Annotating, indexing and searching the Web of entities and relationships

Soumen Chakrabarti

http://soumen.in/doc/CSAW/

Motivation

- Fair to say that few areas of computer science have had as much impact on our lives in the last 15 years as the Internet, the Web, search engines, e-commerce, and social media
- Rapid rise of some of the largest and most accomplished companies on earth
- Draws heavily from data structures, algorithms, databases, probability and statistics, machine learning, parallel computing, economics and game theory, social sciences, …(“whatever works”)

Highlights

- **1970** Arthur C. Clarke predicts that satellites would "bring the accumulated knowledge of the world to your fingertips" using a console combining the functionality of the photocopier, telephone, television and a small computer, allowing data transfer and video conferencing around the globe.
- **1990** Tim Berners-Lee &co build first prototype at CERN
- **1992** Mosaic Web brower developed
- **1994** Netscape, Yahoo! founded
- **1995** Alta Vista launched by Digital Equipment Corp
- **1996** Backrub, later called Google, founded
- **1998** Goto.com introduces paid search and ads
- **2003** Google launches AdSense
- **2004** Facebook founded
- **2005** YouTube founded
- **2006** Twitter founded
- **2009** Bing launched by Microsoft

Infrastructure
Crawling
Indexing
PageRank
De-spamming
Text mining
Auctions
Social media
Personalization
Collaborative filtering
Query+click log mining
Local search

Where's the action now?

"Over the next few months, [Google] will also present more facts and direct answers to queries"

"will better match search queries with a database containing hundreds of millions of "entities"—people, places and things—which the company has quietly amassed in the past two years."

"Things, not strings" — Google

"Web of Objects" — Yahoo

--- Amit Singhal (Google) to Wall Street Journal, March 2012

This tutorial

- Basic search
  - Text indexing
  - Query execution
  - Relevance ranking
- Augmenting text with entity annotations
  - Local and collective disambiguation
  - Data structures
- Entity search
  - Collecting supporting snippets
  - Scoring candidate entities

Basic search

Abstract word and document model

- Define a **word** as any non-empty maximal sequence of characters from a restricted set
  - E.g. [a-zA-Z0-9] or [^ \t\r\n] etc.
  - Some languages do not have easy delimiters
- Set of all words found over all documents in corpus is the corpus **vocabulary**
- Can arbitrarily order words and number them
- Henceforth, word ←→ integer word ID
- First cut: document = **set** of word IDs
- Later, **bag** (multiset), finally, **sequence**

Toy corpus with two documents

d_1 my care is loss of care with old care done

d_2 your care is gain of care with new care won

Corpus

$d_1 = \{6, 1, 4, 5, 8, 1, 10, 9, 1, 2\}$

$d_2 = \{12, 1, 4, 3, 8, 1, 10, 7, 1, 11\}$

Document representation as sequence

$d_1 = \{1, 2, 4, 5, 6, 8, 9, 10\}$

$d_2 = \{1, 3, 4, 7, 8, 10, 11, 12\}$

Document representation as set

Vocabulary

1	care
2	done
3	gain
4	is
5	loss
6	my
7	new
8	of
9	old
10	with
11	won
12	your

Boolean queries

- Examples with set representation:
  - Document/s containing “care” and “done”
  - Document/s containing “care” but not “old”
- Examples with sequence representation:
  - Documents containing phrase “new care”
  - Documents where “care” and “done” occur within 3 tokens of each other
- Can build more complex clauses
  - Has phrase “care with” but not “old”

Toy corpus as binary matrix

w→	1	2	3	4	5	6	7	8	9	10	11	12
d_1	1	1	0	1	1	1	0	1	1	1	0	0
d_2	1	0	1	1	0	0	1	1	0	1	1	1

- Very sparse, most entries zero
  - 10^9 Web pages, each has 100 distinct words
  - Corpus vocabulary may be larger than 10^6
- When reading corpus, docs arrive one by one
- I.e., matrix is revealed a **row** at a time
- To run Boolean query, must probe by **columns**

Index = transpose + compress columns

Terms→

↓Documents

	Anthony	Brutus	Caesar	Calpurnia	Cleopatra	mercy	worser
Antony and Cleopatra	1	1	1	0	1	1	1
Julius Caesar	1	1	1	1	0	0	0
The Tempest	0	0	0	0	0	1	1
Hamlet	0	1	1	0	0	1	1
Othello	0	0	1	0	0	1	1
Macbeth	1	0	1	0	0	1	0

- Anthony $\wedge$ mercy → 110001$\wedge$101111=100001
- Sparse $\Rightarrow$ rather record doc IDs than long bit map
- Anthony→(1,2,6) mercy→(1,3,4,5,6) result=(1,6)

Variable length gap codes

- Anthony→(1,2,6) mercy→(1,3,4,5,6)
  - Doc IDs in increasing order to make merge easy
- Instead record gaps
  - Anthony→(1,1,4) mercy→(1,2,1,1,1)
- Can decompress cheaply on the fly
- Useful only if we avoid fixed size integers
- Will briefly discuss two approaches
  - (Elias) gamma codes and extensions
  - Word aligned code with continuation bits
- Space saved vs. decompression speed

Gamma code

- We can compress better with bit-level codes
  - The Gamma code is the best known of these.
- Represent a gap G as a pair length and offset
- offset is G in binary, with the leading bit cut off
  - For example 13 → 1101 → 101
- length is the length of offset
  - For 13 (offset 101), this is 3.
- We encode length with unary code: 1110.
- Gamma code of 13 is the concatenation of length and offset: 1110101
- Binary to other radix → Golomb(-Rice) code

Gamma code examples

number	length	offset	γ-code
0			none
1	0		0
2	10	0	10,0
3	10	1	10,1
4	110	00	110,00
9	1110	001	1110,001
13	1110	101	1110,101
24	11110	1000	11110,1000
511	111111110	11111111	111111110,11111111
1025	11111111110	0000000001	11111111110,0000000001

Variable Byte (VB) codes

- For a gap value *G*, we want to use close to the fewest bytes needed to hold $\log_2 G$ bits
- Begin with one byte to store *G* and dedicate 1 bit in it to be a continuation bit *c*
- If $G \leq 127$, binary-encode it in the 7 available bits and set $c = 1$
- Else encode *G*'s lower-order 7 bits and then use additional bytes to encode the higher order bits using the same algorithm
- At the end set the continuation bit of the last byte to 1 ($c = 1$) – and for the other bytes $c = 0$.

Example

docIDs	824	829	215406
gaps		5	214577
VB code	00000110 10111000	10000101	00001101 00001100 10110001

Postings stored as the byte concatenation
000001101011100010000101000011010000110010110001

Key property: VB-encoded postings are uniquely prefix-decodable.

For a small gap (5), VB uses a □hole byte.

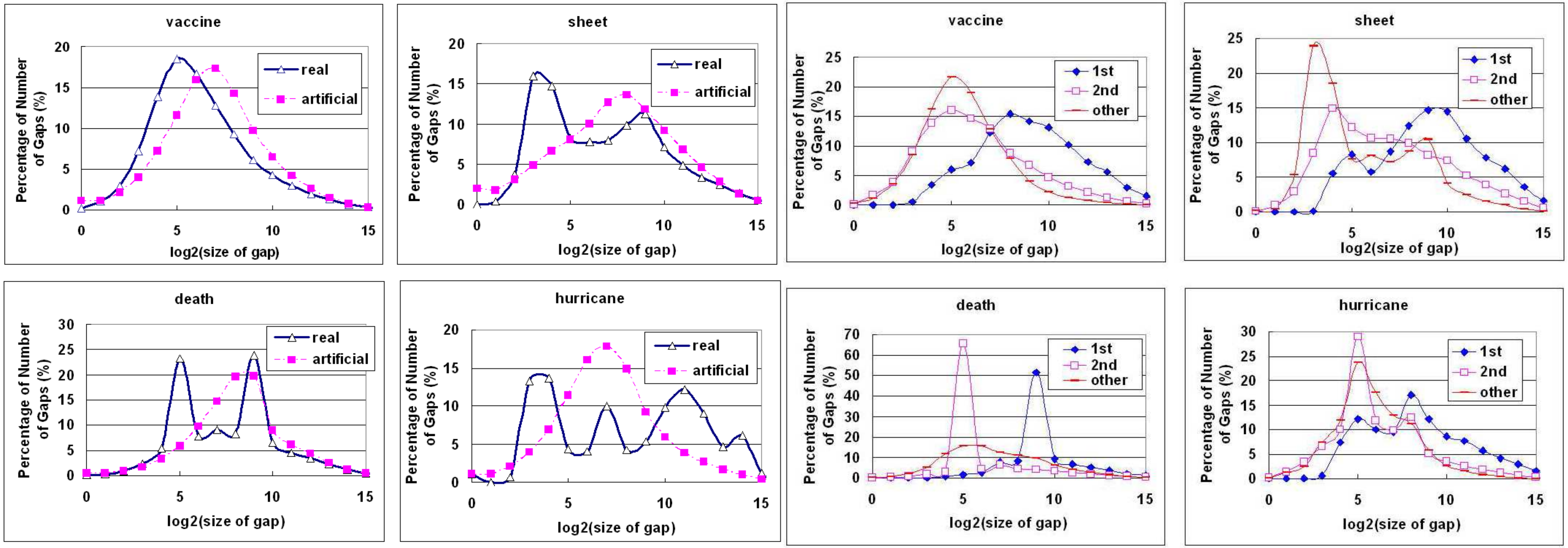
Sample index sizes (RCV1)

