Building Blocks for Semantic Search Engines: Ranking and Compact Indexing for Entity-Relation Graphs

Soumen Chakrabarti
www.cse.iitb.ac.in/~soumen
IIT Bombay

(In fewer words)

Ranking and Indexing for Semantic Search

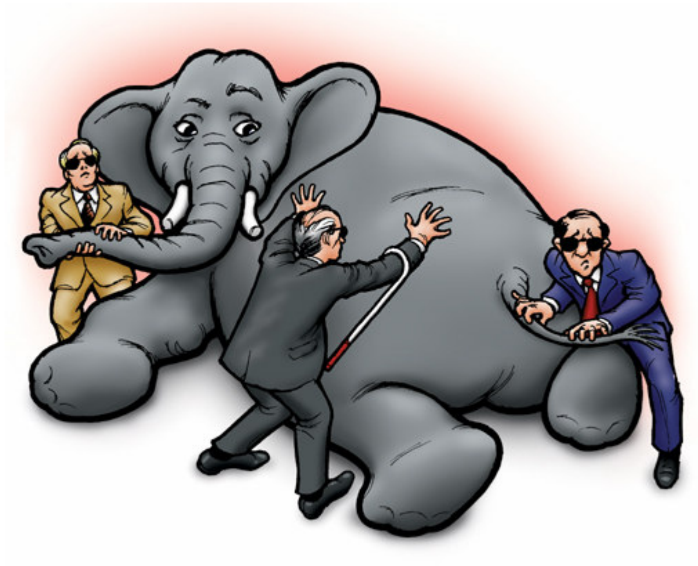
with

Alekh Agarwal, Sujatha Das
Vijay Krishnan, Kriti Puniyani

Supported by
IBM, Microsoft, Yahoo!

Working notion of semantic search

- Exploiting in conjunction
  - “Strings with meaning” – entities and relations
  - “Uninterpreted strings” – as in IR
- “Is-a” and other relations
- Proximity
- Conductance
- Can approximate many info needs
- “Warehousing” not enough

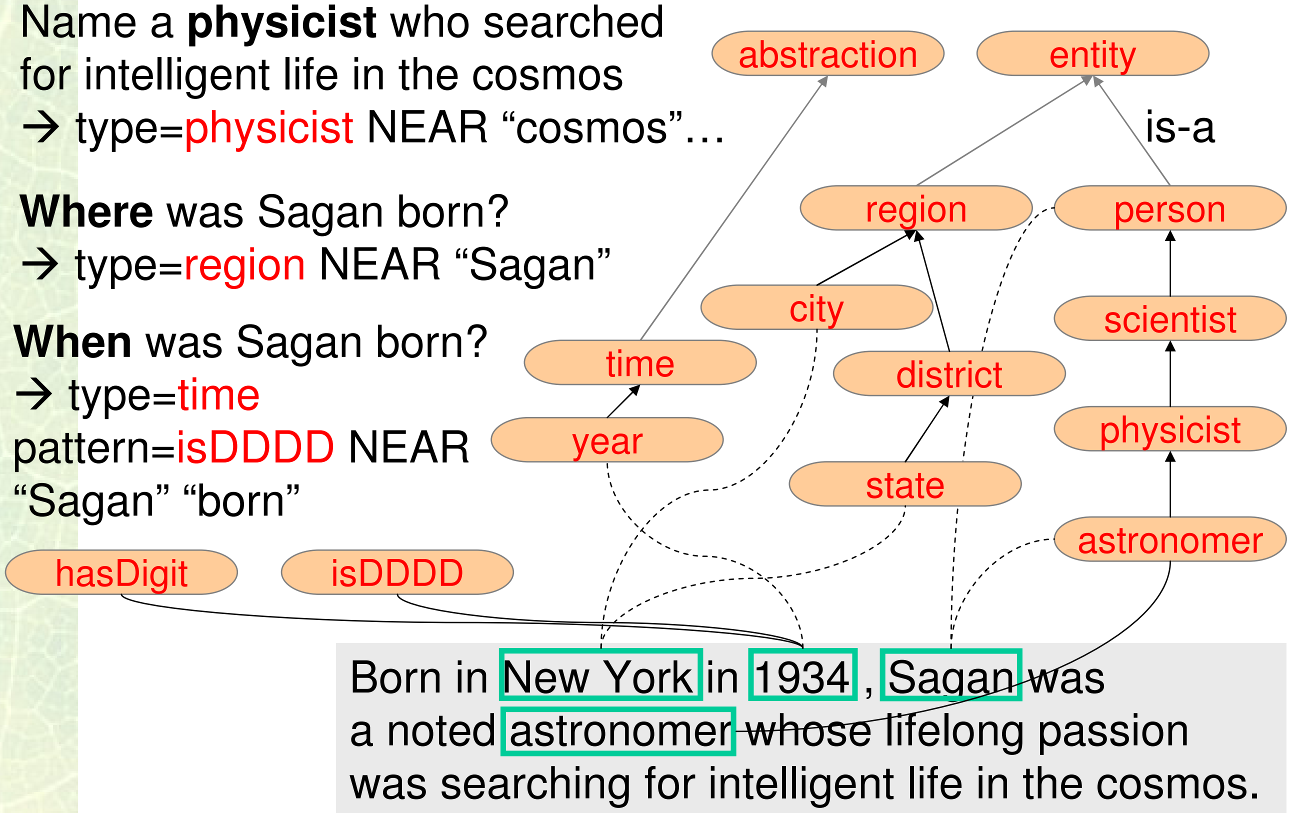


Type-annotated corpus and query e.g.

