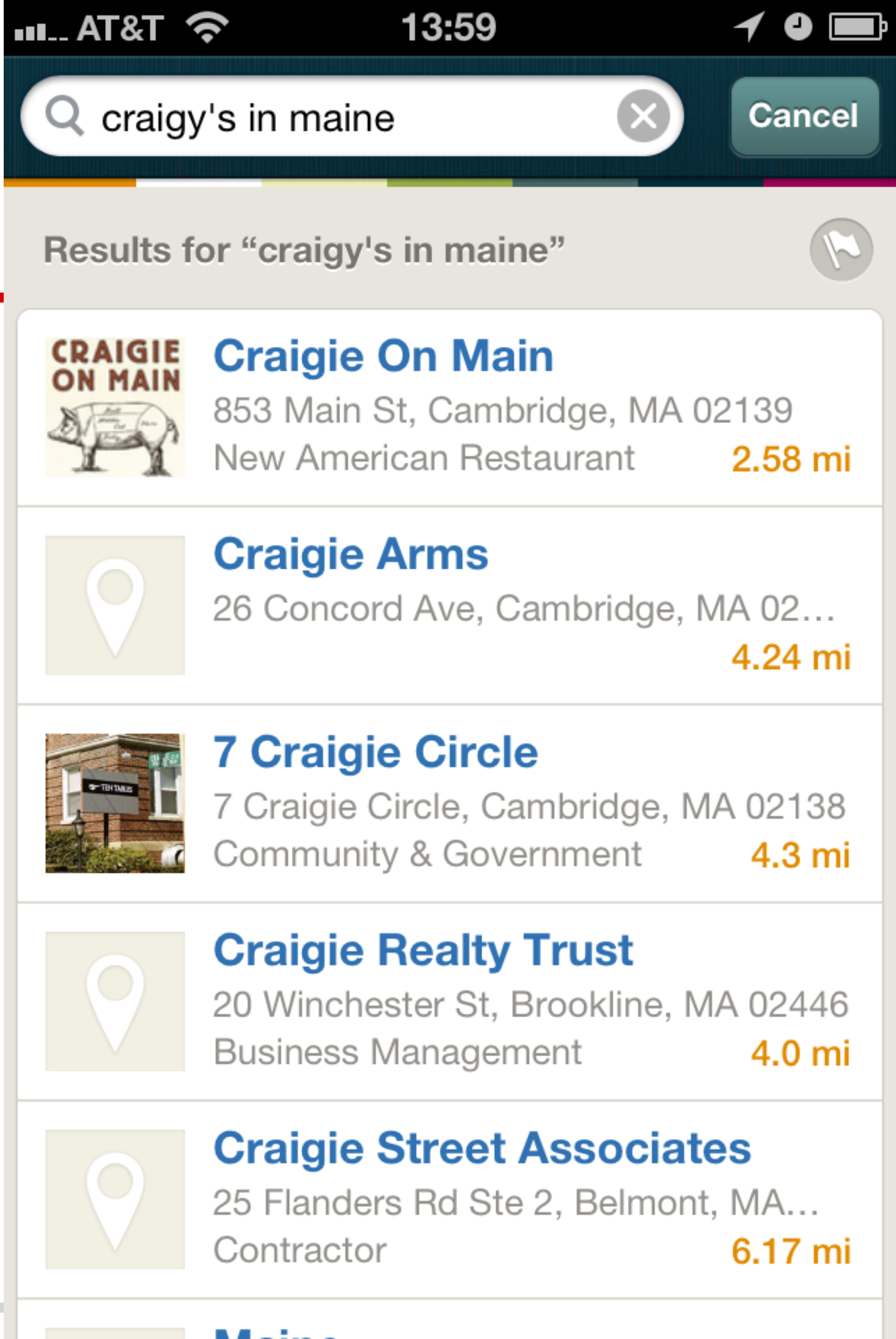
√ Efficient access to large collections

- Flexible matching
- Ranked retrieval

Flexible matching

How did this match?

(We said *terms* are words, and we're just matching terms, but this top result doesn't share any words with the query!)



craigy's in maine

Results for “craigy's in maine”

Craigie On Main
853 Main St, Cambridge, MA 02139
New American Restaurant 2.58 mi

Craigie Arms
26 Concord Ave, Cambridge, MA 02…
4.24 mi

7 Craigie Circle
7 Craigie Circle, Cambridge, MA 02138
Community & Government 4.3 mi

Craigie Realty Trust
20 Winchester St, Brookline, MA 02446
Business Management 4.0 mi