Data structure	Size in MB
dictionary, fixed-width	11.2
dictionary, term pointers into string	7.6
with blocking, k = 4	7.1
with blocking & front coding	5.9
collection (text, xml markup etc)	3,600.0
collection (text)	960.0
Term-doc incidence matrix	40,000.0
postings, uncompressed (32-bit words)	400.0
postings, uncompressed (20 bits)	250.0
postings, variable byte encoded	116.0
postings, γ–encoded	101.0

Positional queries and indices

- Document containing phrase “New York”
- … “belt” within 4 words of phrase “key ring”
- Relax a positional query to AND
  - “new york” → “new” AND “york”
- Not all docs that pass the AND filter will have the phrase
- To filter, must read the document
- Random seek, very slow
- Solution: in the posting list, retain not only the doc ID, but also the word offset (position) where the word occurred → dgap and pgap

Study of pgap distributions



Pgap distribution very different from what would result by random placement of tokens in document of given length

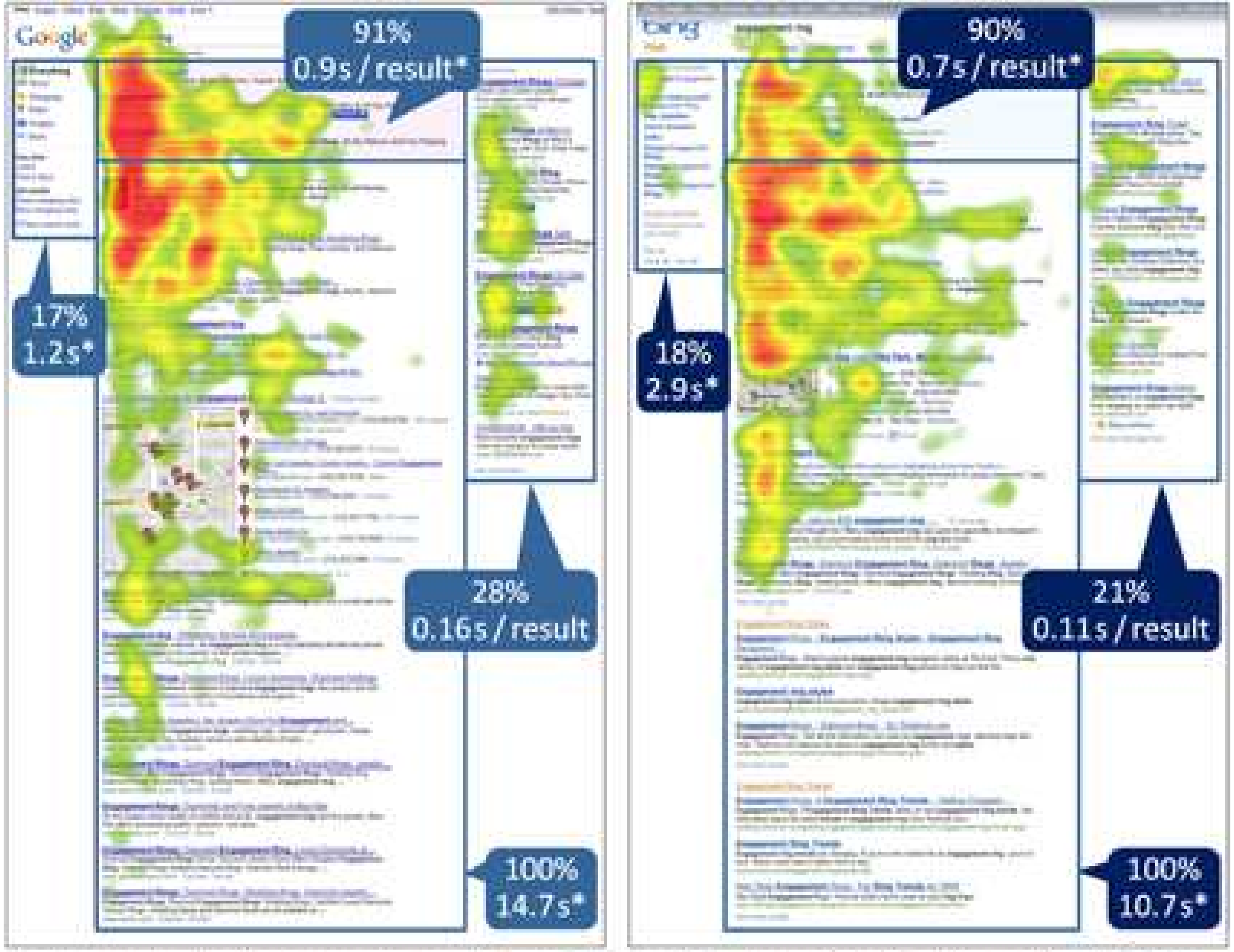
Gap from beginning to first term occurrence very different from second and subsequent gaps

X-axis=log(pgap), y-axis=frequency

Relevance ranking

Eye tracking study on search results

Google→



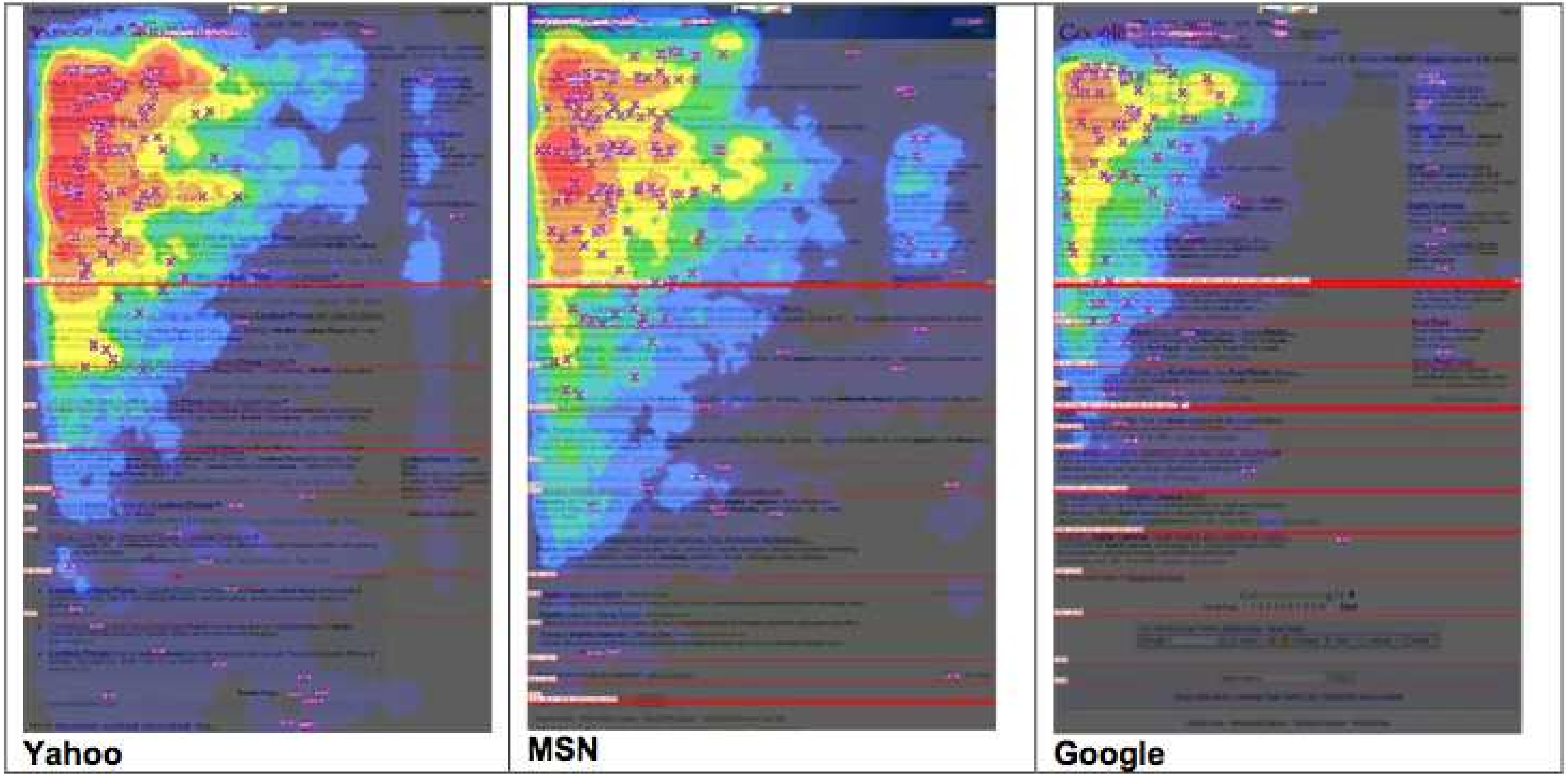
←Bing

Heatmaps showing the aggregate gaze time of all 24 participants on Google (left) and Bing (right) for one of the transactional tasks. The red color indicates areas that received the most total gaze time

http://blog.mediative.com/en/2011/08/31/eye-tracking-google-through-the-years/

tasks. Asterisks indicate values that were significantly different between Google and Bing at alpha = .1.