Name a **physicist** who searched for intelligent life in the cosmos
→ type=physicist NEAR “cosmos”…

Where was Sagan born?
→ type=region NEAR “Sagan”

When was Sagan born?
→ type=time pattern=isDDDD NEAR “Sagan” “born”



The query class we address

- Find a token span *w* (in context) such that
  - *w* is a mention of entity *e*
    - "Carl Sagan" or "Sagan" is a mention of the concept of that specific physicist
  - *e* is an instance of **atype** *a* given in the query
    - Which *a*=physicist …
  - *w* is "NEAR" a set of **selector** strings
    - "searched", "intelligent", "life", "cosmos"
- All uncertain/imprecise; we focus on #3
- Yet surprisingly powerful: correct answer within top 3—4 *w*'s for TREC QA benchmark

Contribution 1: What is "NEAR"?

- XQuery and XPath full text support
  - (distance at most|window) 10 words [ordered] – hard proximity clause, not learnt
  - ftcontains … with thesaurus at … relationship "narrower terms" at most ℓ levels
- No implementation combining "narrower terms" and "soft" proximity ranking
- Search engines favor proximity in proprietary ways

A learning framework for graph proximity

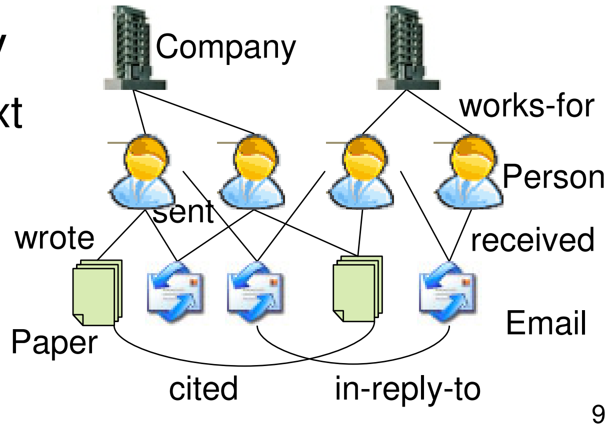
Contribution 2: Indexing annotations

- type=person NEAR theory relativity → type in {physicist, politician, cricketer,…} NEAR theory relativity
  - Large fanout at query time, impractical
- Complex annotation indexes tend to be large
  - Binding Engine (WWW 2005): 10x index size blowup with only a handful of entity types
  - Our target: 18000 atypes today, more later
- Workload-driven index and query optimization
  - Exploit skew in query atype workload

Part 1: Scoring and Ranking Nodes in Graphs

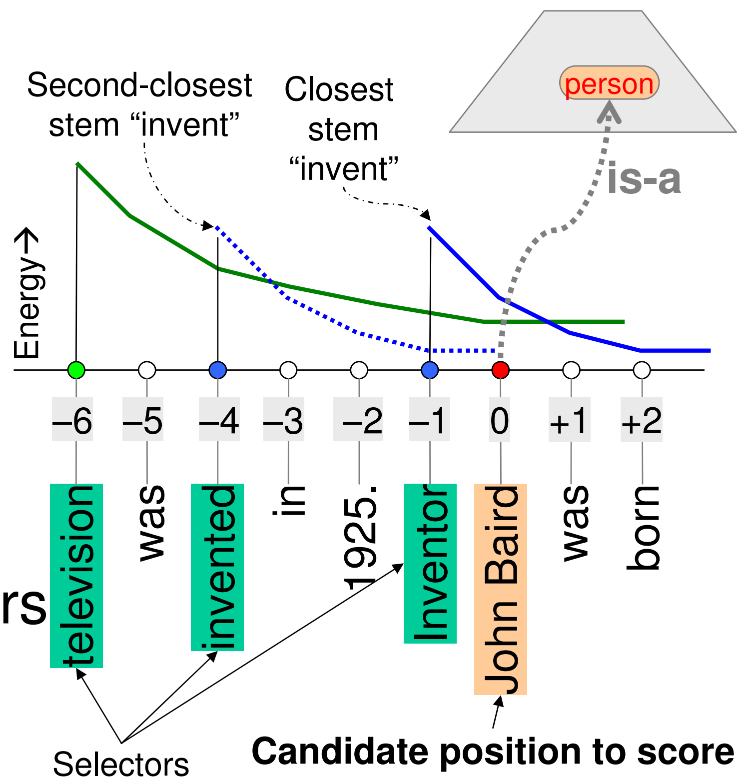
Two flavors of ranking problems

- The restricted query class we just discussed
  - 0/1 type membership via “perfect” taxonomy
  - NEAR captured via token rareness and distance between match tokens and candidate token
- General typed entity-relationship (ER) graph
  - Typed edges and nodes with text
  - Random walk biased by
    - Query matching node text
    - Semantics of edge types
  - Learn walk parameters, don’t guess them