Craigie Street Associates
25 Flanders Rd Ste 2, Belmont, MA…
Contractor 6.17 mi

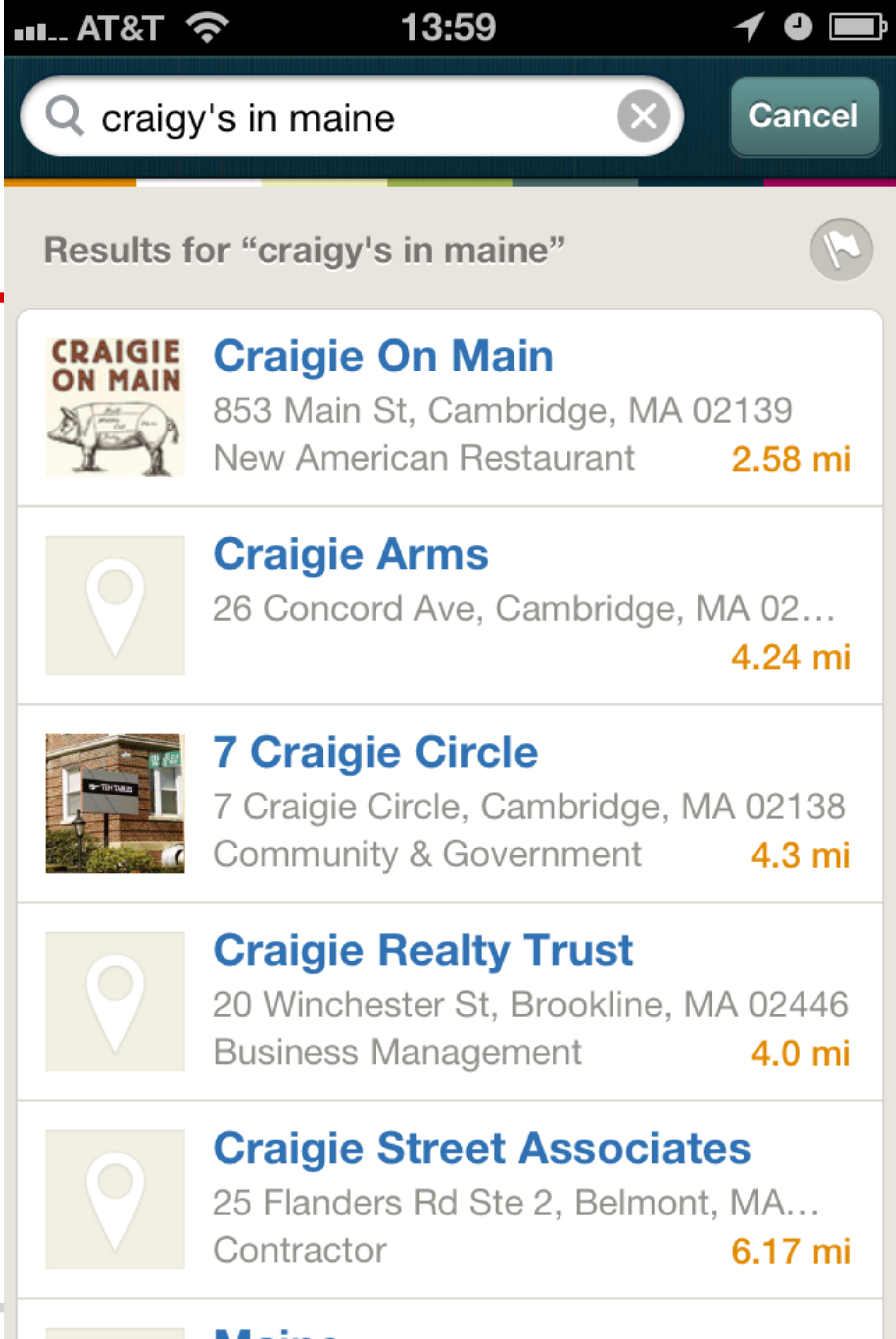
Maine

Flexible matching

How did this match?

string	*terms (after analysis)*
craigy's in maine	<craigi> <main>
Craigie On Main	<craigi> <main>

Goal: since we're just matching terms, use clever term choices to fine-tune matching



Results for “craigy's in maine”

Craigie On Main
853 Main St, Cambridge, MA 02139
New American Restaurant 2.58 mi

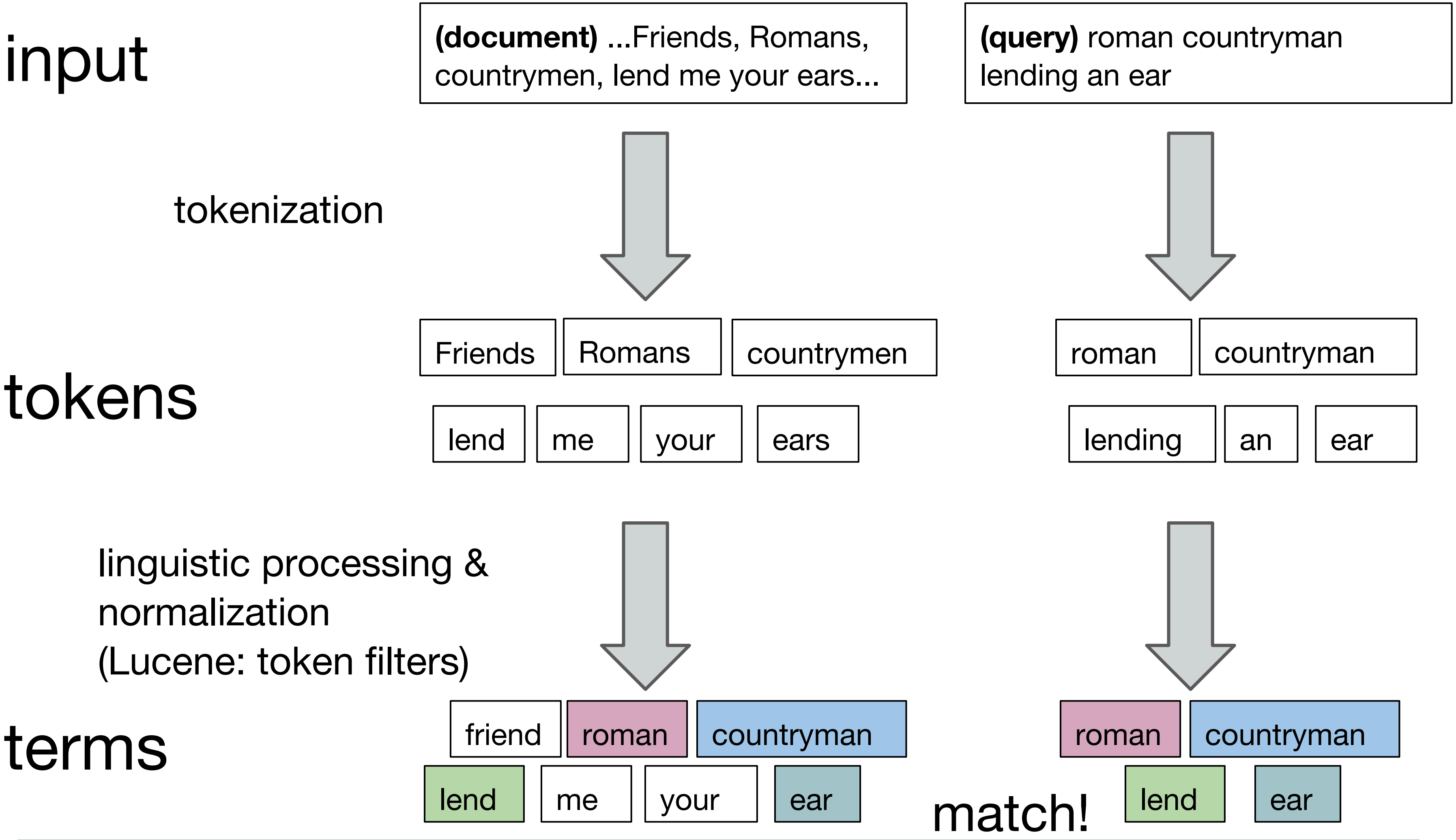
Craigie Arms
26 Concord Ave, Cambridge, MA 02…
4.24 mi

7 Craigie Circle
7 Craigie Circle, Cambridge, MA 02138
Community & Government 4.3 mi

Craigie Realty Trust
20 Winchester St, Brookline, MA 02446
Business Management 4.0 mi

Craigie Street Associates
25 Flanders Rd Ste 2, Belmont, MA…
Contractor 6.17 mi

Analysis: from strings to terms



input

(document) ...Friends, Romans, countrymen, lend me your ears...