Yahoo, MSN, Google (2006)



- Basic ranking principle: most relevant first
  - How to measure relevance of document to query?
- Not always valid, e.g., diversity

Basic ranking principles

- Doc most relevant for single word query?
  - Word occurs or not (0/1) loses much info
  - Doc with most occurrences should win
  - Word occurs 0 vs. 1 times, …, 25 vs. 27 times?
- In multi-word query, all words not equally important
  - `dual boot computer windows ubuntu`
  - If a word appears in every single doc, how important is it for ranking?
- On the Web, hundreds of other signals: hyperlink popularity, clickthrough, spam

Term frequency tf(d,t)

- $n(d,t)$ = number of times t occurs in d
- Raw term frequency not what we want
  - Doc with 10 occurrences of t is more relevant (to query t) than doc with 1 occurrence
  - But not 10 times more relevant
- Diminishing returns transform: log function
  - Can we learn the form of this function from data?

$$\mathrm{tf}(d,t) = \begin{cases} 1 + \log n(d,t), & n(d,t) > 0 \\ 0, & \text{otherwise} \end{cases}$$

Inverse document frequency idf(t)

- N = number of docs in corpus
- $1 \le N(t) \le N$ = number of docs where t occurs one or more times
- If $N(t)=N$ then t is useless for discriminating between documents
- $N/N(t)$ is a measure of rarity of term t
- Using again a diminishing-returns function

$$\mathrm{idf}(t) = \log \frac{N}{N(t)}$$

- Does not matter to single term queries

Vector space model

- Suppose corpus vocabulary is V
- Each document is a vector in $\mathbb{R}^{|V|}$
- The t^{th} component is given by

$$\text{tfidf}(d,t) = \text{tf}(d,t)\text{idf}(t)$$

- The query q is interpreted as a set of terms
- Suppose score of doc d given query q is

$$\text{score}(q,d) = \sum_{t \in q \cap d} \text{tfidf}(d,t)$$

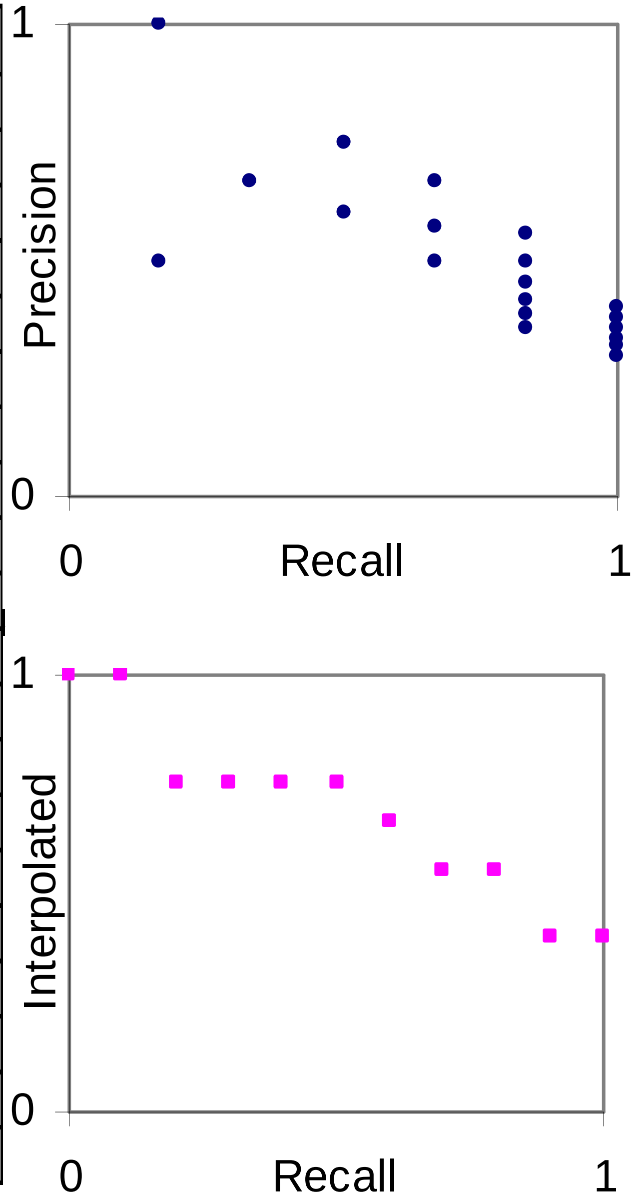
- Long docs have unfair advantage
  - Copy-paste a doc five times---score?
- Solution: scale doc vectors to unit length

In defense of heuristics

- TFIDF cosine and closely related BM25 perform very well
- Some parts of TFIDF can be justified in probabilistic or information theoretic terms
- Modern search engines combine TFIDF with hundreds of other signals
  - TFIDF and Jaccard similarity between query and different text fields: title, header, anchor text, …
  - Link-based scores: PageRank, spam flags, …
- Combine signals using machine learning
  - Relevance judgments from editors, past clicks

Evaluation: Recall, precision, F1

k	r_k
1	1
2	
3	1
4	1
5	
6	1
7	
8	
9	1
10	
11	
12	
13	
14	
15	1
16	
17	
18	
19	
20	



- As recall is increased, precision generally (but not always) decreases
- Interpolated precision fixes this anomaly
- F1 is harmonic mean

$$F_1 = \frac{2PR}{P+R}$$

- F1 small if one is compromised severely for the sake of the other

Evaluation: Average precision

- "Informational" queries
- High-ranking relevant hits matter a lot
- But user continues exploring (with increasing satiation or fatigue)
- Accrues reward for each additional relevant doc, but reward decreases with rank (fatigue)
- Fix one query, suppose there are R relevant documents at ranks $1 \leq p_1 < p_2 < \cdots < p_R$
- Precision at rank p_i is (i/p_i)
- Over all relevant ranks: $\mathrm{AvgPrec} = \frac{1}{R} \sum_{i=1}^{R} \frac{i}{p_i}$

Evaluation: NDCG

- Fix query q
- Relevance level of document ranked j wrt q is $r_q(j)$
- $r_q(j)=0$ means totally irrelevant
- Response list is inspected up to rank L
- Discounted cumulative gain for query q is

$$\mathrm{NDCG}_q = \underbrace{Z_q}_{\text{normalized}} \overbrace{\sum_{j=1}^{L}}^{\text{cumulative}} \frac{\overbrace{r_q(j)}^{\text{gain}}}{\underbrace{\log(1+j)}_{\text{rank discount}}}$$

- Z_q is a normalization factor that ensures the perfect ordering has $\mathrm{NDCG}_q = 1$
- Overall NDCG is average of NDCG_q over all q
- No notion of recall, only precision at low ranks

Augmenting text with entity annotations

Challenging queries

- Select-project: Price of Opteron motherboards with at least two PCI express slots
- Join: Artists who got Oscars for both acting and direction
- OLAP/tabulation: Is the number of Oscars won directly related to production budget?
- Uncertainty/consensus: Exxon Valdez cleanup cost

Why difficult?

- No variables
  - **?a** acts, **?a** directs movies
- No types
  - ?m $\in$ ***Motherboard***, ?p $\in$ ***MoneyAmount***
- No predicates
  - ?m **sells for** ?p, ?m **costs** ?p
- No aggregates
  - Large variation in Exxon Valdez estimate
- SQL, Web search, “query language envy”

What if we could ask…

- ?f $\in^{+}$ Category:FrenchMovie
- ?a $\in$ QType:Number
- ?p $\in$ QType:MoneyAmount
- ?c1, ?c2 are snippet contexts
- InContext(?c1, ?f, ?a, +oscar, won),
- InContext(?c2, ?f, ?p, +"production cost") or InContext(?c2, ?f, ?p, +budget)
- Aggregate(?c1, ?c2)
- Answer: list of ⟨?f, ?a, ?p⟩ tuples

