A Search Engine for Natural Language Applications
AND
Relational Web Search: A Preview

Michael J. Cafarella
(joint work with Michele Banko, Doug Downey, Oren Etzioni, Stephen Soderland)
CSE454
University of Washington
November 15, 2005

Outline

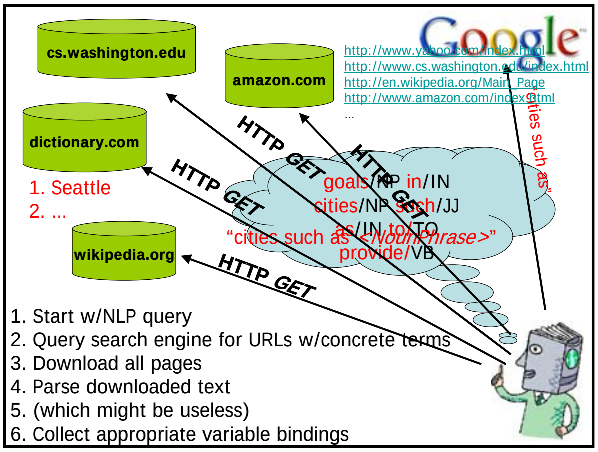
- NLP applications and the web corpus
- The Bindings Engine
  - Query language
  - Neighbor Index
- Experiments
  - >300x performance gain
  - ~4x space penalty
- Novel applications
- Relational Web Search: Preview

2

NLP Applications: An Example

- KnowItAll (Etzioni et al, WWW04, AIJ05): unsupervised web-scale info extraction
  - Generate candidate fact-extractions from web
  - Use extraction frequencies to assess probability that a candidate extraction is true
- Simplified algorithm:
  - Search web for various hypernym-phrases (e.g., "cities such as X" where X is a noun)
  - Count hits for each unique X
  - Use hit counts as inputs to trained classifier
  - Sort Xs by classifier probability, then threshold

3



Pointwise Mutual Information Information Retrieval (PMI-IR)

- Turney uses PMI-IR to find semantic orientation (ACL02)
- Estimate co-occur probs using hitcounts

$$SO(phrase) = \log\left(\frac{hits(phrase\ NEAR\ "excellent")hits("poor")}{hits(phrase\ NEAR\ "poor")hits("excellent")}\right)$$

- Anything more than NEAR (e.g., breaking at sentences) requires original text
- If there are 10k phrases and 14 reference words, we need 140k search queries

5

Search engine inefficiencies

- Search engines ill-suited to the task
  - System must download and parse many docs for each query; many end up as useless
  - Worst of all, each downloaded document likely requires a disk seek
  - Certain apps require #queries = #candidates x #phrases
- Problems common to many NLP apps

6

Bindings Engine

- Bindings Engine (BE) is search engine where:
  - No downloads during query processing
  - Disk seeks constant in corpus size
  - #queries = #phrases
- BE's approach:
  - "Variabilized" search query language
  - Pre-processes all documents before query-time
  - Integrates variable/type data with inverted index, minimizing query seeks

7

Query language

cities such as *<NounPhrase>*

President Bush *<Verb>*

<NounPhrase> is the capital of *<NounPhrase>*

reach me at *<phone-number>*

- Any sequence of concrete terms and typed variables
- (some limitations posed by current index)
- NEAR is insufficient
- Paper also discusses functions, which modify variable bindings (e.g., "*head(<NounPhrase>)*")

8

BE processing model

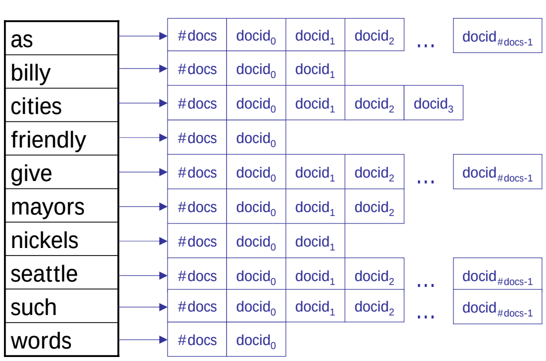
- Like a search engine, BE:
  - Downloads a corpus of pages
  - Creates an index
  - Uses index to process queries efficiently
- BE further requires:
  - Set of indexed types (e.g., "NounPhrase"), with a "recognizer" for each
  - String processing functions (e.g., "head()")
- A BE system can only process types and functions that its index supports