(query) roman countryman lending an ear

tokenization

tokens

Friends Romans countrymen lend me your ears

roman countryman lending an ear

linguistic processing &
normalization
(Lucene: token filters)

terms

friend roman countryman lend me your ear

roman countryman lend ear

match!

Tokenizing English text

Goal: break a string into *tokens* (so we can later *filter* those tokens to create *terms*)

Mr. O'Neill thinks that Hewlett-Packard's www.autonomy.com acqusition "didn't go splendidly."

Whitespace tokenizer creates tokens from adjacent sequences of non-whitespace characters (try it)

Mr.	O'Neill	thinks	that	Hewlett-Packard's	www.autonomy.com	acquisition	"didn't
go	splendidly."						

Letter tokenizer divides text at non-letters (creating maximal strings of adjacent letters) (try it)

Mr	O	Neill	thinks	that	Hewlett	Packard	s
www	autonomy	com	acqusition	didn	t	go	splendidly

Standard tokenizer uses a grammar that implements Unicode Text Segmentation and recognizes URLs (try it)

Mr	O'Neill	thinks	that	Hewlett	Packard's	www.autonomy.com	acquisition
didn't	go	splendidly					

Token filters: from tokens to terms

Is tokenization sufficient? Unlikely: consider "iPhone 5", "IPhone 5", "Iphone 5", "iphone 5"

Goal: normalize tokens so that terms from document match terms from query

input token	*after lowercase*	*after English possessive*	*after whitespace trimming*	*after trim, lowercase, posessive*
< IBM's>	< ibm's>	< IBM>	<IBM's>	<ibm>

Token filters: stemming

- "bank holiday", "bank holidays", "banking holiday", and "banking holidays" all refer to the same concept
- A query for any of those phases should match a document with any of those phrases
- *Stemming* normalizes words by removing inflections

input tokens	*after Porter stemmer*
<bank> <holiday>	<bank> <holidai>
<banks> <holidays>	<bank> <holidai>
<banking> <holiday>	<bank> <holidai>
<banking> <holidays>	<bank> <holidai>

"Why was this document returned in my search results? It doesn't have the words from my query!"

Token filters: omitting stopwords

- Every document contains "a", "an", "of", "the"
- Generally not useful to store *stop words*, so we omit

input tokens	*after stop word filter*
<the> <library> <is> <closing> <at> <10>	<library> <closing> <10>
<the> <iphone> <5> <will> <be> <available> <at> <6> <in> <the> <evening>	<iphone> <5> <available> <6> <evening>
<to> <be> <or> <not> <to> <be>	

"Why can't I find the phrase 'to be or not to be'?"

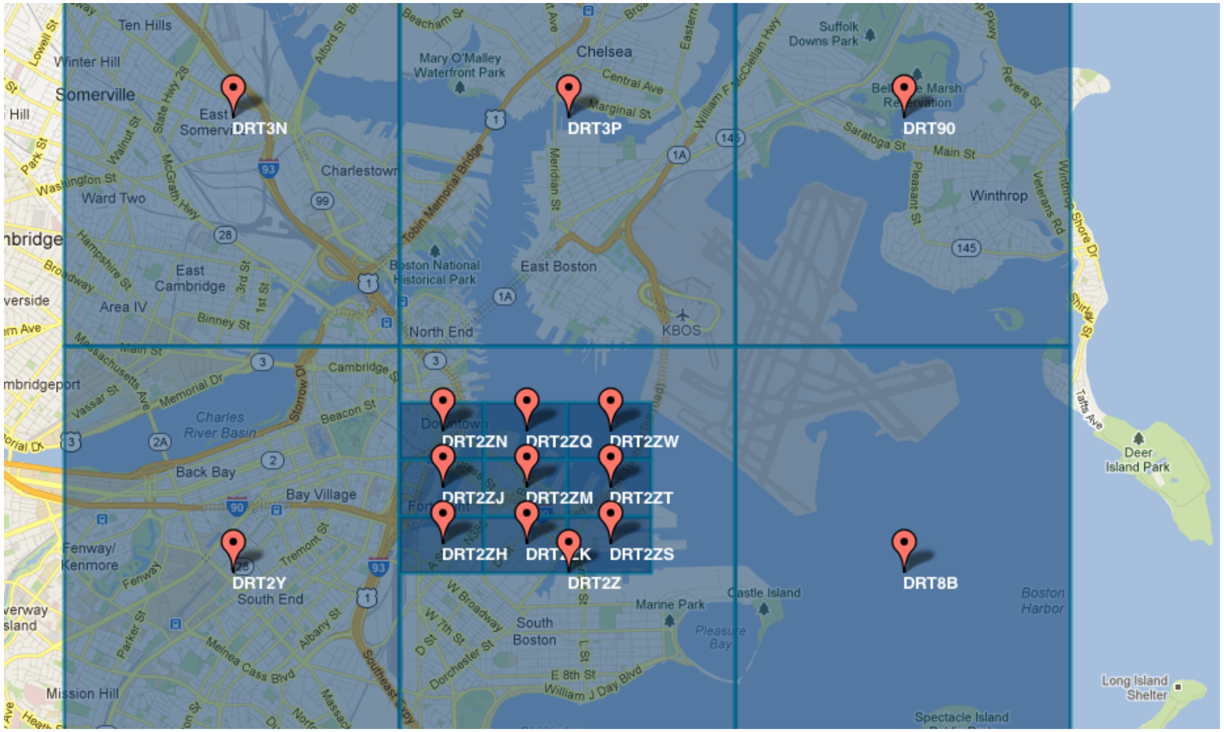
When all you have is a postings list, everything looks like term matching

input tokens	*after phonetic filter (nysiis)*
<Jeff> <Lupien>	<JAF> <LAPAN>
<Jeff> <Lupeen>	<JAF> <LAPAN>
<Jefe> <Lupean>	<JAF> <LAPAN>
<Jefe> <Loupeam>	<JAF> <LAPAN>

input tokens	*after shingle (no unigrams) filter*
<The> <quick> <brown> <fox> <jumped> <over> <the> <dog>	<The quick> <quick brown> <brown fox> <fox jumped> <jumped over> <over the> <the dog>

input string	*after keyword analysis (or not_analyzed)*
The quick brown fox jumps over the lazy dog.	<The quick brown fox jumps over the lazy dog.>