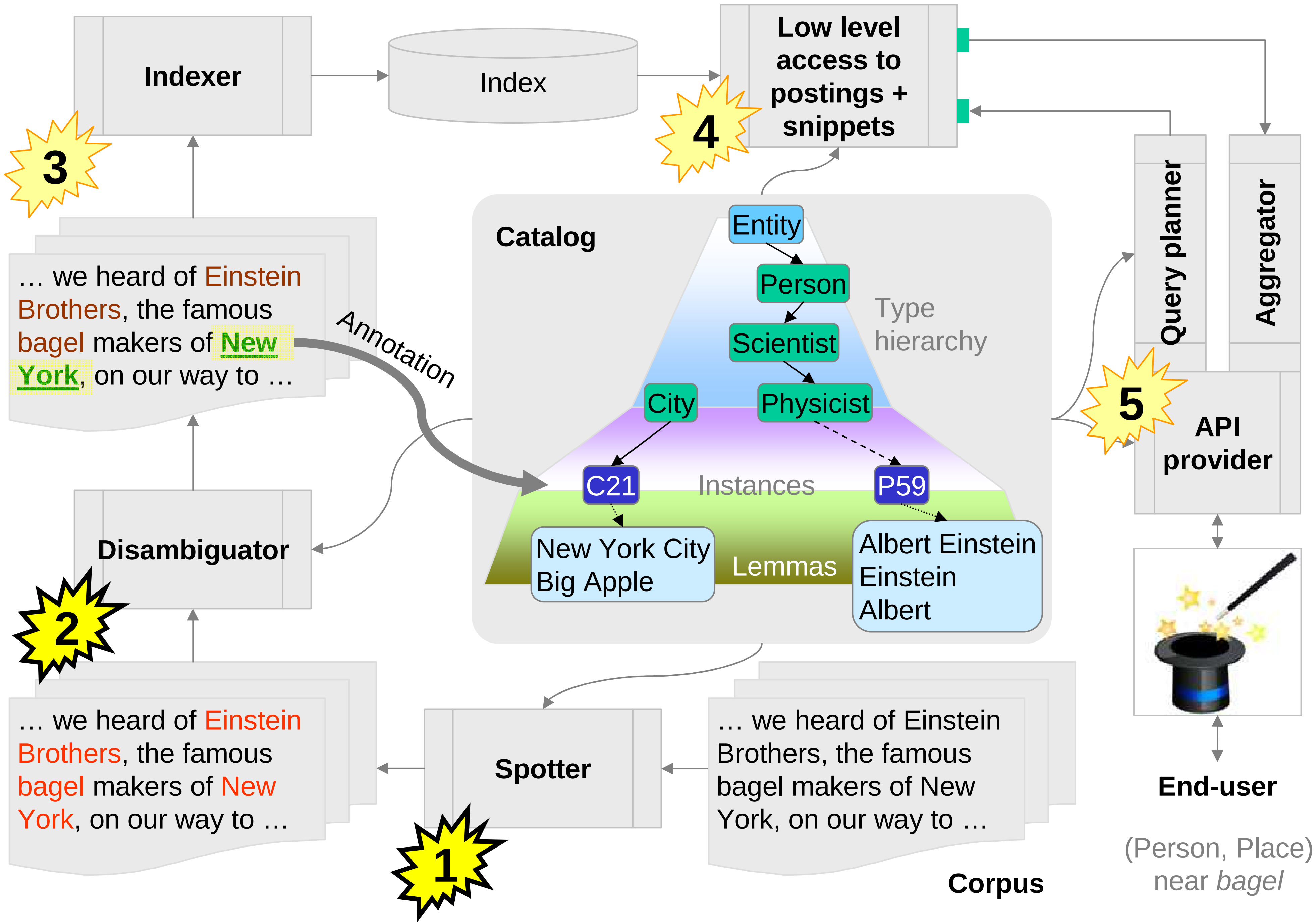
Extended document representation

- Earlier, document = sequence of tokens
- Tokens have offsets 0, 1, …
- Now, document is presented as
  - Sequence of **slots** — simply positions
  - One or more **fields** over these slots
  - A slot in each field may have 0, 1 or more tokens

Fields ↓

Slot offset	0	1	2	3	4	5	6	7	8
Raw text	Google	bought	YouTube	for	$	50	M	in	cash
Normalized tokens	google	buy	youtube	for	$	50	m	in	cash
Entities									
Types	company		company		money				

"Catalog" provides standardized entities and types



Indexer
Index
Low level access to postings + snippets
3
4
Catalog
Entity
Person
Scientist
Physicist
City
Type hierarchy
C21
P59
Instances
New York City
Big Apple
Lemmas
Albert Einstein
Einstein
Albert
Annotation
… we heard of Einstein Brothers, the famous bagel makers of New York, on our way to …
Query planner
Aggregator
5
API provider
Disambiguator
2
… we heard of Einstein Brothers, the famous bagel makers of New York, on our way to …
Spotter
1
… we heard of Einstein Brothers, the famous bagel makers of New York, on our way to …
Corpus
End-user
(Person, Place) near *bagel*

Mentions and spots

The lack of **memory** and time efficient **libraries** in the **free software world** has been the main motivation to create the C **Minimal Perfect Hashing Library**, a portable **LGPL library**.

- A **mention** is any token segment that may be a reference to an entity in the catalog
- Mention + limited token context = **spot**
- Mentions and spots may overlap
- S_0: set of all spots on a page
- $s \in S_0$: one spot among S_0

A massive similarity join

… the New York Times reported on school library budgets …

York University
Duke of York
…

New York City
New York State
York Universi
…

New York Times
Time Magazine

Library, a collection of books...
Library (computing), a collection of subprograms...
Library (Windows 7), virtual folder that aggregates...
Library (electronics), a collection of cells, macros...
Library (biology), a collection of molecules...
Library Records, a record label
"The Library" (Seinfeld)
Library (UTA station), a transit station...
Library of Congress

Wikipedia:
2.5M entities
2.8M “lemmas”
7M lemma tokens
IDF, prefix/exact match, case, …

Disambiguation

- *s* is a spot with a mention of some entity
- Γ_s is the set of candidate entities for *s*
- $\gamma \in \Gamma_s$ is one candidate entity for *s*
- *s* may be best left unconnected to any entity in the catalog (“no attachment”, NA)
  - Most people mentioned on the Web are missing from Wikipedia
  - But all known constellations are in there

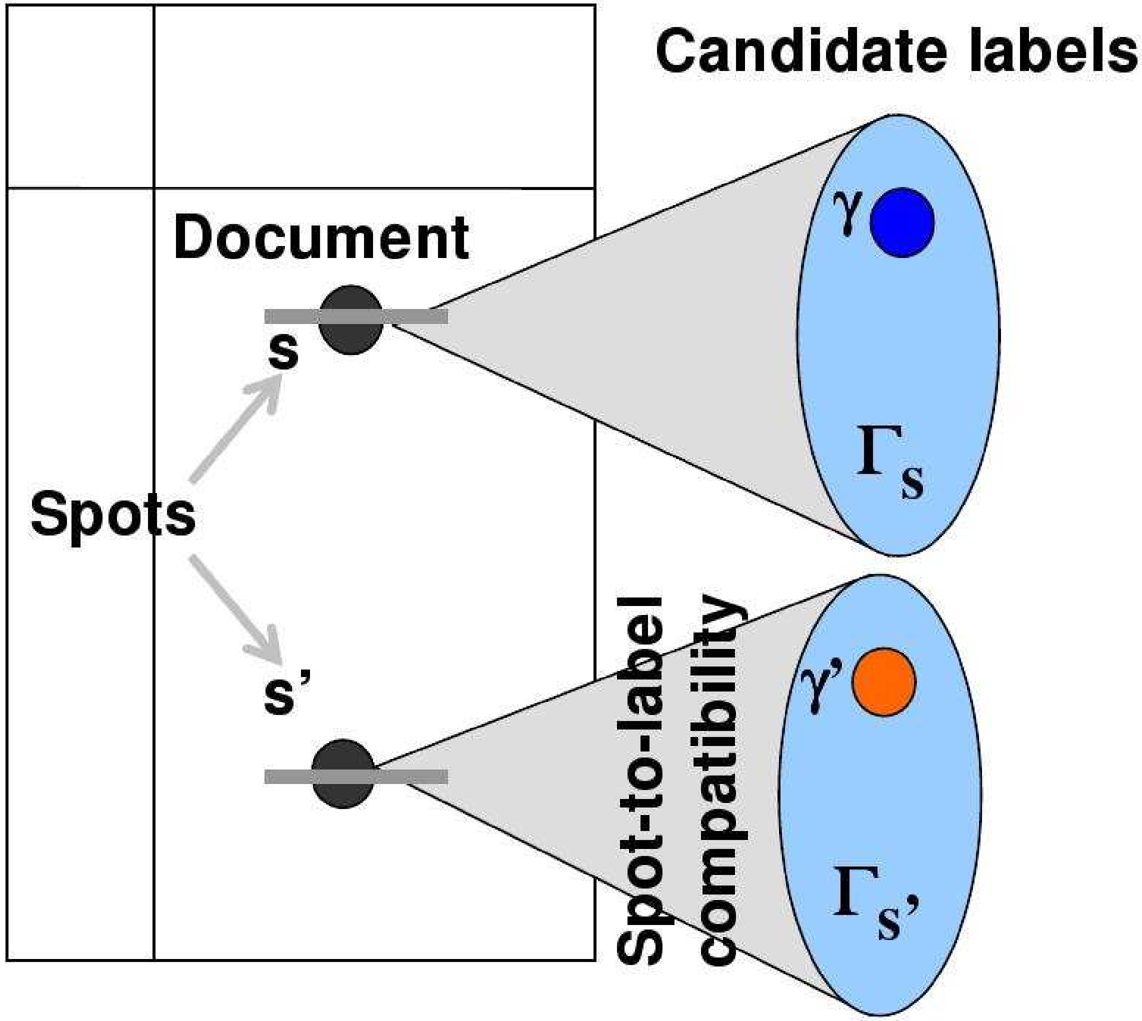
Local context signal

Jacksonville Jaguars

Jaguar (Car)

Jaguar (Animal)

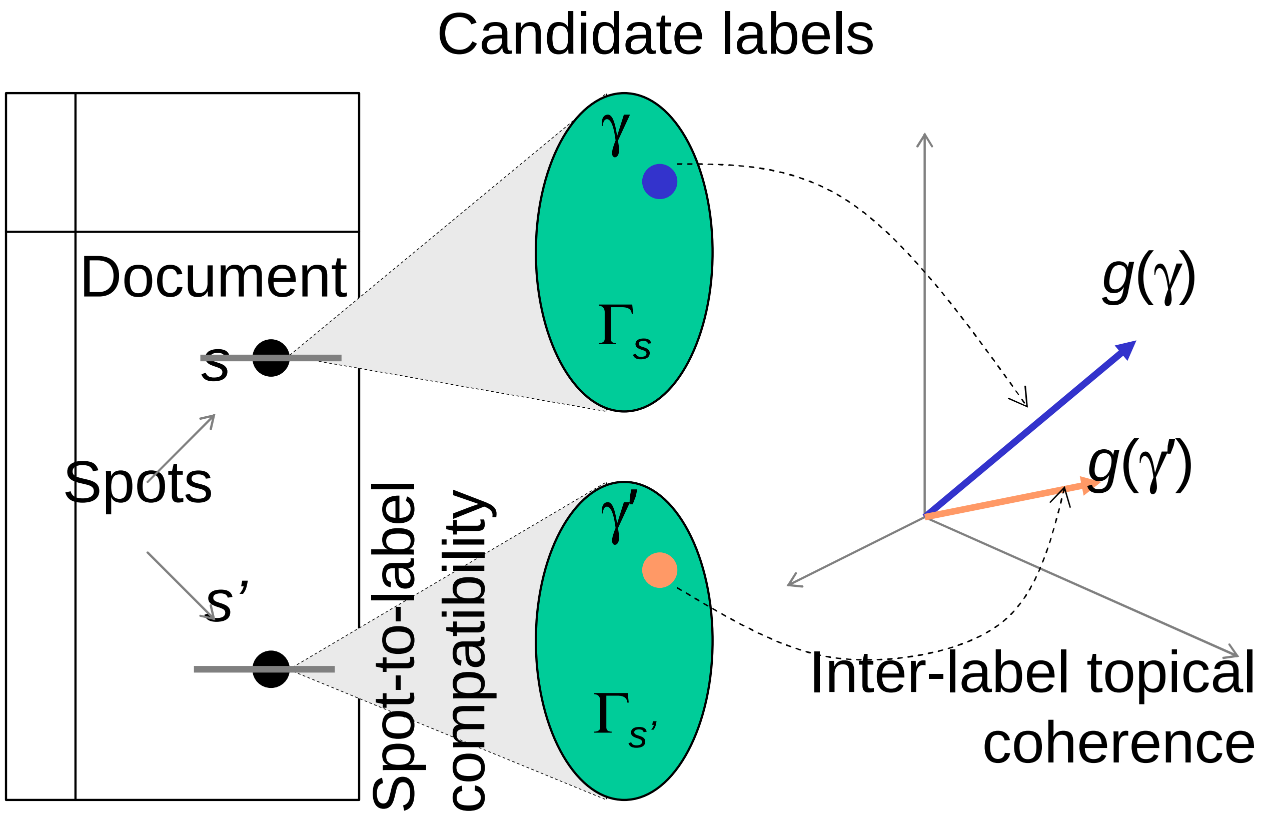
On first getting into the 2009 **Jaguar** XF, it seems like the ultimate in **automotive tech**. A red **backlight** on the **engine** start **button** **pulses** with a **heartbeat** cadence.