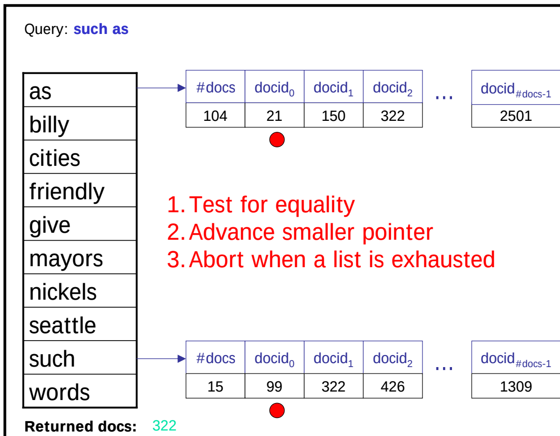
9

Index design

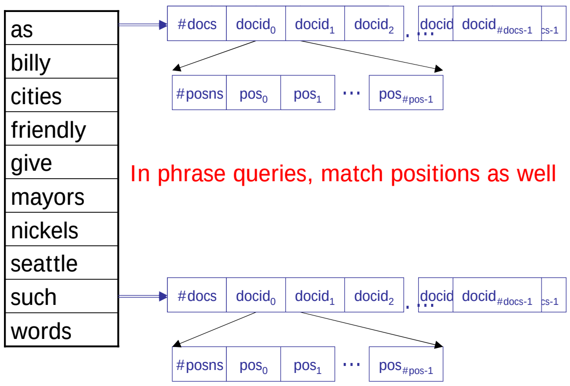
- Search engines handle scale with inverted index
  - Single disk seek per term
  - Mainly sequential reads
- Disk analysis
  - Seeks require ~5 ms, so only 200/sec
  - Sequential reads transfer 10-40 MB/sec
- Inverted index minimizes expensive seeks; BE should do the same
- Parallel downloads are just parallel, distributed seeks; still very costly

10





“such as”



Neighbor Index

- At each position in the index, store “neighbor text” that might be useful
- Let’s index <NounPhrase> and <Adj-Term>

“I love cities such as Seattle.”

Left	Right
	AdjT: “love”

14

Neighbor Index

- At each position in the index, store “neighbor text” that might be useful
- Let’s index <NounPhrase> and <Adj-Term>

“I love cities such as Seattle.”

Left	Right
AdjT: “I” NP: “I”	AdjT: “cities” NP: “cities”

15

Neighbor Index

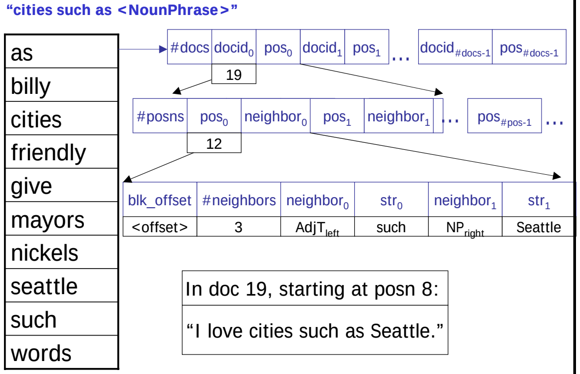
Query: “cities such as <NounPhrase>”

“I love cities such as Seattle.”

Left	Right
AdjT: “such”	AdjT: “Seattle” NP: “Seattle”

16

“cities such as <NounPhrase>”



1. Find phrase query positions, as with phrase queries
2. If term is adjacent to variable, extract typed value

Asymptotic analysis

- *k* concrete terms in query
- *B* bindings found for query
- *N* documents in corpus
- *T* indexed types in corpus

	Query Time (in seeks)	Index Space
BE	*O(k)*	*O(N * T)*
Std Model	*O(k + B)*	*O(N)*