Geospatial search: geohash terms



Wikipedia: Geohash is "a hierarchical spatial data structure which subdivides space into buckets of grid shape."

elasticsearch: geo_shape filter

The story so far

Analysis converts *documents* to *terms*:

- *tokenizers* map a string to a sequence of *tokens*
- *token filters* transform a sequence of tokens

Thinking with terms: searches just traverse the postings list, so cast your problems as term matching

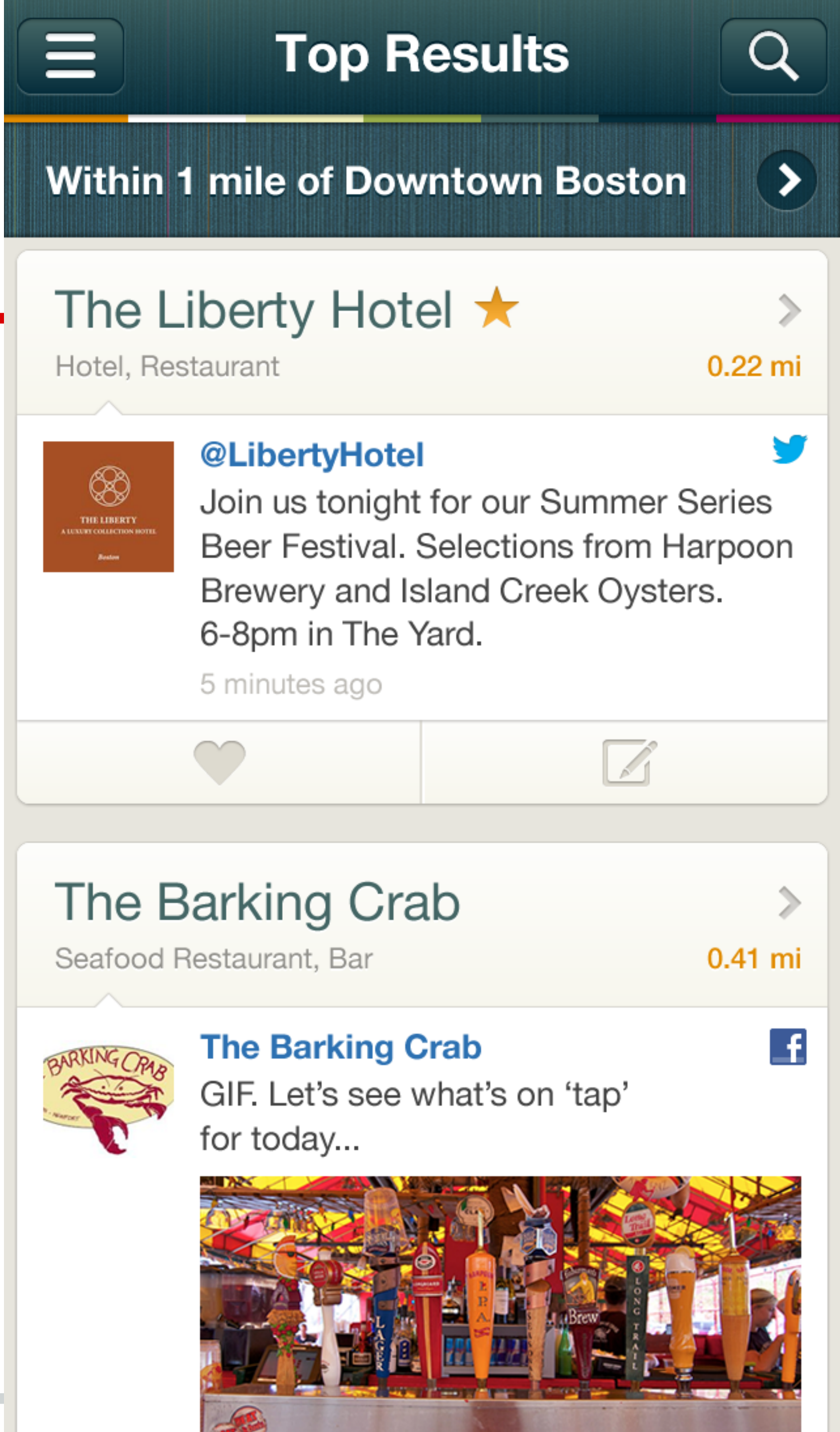
IR engines provide:

√ Efficient access to large collections

√ Flexible matching

- Ranked retrieval

Ranked retrieval



Top Results

Within 1 mile of Downtown Boston

The Liberty Hotel ★

Hotel, Restaurant

0.22 mi

@LibertyHotel

Join us tonight for our Summer Series Beer Festival. Selections from Harpoon Brewery and Island Creek Oysters. 6-8pm in The Yard.

5 minutes ago

The Barking Crab

Seafood Restaurant, Bar

0.41 mi

The Barking Crab

GIF. Let's see what's on 'tap' for today...