Exploiting collective info

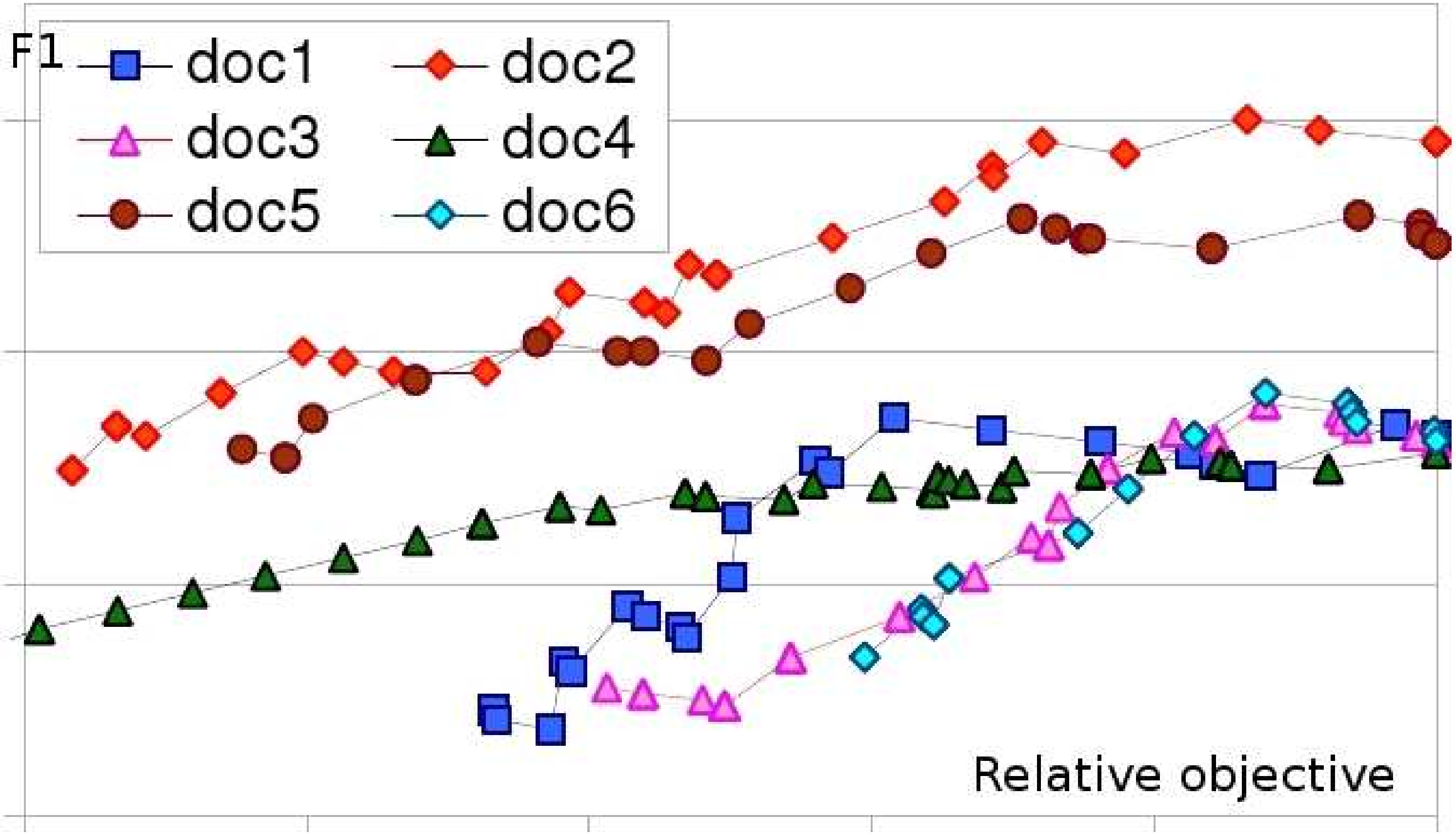
- Let $y_s \in \Gamma_s \cup \mathrm{NA}$ be the variable representing entity label for spot s
- Pick all y_s together optimizing global objective

Collective formulation



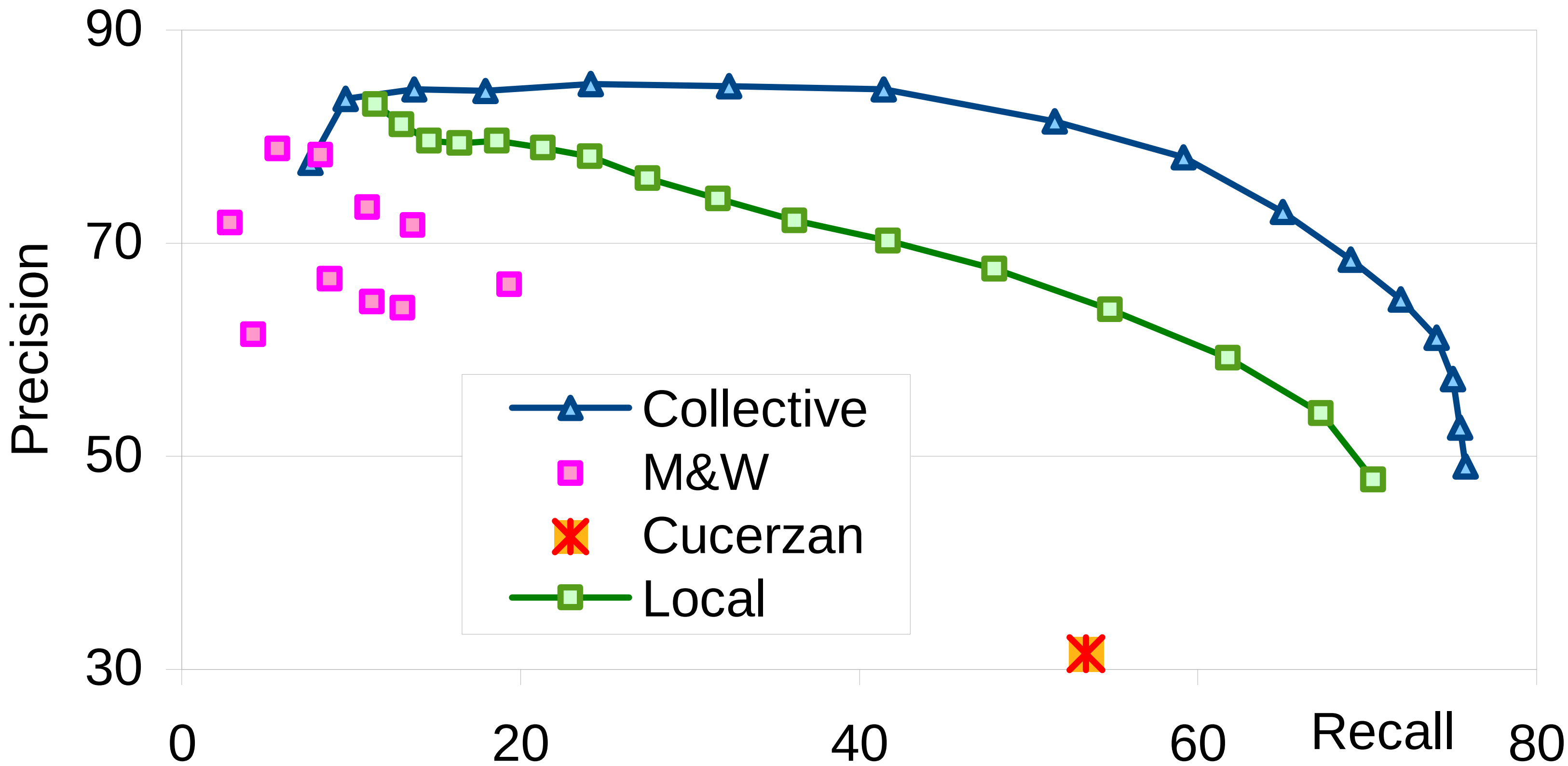
- Embed entities as vector $g(\gamma)$ in feature space
- Maximize local compatibility + global coherence