18

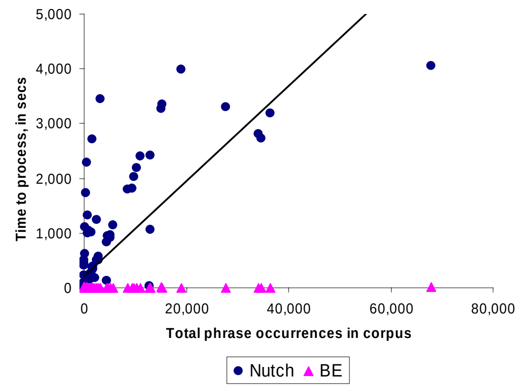
- *B* and *N* scale together; *k* often small; *T* often exclusive

Experimental details

- 20 machine cluster, 50m page corpus
- BE types: *<NounPhrase>* & *<Adj-term>*
- Experiment 1: 150 assorted queries with 1 variable, 2-3 concrete terms. BE vs Nutch-based "standard implementation"
- Experiment 2: KnowItAll system test

19

Experiment 1: Processing speed



Experiment 2: KnowItAll on BE

Num Extractions	Std Imp/ Google	BE	Speedup
10k	5,976s		
50k	29,880s		
150k	89,641s		

21

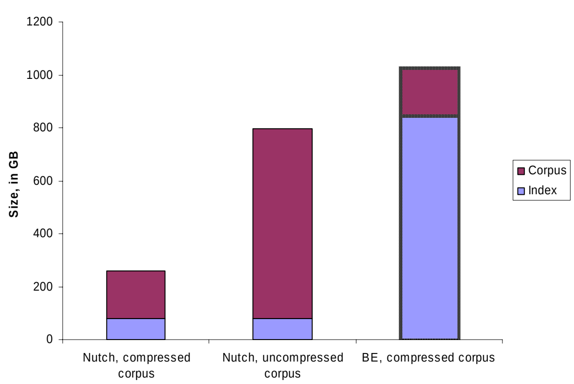
Experiment 2: KnowItAll on BE

Num Extractions	Std Imp/ Google	BE	Speedup
10k	5,976s	95s	63x
50k	29,880s	95s	314x
150k	89,641s	N/A	N/A

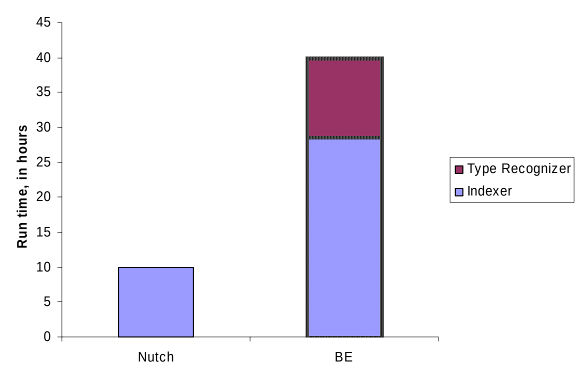
22

•BE still has to perform sequential reads, is not optimized

Costs: Index size



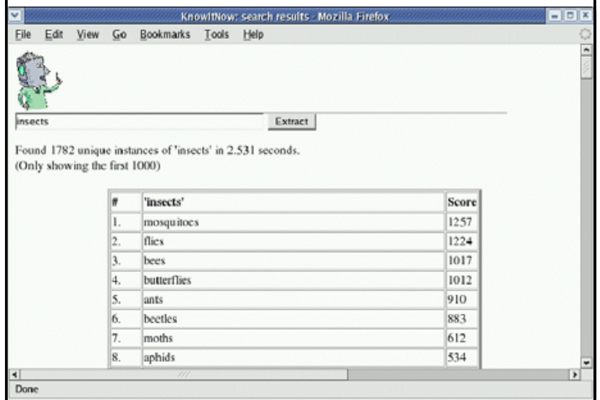
Costs: Index construction time



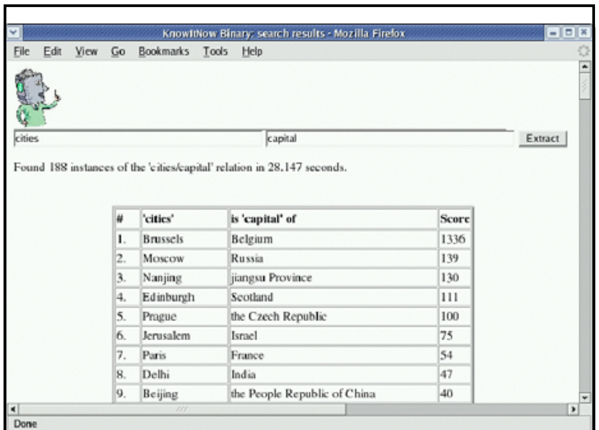
Novel applications: Interactive Information Extraction