Ranked retrieval

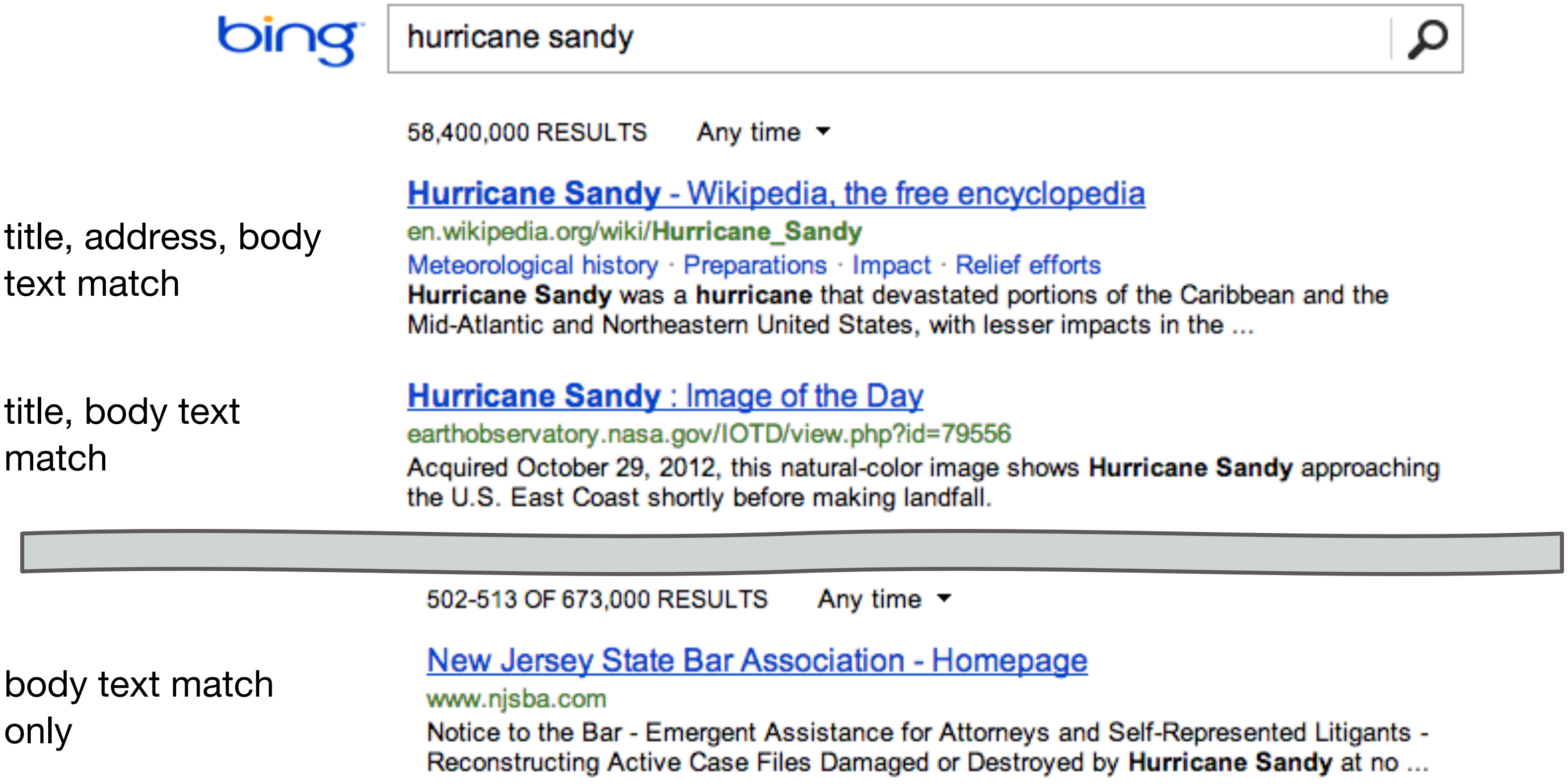
In the Boolean model, a document is either *relevant* or *not relevant* to a particular query

It's (usually) impractical to look through all relevant results

Ranked retrieval: for each relevant document, compute a score with respect to the query, and then sort documents based on that score

Ranked retrieval: field weights

Intuition: a document with query terms in its title (or URL) is more relevant for that query



hurricane sandy

58,400,000 RESULTS Any time

title, address, body text match

Hurricane Sandy - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Hurricane_Sandy**
Meteorological history · Preparations · Impact · Relief efforts
Hurricane Sandy was a **hurricane** that devastated portions of the Caribbean and the Mid-Atlantic and Northeastern United States, with lesser impacts in the ...

title, body text match

Hurricane Sandy : Image of the Day
earthobservatory.nasa.gov/IOTD/view.php?id=79556
Acquired October 29, 2012, this natural-color image shows **Hurricane Sandy** approaching the U.S. East Coast shortly before making landfall.

502-513 OF 673,000 RESULTS Any time

body text match only

New Jersey State Bar Association - Homepage
www.njsba.com
Notice to the Bar - Emergent Assistance for Attorneys and Self-Represented Litigants - Reconstructing Active Case Files Damaged or Destroyed by **Hurricane Sandy** at no ...

Ranked retrieval: field weights

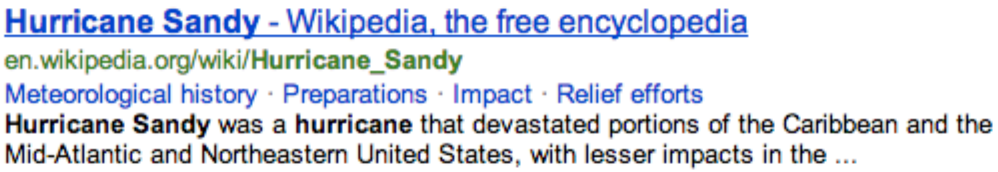
Assign weights to fields, use the Boolean model, and then score each document:

$$score(q, d) = matched(q, d, field1) * weight(field1) + matched(q, d, field1) * weight(field2) + \ldots$$

where *matched(q, d, f)* is 1 if the document *d* matched the query *q* in field *f* and 0 otherwise

query: hurricane AND sandy

field	*weight*
title	0.5
url	0.1
body	0.4



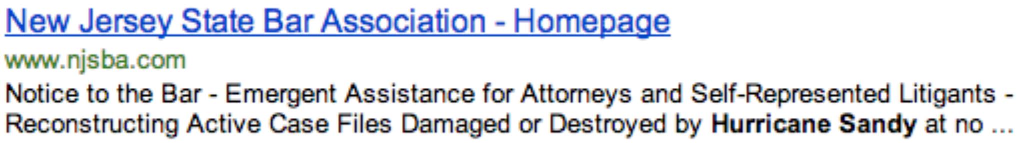
Hurricane Sandy - Wikipedia, the free encyclopedia

en.wikipedia.org/wiki/**Hurricane_Sandy**

Meteorological history · Preparations · Impact · Relief efforts

Hurricane Sandy was a **hurricane** that devastated portions of the Caribbean and the Mid-Atlantic and Northeastern United States, with lesser impacts in the ...