Collective model validation



- Local hill-climbing to improve collective obj
- Get F1 accuracy using ground truth annotations
- Very high positive correlation

Collective accuracy



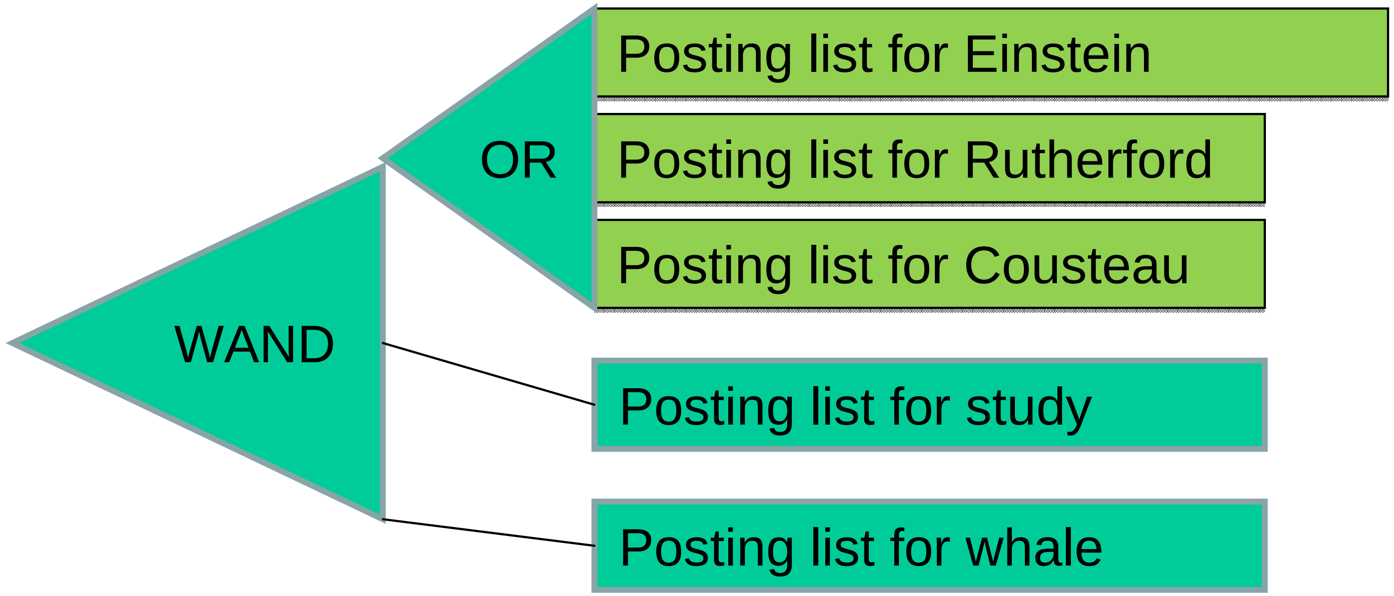
- ~20,000 spots manually labeled in Web docs
- Local = using per-spot context separately
- Collective = relaxing collective integer program
- Cucerzan, M&W = prior art

Typed entity search

Example of typed entity search

- Entity annot = doc ID, token span, entity ID
- But the query will typically ask for entities by a target type
  - ?s $\in$ Category:Scientist
  - InContext(?c, ?s, study whale)
- And want ?s to be instantiated to candidate scientist
- Which are then ranked by aggregating evidence contribution from contexts ?c
- Therefore need to index types as well

Query expansion vs. posting materialization



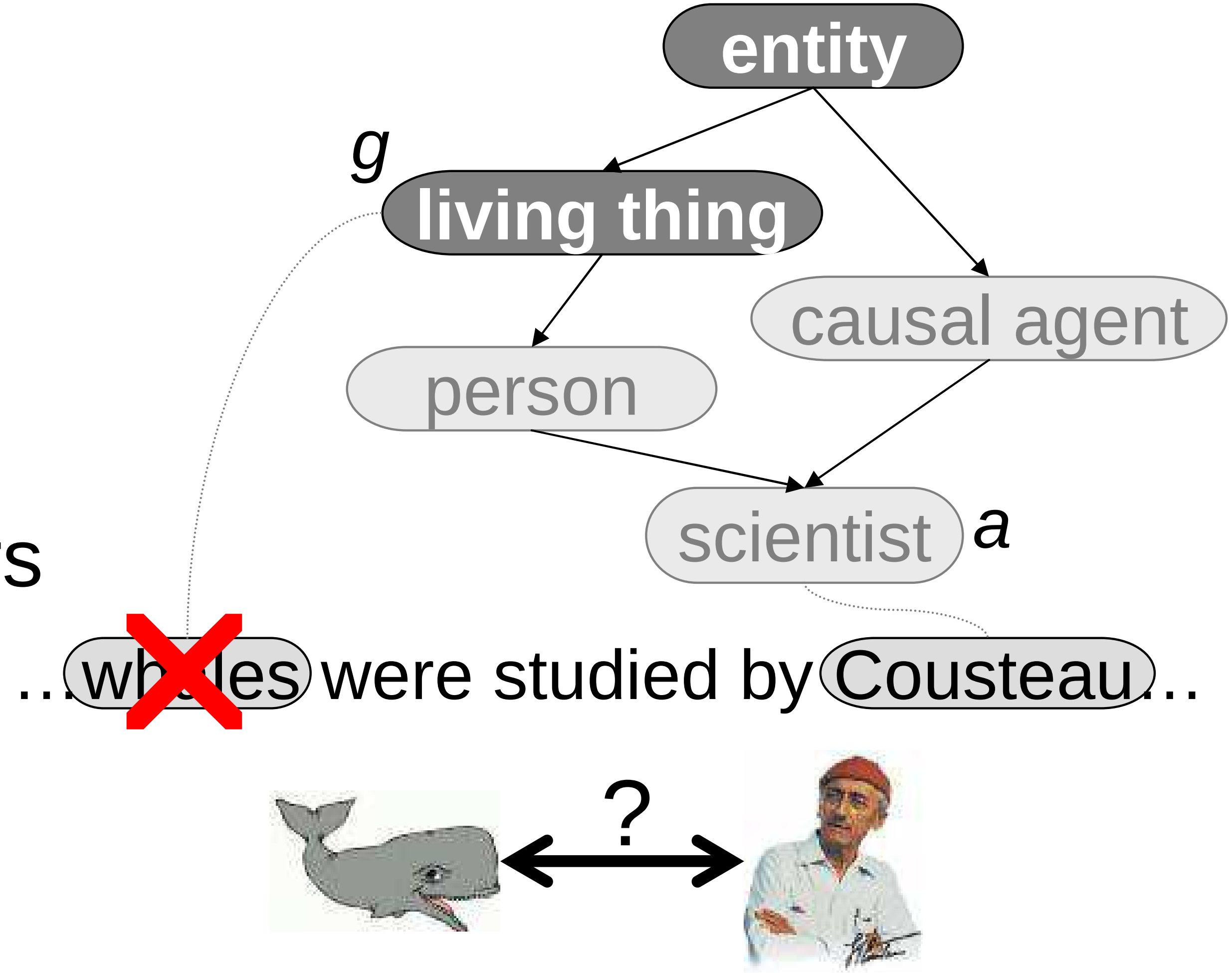
- Did Einstein, Rutherford...study whales?
  - WordNet knows 650 scientiest, 860 cities
- OR merge is too slow
- Can’t scan word postings many times over
- Limited materialization of the OR?

Indexing for InContext queries

- Index expansion
  - Costeau→scientist→person→organism→ living_thing→…→entity
  - Pretend all these tokens appear wherever Cousteau does, and index these
- Works ok for small type sets (5—10 broad types), but
  - WordNet: 15k internal, 80k total noun types
  - Wikipedia: 250k categories
- Index size explosion unacceptable

Pre-generalize

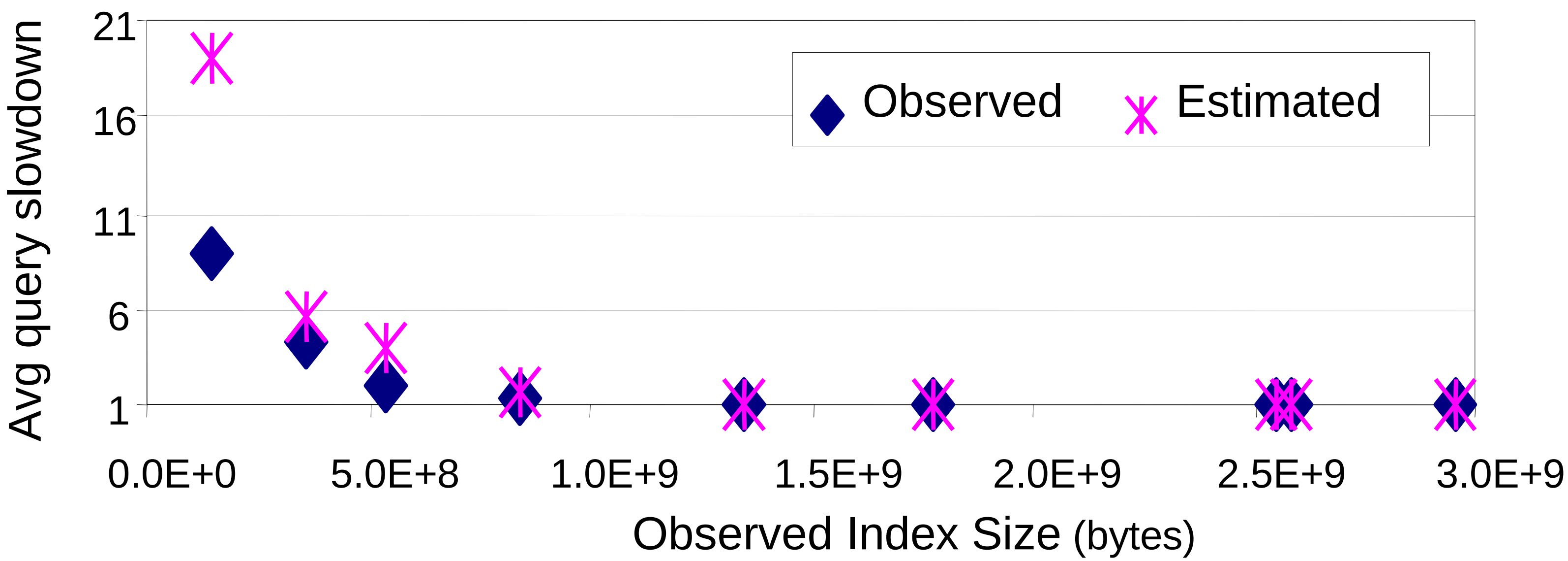
- Index a subset $R \subset A$
- Query type $a \notin R$
- Want k answers
- Probe index with g, ask for $k' > k$ answers