- BE is fast enough to allow new interaction models
  - KnowItAll is a batch process
  - KnowItNow is interactive; approximates core of KnowItAll using a few BE queries

25



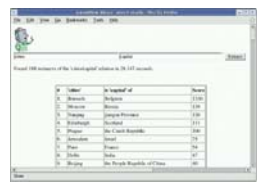
#	'insects'	Score
1.	mosquitoes	1257
2.	flies	1224
3.	bees	1017
4.	butterflies	1012
5.	ants	910
6.	beetles	883
7.	moths	612
8.	aphids	534



#	'cities'	is 'capital' of	Score
1.	Brussels	Belgium	1336
2.	Moscow	Russia	139
3.	Nanjing	jiangsu Province	130
4.	Edinburgh	Scotland	111
5.	Prague	the Czech Republic	100
6.	Jerusalem	Israel	75
7.	Paris	France	54
8.	Delhi	India	47
9.	Beijing	the People Republic of China	40

Relational Web Search

- Consider that last slide for a sec:

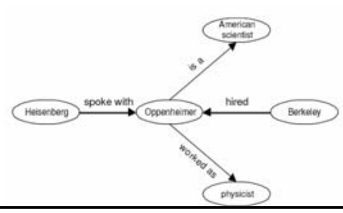


- It looks a lot like a database table
- Can web-style search generate structured output, instead of just a list of docs?

28

Relational Web Search (2)

- Modern search works treats docs as bags of words; no internal structure
- Instead, we use the corpus to assemble a huge *entity-relation* graph



29

Relational Web Search (3)

- We automatically extract it from the doc; it's called the *extraction graph*
- All searches are done over the E.G., not the original document set
- Lets us perform various queries:
  - Qualified-list ("west coast liberal arts colleges")
  - Unnamed-item ("tallest inactive volcano in Africa")
  - Relationship (describe relation between Bill Clinton and Justice Ginsberg)
  - Tabular (database table of cities and capitals)

30

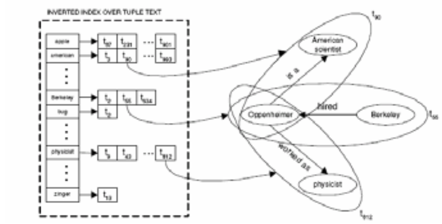
The Extraction Graph

- "Is-A" edges come from KnowItAll
- Predicate edges found by looking for certain linguistic patterns
- From 90m docs, we have:
  - 652m object-relation-object triples
  - 227m nodes
  - 544m edges
- ~71.7% of "Is-A" correct
- ~44% of predicate edges correct

31

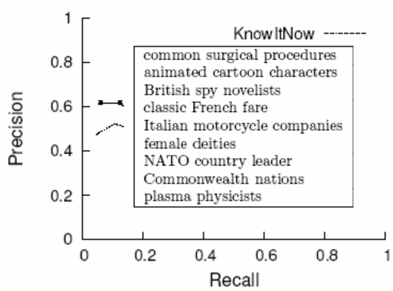
Searching

- Perform "spreading activation" search on graph
- Each edge has a "decay factor" that retards spread



32

Results: qualified-list



33

Results: tabular query

English philosopher	was born in	returned to	published
Isaac Newton	Lincolnshire		Philosophiae Naturalis Principia...
David Hume	Edinburgh	Scotland	idea
John Locke	Somerset, England	England	Some Consideration of the Consequences...
Thomas Paine	Thetford		Common Sense
Adam Smith	Kirkcaldy	England	Wealth of Nations
Hobbes	Wiltshire town	England	Leviathan

34

Questions?

- Thanks
- We're hiring!
- Comments to mjc@cs.washington.edu

35