$score = 1 * 0.5 + 1 * 0.1 + 1 * 0.4 = 1$



New Jersey State Bar Association - Homepage

www.njsba.com

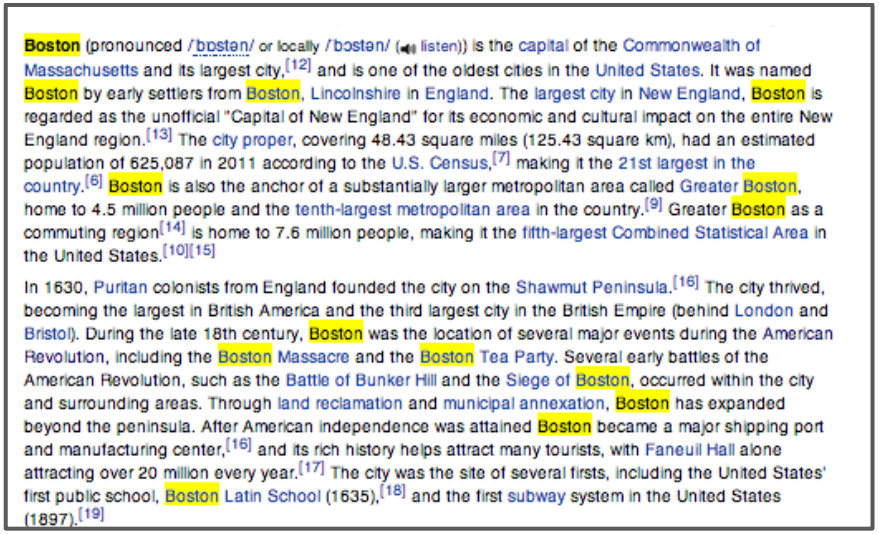
Notice to the Bar - Emergent Assistance for Attorneys and Self-Represented Litigants - Reconstructing Active Case Files Damaged or Destroyed by **Hurricane Sandy** at no ...

$score = 0 * 0.5 + 0 * 0.1 + 1 * 0.4 = 0.4$

Exercise: Can we implement field-specific matching using a single postings list?

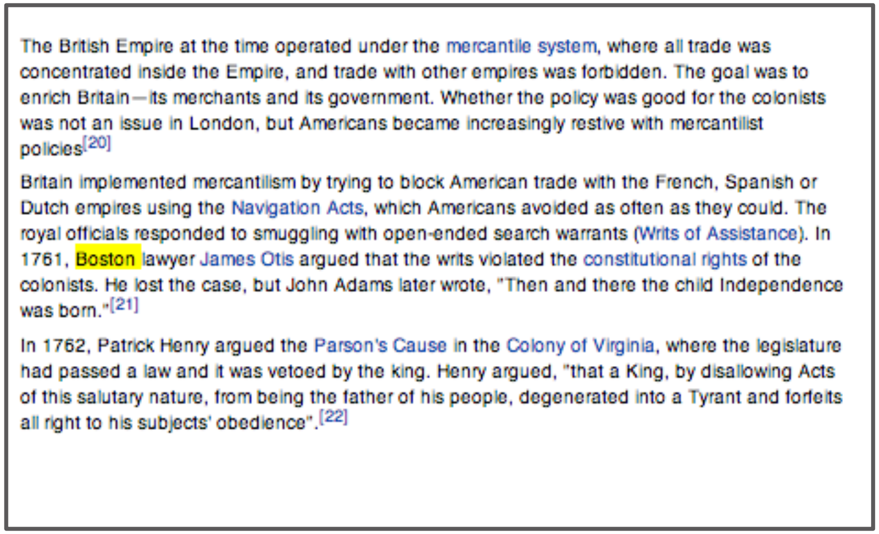
Ranked retrieval: term frequency

Both documents match the query "Boston" – but the left is more relevant



Boston (pronounced /bɒstən/ or locally /bɔstən/ (listen)) is the capital of the Commonwealth of Massachusetts and its largest city,[12] and is one of the oldest cities in the United States. It was named Boston by early settlers from Boston, Lincolnshire in England. The largest city in New England, Boston is regarded as the unofficial "Capital of New England" for its economic and cultural impact on the entire New England region.[13] The city proper, covering 48.43 square miles (125.43 square km), had an estimated population of 625,087 in 2011 according to the U.S. Census,[7] making it the 21st largest in the country.[6] Boston is also the anchor of a substantially larger metropolitan area called Greater Boston, home to 4.5 million people and the tenth-largest metropolitan area in the country.[9] Greater Boston as a commuting region[14] is home to 7.6 million people, making it the fifth-largest Combined Statistical Area in the United States.[10][15]

In 1630, Puritan colonists from England founded the city on the Shawmut Peninsula.[16] The city thrived, becoming the largest in British America and the third largest city in the British Empire (behind London and Bristol). During the late 18th century, Boston was the location of several major events during the American Revolution, including the Boston Massacre and the Boston Tea Party. Several early battles of the American Revolution, such as the Battle of Bunker Hill and the Siege of Boston, occurred within the city and surrounding areas. Through land reclamation and municipal annexation, Boston has expanded beyond the peninsula. After American independence was attained Boston became a major shipping port and manufacturing center,[16] and its rich history helps attract many tourists, with Faneuil Hall alone attracting over 20 million every year.[17] The city was the site of several firsts, including the United States' first public school, Boston Latin School (1635),[18] and the first subway system in the United States (1897).[19]



The British Empire at the time operated under the mercantile system, where all trade was concentrated inside the Empire, and trade with other empires was forbidden. The goal was to enrich Britain—its merchants and its government. Whether the policy was good for the colonists was not an issue in London, but Americans became increasingly restive with mercantilist policies[20]

Britain implemented mercantilism by trying to block American trade with the French, Spanish or Dutch empires using the Navigation Acts, which Americans avoided as often as they could. The royal officials responded to smuggling with open-ended search warrants (Writs of Assistance). In 1761, Boston lawyer James Otis argued that the writs violated the constitutional rights of the colonists. He lost the case, but John Adams later wrote, "Then and there the child Independence was born."[21]

In 1762, Patrick Henry argued the Parson's Cause in the Colony of Virginia, where the legislature had passed a law and it was vetoed by the king. Henry argued, "that a King, by disallowing Acts of this salutary nature, from being the father of his people, degenerated into a Tyrant and forfeits all right to his subjects' obedience".[22]