Post-filter

- Fetch k' high-scoring (mentions of) entities $w \in^{+} g$
- **Check if $w \in^{+} a$** as well; if not, discard
- If $< k$ survive, restart with larger k (expensive!)

Index size vs. query slowdown



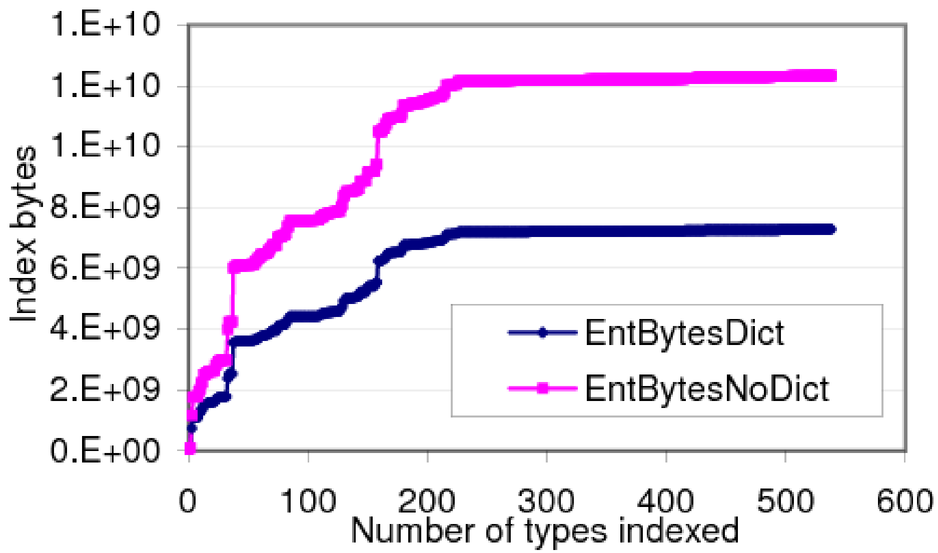
- Annotated TREC corpus
- Space = 520MB < inverted index = 910MB
- Query slowdown ≈ 1.8
- From TREC to Web?

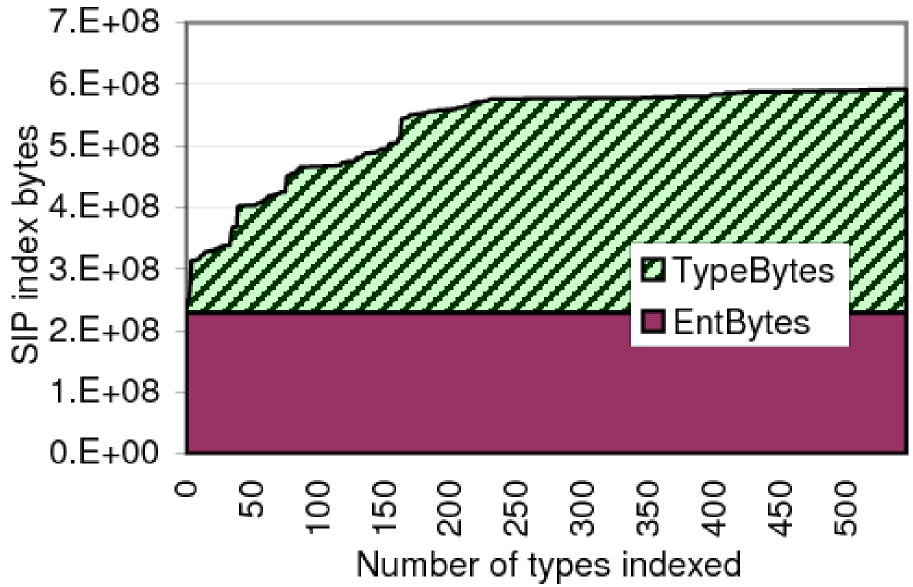
Corpus/Index	Gbytes
Original corpus	5.72
Gzipped corpus	1.33
Stem index	0.91
Full type index	4.30
Reachability index	0.01
Forward index	1.16
Atype subset index	0.52

The SIP index

- To post-filter, need to check if $w \in^{+} a$
- Are query word matches near the entity mention?
- Have to report only 10 top entities, but
- There could be millions of snippets supporting each entity
- Cannot seek to each snippet
- **SIP** index: snippet interleaved postings
  - Embed ***w*** (the entity) in the type index itself

Compressing the SIP entity ID



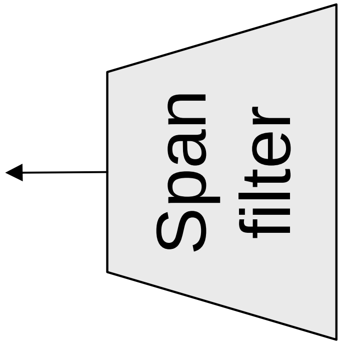


- Millions of entities globally, but …
- Few mentioned in one particular document
- Write entity dictionary (list of IDs) at beginning of doc block, then inline dictionary offset in posting
- 40% type index size reduction!
- Comparable to entity index in size

Modified query processor; some statistics

?s ∈ Scientist, InContext(?c, ?s, study whale)