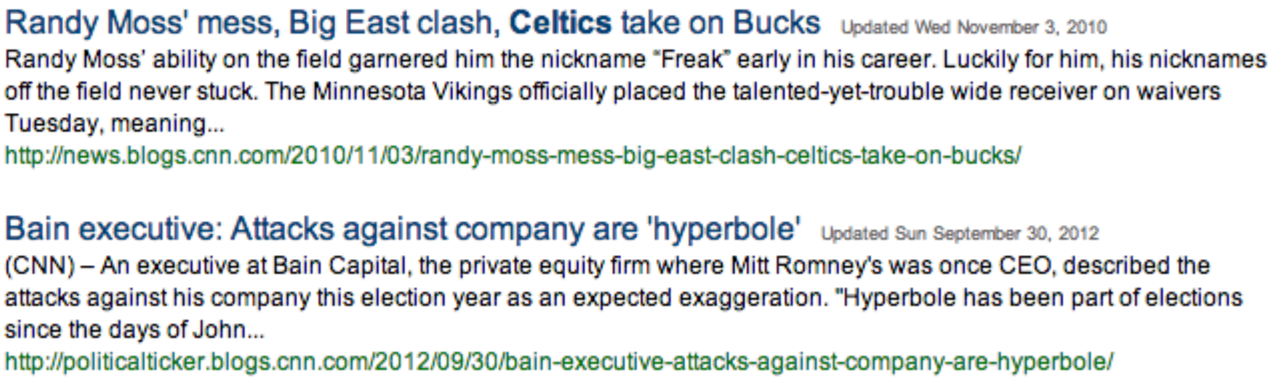
Intuition: documents that mention query terms more often are more relevant for that query

Term frequency: *tf(t, d)*: the number of occurrences of the term *t* in the document *d*.

Ranked retrieval: document frequency

Many cnn.com articles contains "Boston" (6,956); fewer (356) contain "Celtics". For the query "Boston Celtics", a document that has only "Celtics" is more relevant than a document that has only "Boston".

Intuition: documents with rare terms from the query are more relevant



Randy Moss' mess, Big East clash, **Celtics** take on Bucks Updated Wed November 3, 2010

Randy Moss' ability on the field garnered him the nickname "Freak" early in his career. Luckily for him, his nicknames off the field never stuck. The Minnesota Vikings officially placed the talented-yet-trouble wide receiver on waivers Tuesday, meaning...

http://news.blogs.cnn.com/2010/11/03/randy-moss-mess-big-east-clash-celtics-take-on-bucks/

Bain executive: Attacks against company are 'hyperbole' Updated Sun September 30, 2012

(CNN) – An executive at Bain Capital, the private equity firm where Mitt Romney's was once CEO, described the attacks against his company this election year as an expected exaggeration. "Hyperbole has been part of elections since the days of John...

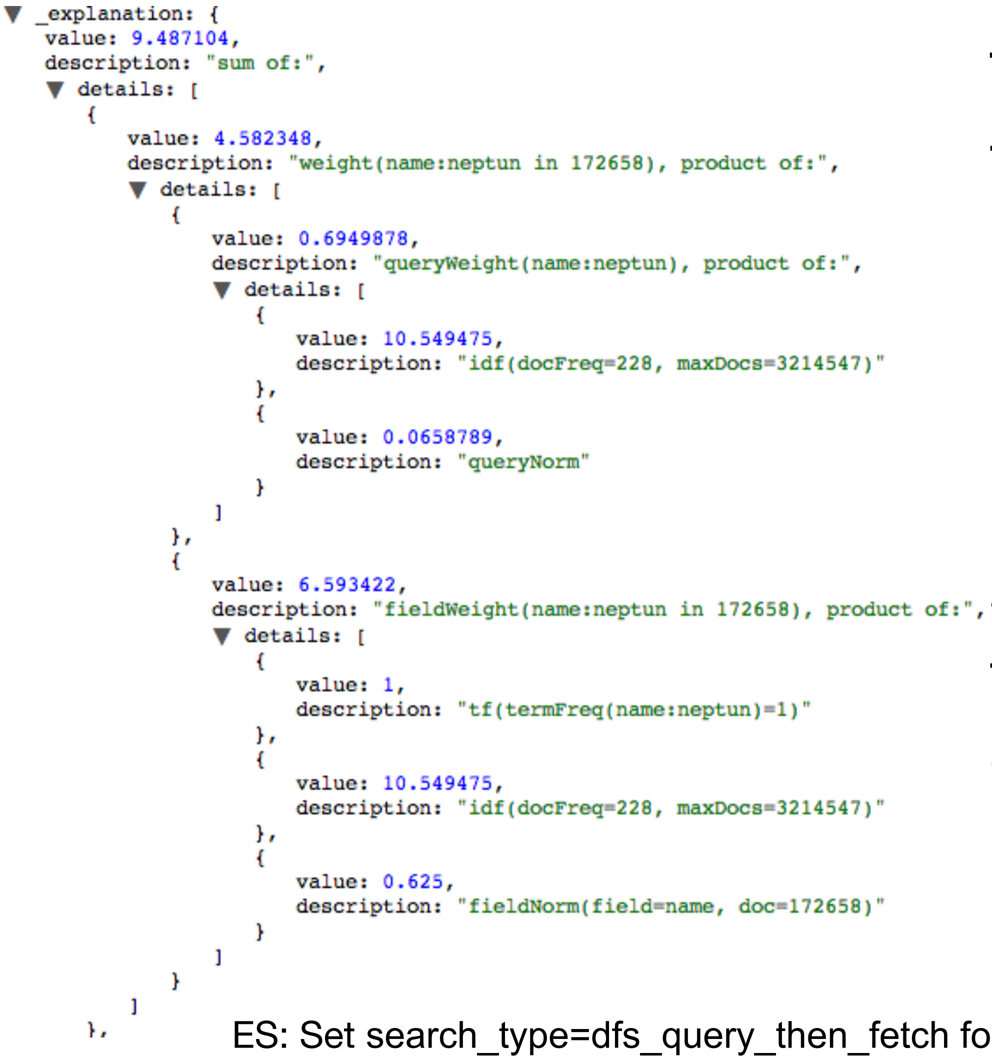
http://politicalticker.blogs.cnn.com/2012/09/30/bain-executive-attacks-against-company-are-hyperbole/

Document frequency: *df(t)*: the number of documents that contain the term *t*

Ranked retrieval: tf-idf

- Term frequency, *tf(t, d)*, suggests the importance of a term *t* within a particular document *d*
- Document frequency, *df(t)*, compensates for terms that appear too often throughout the collection
  - Define inverse document frequency: *idf(t) = log(N / df(t))*, where *N* is the number of documents in the collection
  - *idf(t)* is high for rare terms, low for frequent terms
- For term *t* in document *d*, *tf-idf(d, t) = tf(t, d) * idf(t)*
  - Highest if *t* occurs frequently in *d* and *t* appears in few documents
  - Lower if *t* occurs rarely in *d* or *t* appears in many documents
  - Lowest when *t* is in almost all documents
- Then: *score(q, d) = tf-idf(term1, d) + tf-idf(term2, d) + ...*

Scoring explanations



```
▼ _explanation: {
    value: 9.487104,
    description: "sum of:",
  ▼ details: [
      {
          value: 4.582348,
          description: "weight(name:neptun in 172658), product of:",
        ▼ details: [
            {
                value: 0.6949878,
                description: "queryWeight(name:neptun), product of:",
              ▼ details: [
                  {
                      value: 10.549475,
                      description: "idf(docFreq=228, maxDocs=3214547)"
                  },
                  {
                      value: 0.0658789,
                      description: "queryNorm"
                  }
                ]
            },
            {
                value: 6.593422,
                description: "fieldWeight(name:neptun in 172658), product of:",
              ▼ details: [
                  {
                      value: 1,
                      description: "tf(termFreq(name:neptun)=1)"
                  },
                  {
                      value: 10.549475,
                      description: "idf(docFreq=228, maxDocs=3214547)"
                  },
                  {
                      value: 0.625,
                      description: "fieldNorm(field=name, doc=172658)"
                  }
                ]
            }
          ]
      },
```

To enable, set "explain" to *true* in the search request

"neptun" appeared once in the document and in 228 total documents:

tf=1,

df=228,

idf=1+log(3214547/(228+1))

ES: Set search_type=dfs_query_then_fetch for accurate distributed tf-idf computation

The vector space model

Let *T* be the set of all terms. We can represent each document *d* as a vector *V(d)* having | *T* | components:

$$V(d) = (tf\text{-}idf(d, term1), tf\text{-}idf(d, term2), \ldots)$$

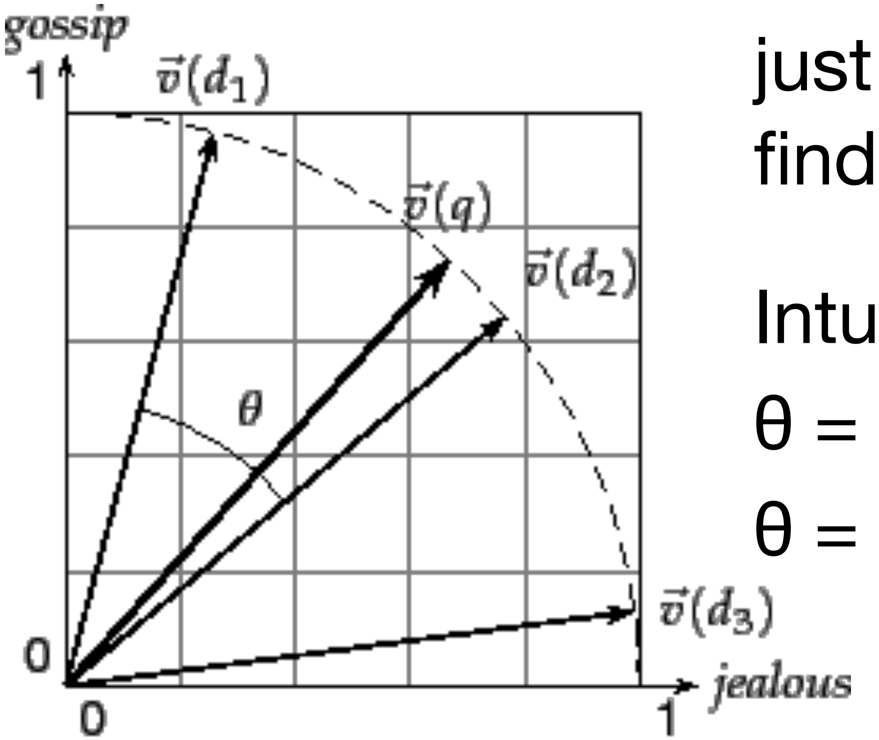
(let *tf-idf(d, t)* be 0 if *d* does not contain *t*)

as a unit vector: $v(d) = V(d) \,/\, \|V(d)\|$

(normalizes for document length)

We can use this approach for queries, also

The vector space model: cosine similarity



A document and a query are just (unit) vectors, so we can find the angle between them

Intuition:

θ = 0° if identical

θ = 90° if completely dissimilar

cos(0°) = 1
cos(30°) = 0.866
cos(45°) = 0.707...
cos(60°) = 0.5
cos(90°) = 0

$$\textit{similarity}(d, q) = \|v(d)\|\ \|v(q)\| \cos\theta = \cos\theta$$
$$= v(d) \bullet v(q)$$

The story so far

Field weighting is a basic *ranked retrieval* approach

Term frequency: *tf(t, d)*: the number of occurrences of the term *t* in the document *d*

Document frequency: *df(t)*: the number of documents that contain the term *t*

tf-idf weighting combines these measures

IR engines provide:

√ Efficient access to large collections

√ Flexible matching

√ Ranked retrieval

Further reading

topic	*elasticsearch implementation*
Boolean model of information retrieval	Filters in the Query DSL
Flexible matching	Analysis Recommended reading: Lucene in Action, 2nd Edition, chapter 4
Vector space model of information retrieval	Scoring overview Similarity details Scoring explanations
Geospatial search	geo_shape query David Smiley's presentation
Numeric range queries	NumericRangeQuery (clever!)

Recommended reading: *Introduction to Information Retrieval* by Manning et al., *Taming Text* by Ingersoll et al.

Thank you

spindle.com/talks

spindle.com/jobs

alex@spindle.com, @alambert