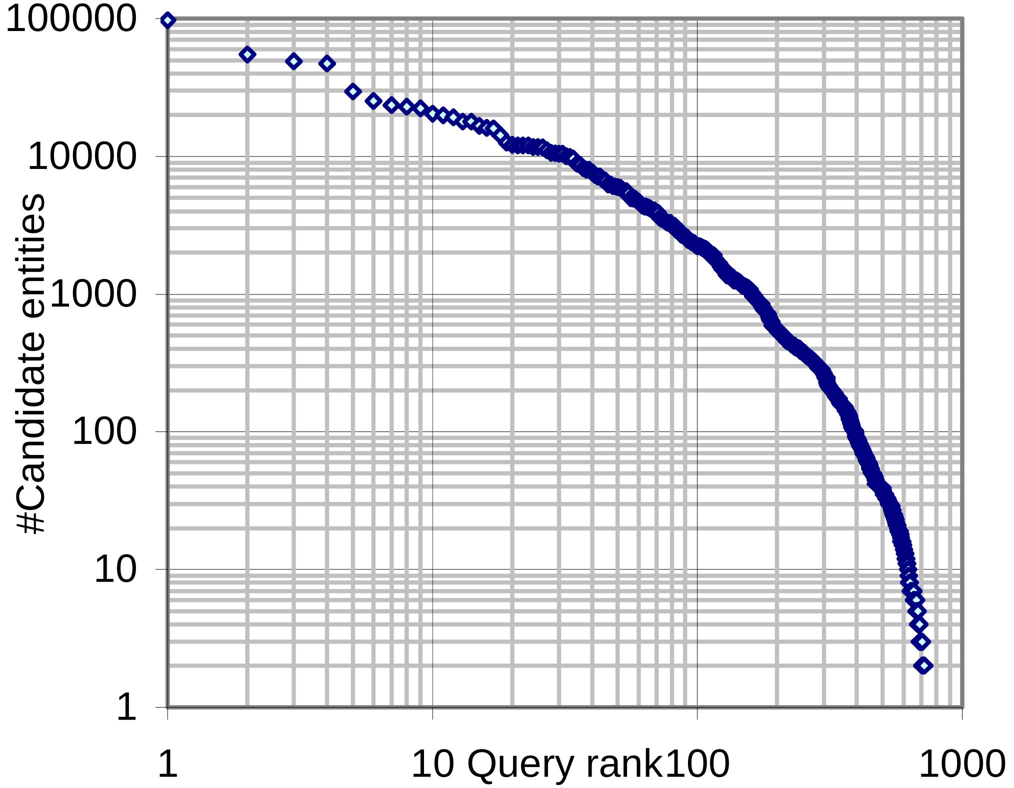
Snippets supporting candidate entities

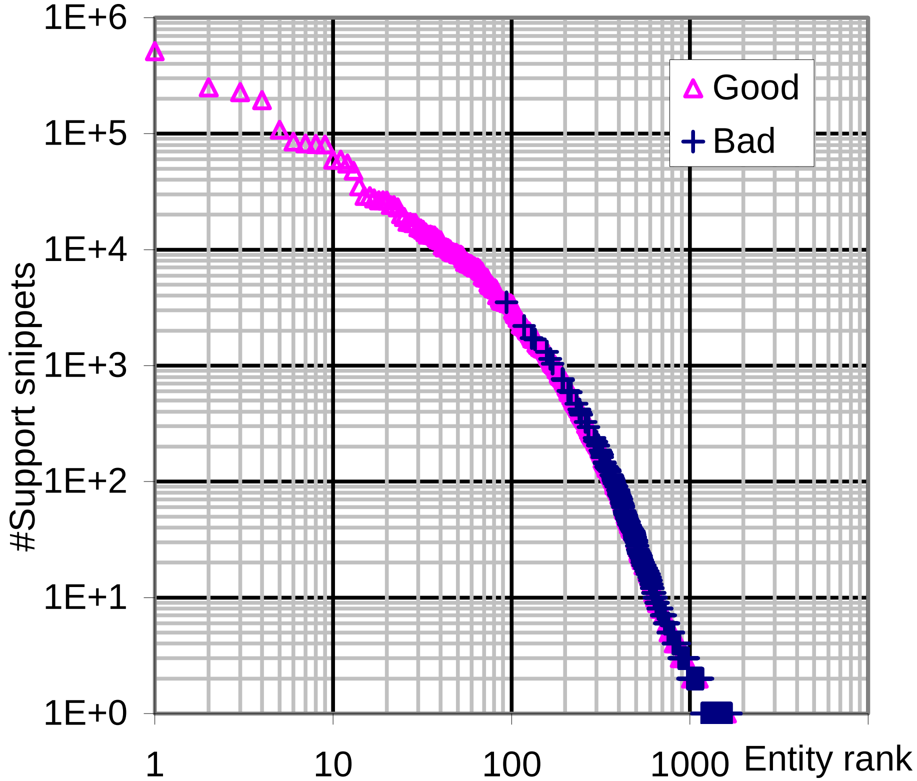


Token posting list for *study*

Token posting list for *whale*

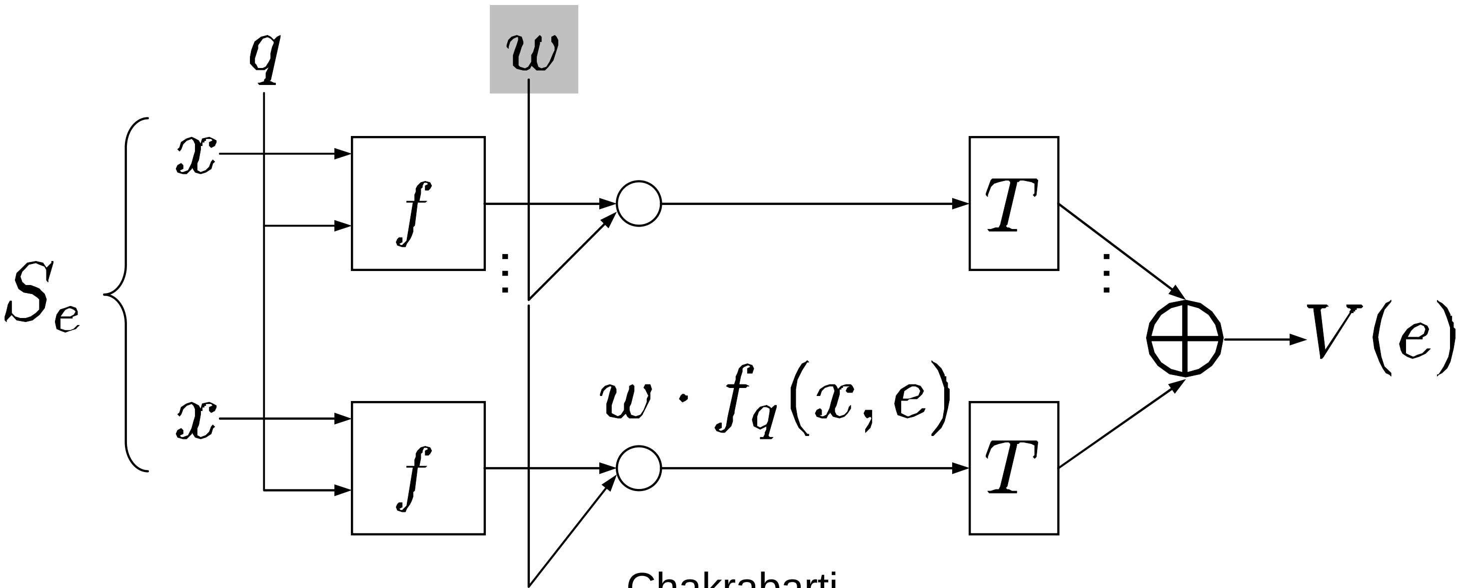
Type SIP posting list for *Scientist*





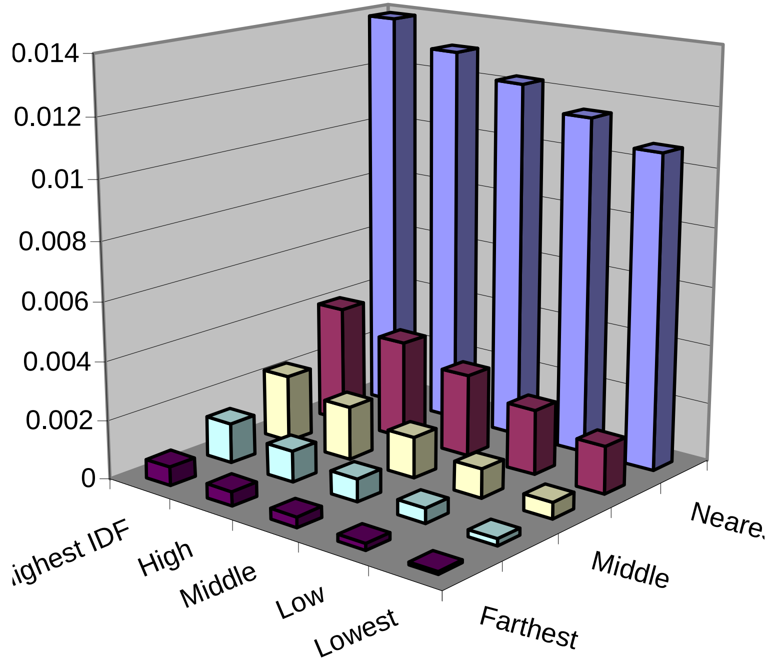
Aggregating evidence from snippets

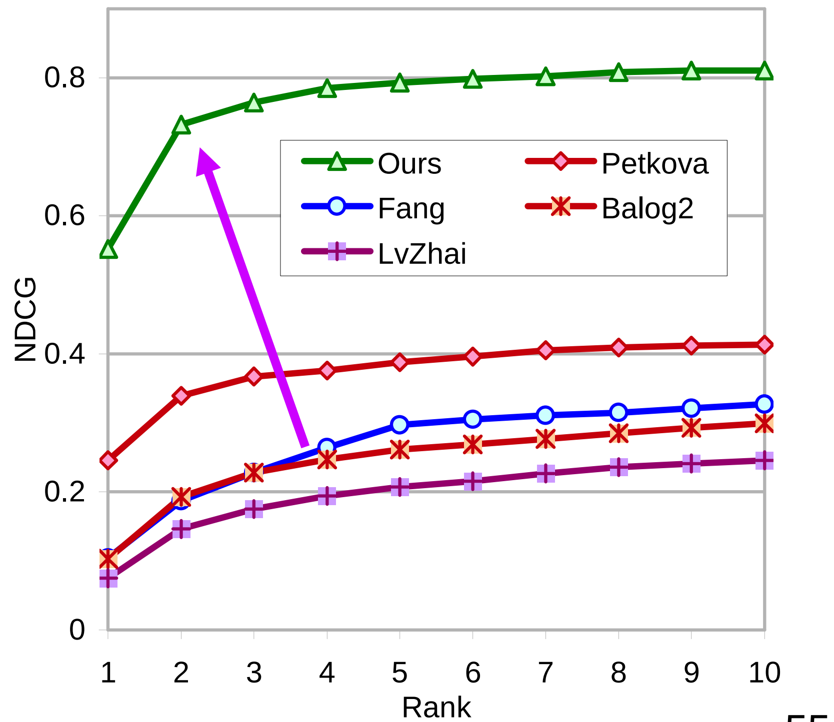
- S_e = snippets supporting entity e, x is one snippet
- $f_q(x,e)$ = feature vector capturing signals
  - Rarity of words in q matching x
  - Proximity between matching words and mention of e
- T superlinear→softmax, sublinear→soft count
- $\oplus$ aggregates score over snippets supporting e
  - Sum, average, soft-or, …



"Non-parametric" models

- Instead of forcing a combination between rarity (IDF) and proximity (distance in words)…
- … discretize to grid
- Learn weight in the product space
- Individual snippet scores quite noisy
- Fix simple aggregations, train parameters
- Better than language model approaches





Conclusion

- How to identify mentions of entities and implied types in unstructured text
- High performance indexing for text as well as semantic annotations
- Ranking for entity search queries
- Active area, breathtaking rate of innovation
  - Lightly supervised bootstrapping of type, entity, and relation catalogs
  - Query interpretation and representation
  - Search and ranking on graph data models (also useful for social networks)

Acknowledgment + references

- Manning, Schutze and Raghavan online book and slides at http://nlp.stanford.edu/IR-book/ (for some slides)
- Mining the Web book Web site at http://www.cse.iitb.ac.in/~soumen/mining-the-web/
- Additional reading list at http://www.cse.iitb.ac.in/~soumen/mining-the-web/Toc2.html
- Project page at http://www.cse.iitb.ac.in/~soumen/doc/CSAW