Learning to score token spans

- type=person NEAR “television” “invent*”
- Rarity of selectors
- Distance from candidate position to selectors
- Many occurrences of one selector
  - Closest is good
- Combining scores from many selectors
  - Sum is good



Learning the shape of the decay function

- For simplicity assume left-right symmetry
- Parameters $(\beta_1,\ldots,\beta_W)$, W=max gap window
- Candidate position characterized by a feature vector $f = (f[1],\ldots,f[W])$
  - If there is a matched selector s at distance j and
  - This is the closest occurrence of s
  - Then set $f[j]$ to *energy*(s), … else 0
- Score of candidate position is $\beta \cdot f$
- If we like candidate u less than v ("$u \prec v$")
  - We want $\beta \cdot f_u \le \beta \cdot f_v$

Ranking feature vectors

- "Hard margin" version

$$\min_{\beta \in R^d} \underbrace{\beta' \beta}_{\text{Regularizer}} \text{ subject to } \beta' f_i - \beta' f_j \le -1 \text{ for all } i \prec j$$

- "Soft margin" version

$$\min_{\beta \in R^d, s \ge 0} \beta' \beta + B \sum_{i \prec j} s_{ij} \text{ subject to } \beta' f_i - \beta' f_j \le -1 + \underbrace{s_{ij}}_{\text{Slack variable}} \text{ for all } i \prec j$$

- Quadratic program, slow (watch KDD 2006)
- By eliminating slack vars, can be rewritten as

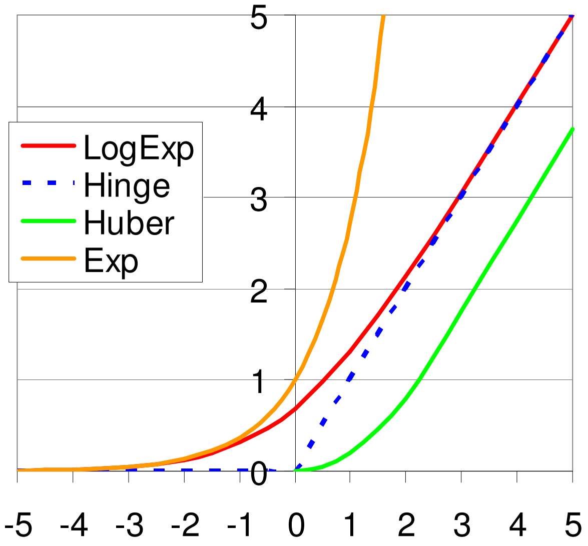
$$\min_{\beta \in R^d} \beta' \beta + B \sum_{i \prec j} \underbrace{\max\{0, 1 + \beta' f_i - \beta' f_j\}}_{\text{Approximate with a smooth function}}$$

Benign loss functions for scoring

- Replace hinge with

$$\min_{\beta \in R^d} \beta' \beta + B \sum_{i \prec j} \boxed{\text{smoothLoss}} \{0, 1 + \beta' f_i - \beta' f_j\}$$

- Differentiable everywhere, use Newton's method
  - Minutes instead of hours
- Can shift and scale β without changing rank
- Can set $\beta_{W+1}=0$ and discourage adjacent β's from differing too much
- Force monotonic decrease (not good)

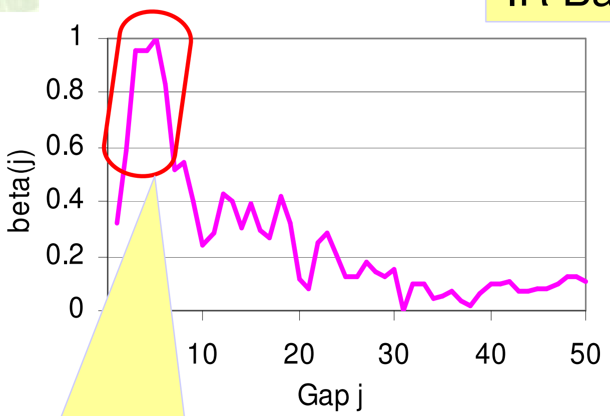


Learning decay function—results

$$\min_{\beta} \sum_{j=1}^{W} (\beta_j - \beta_{j+1})^2 + B \sum_{u \prec v} \text{smoothLoss}(\beta \cdot f_u - \beta \cdot f_v)$$

Discourage adjacent βs from differing a lot

Penalize violations of preference order



Roughly unimodal around gap = 4 and 5

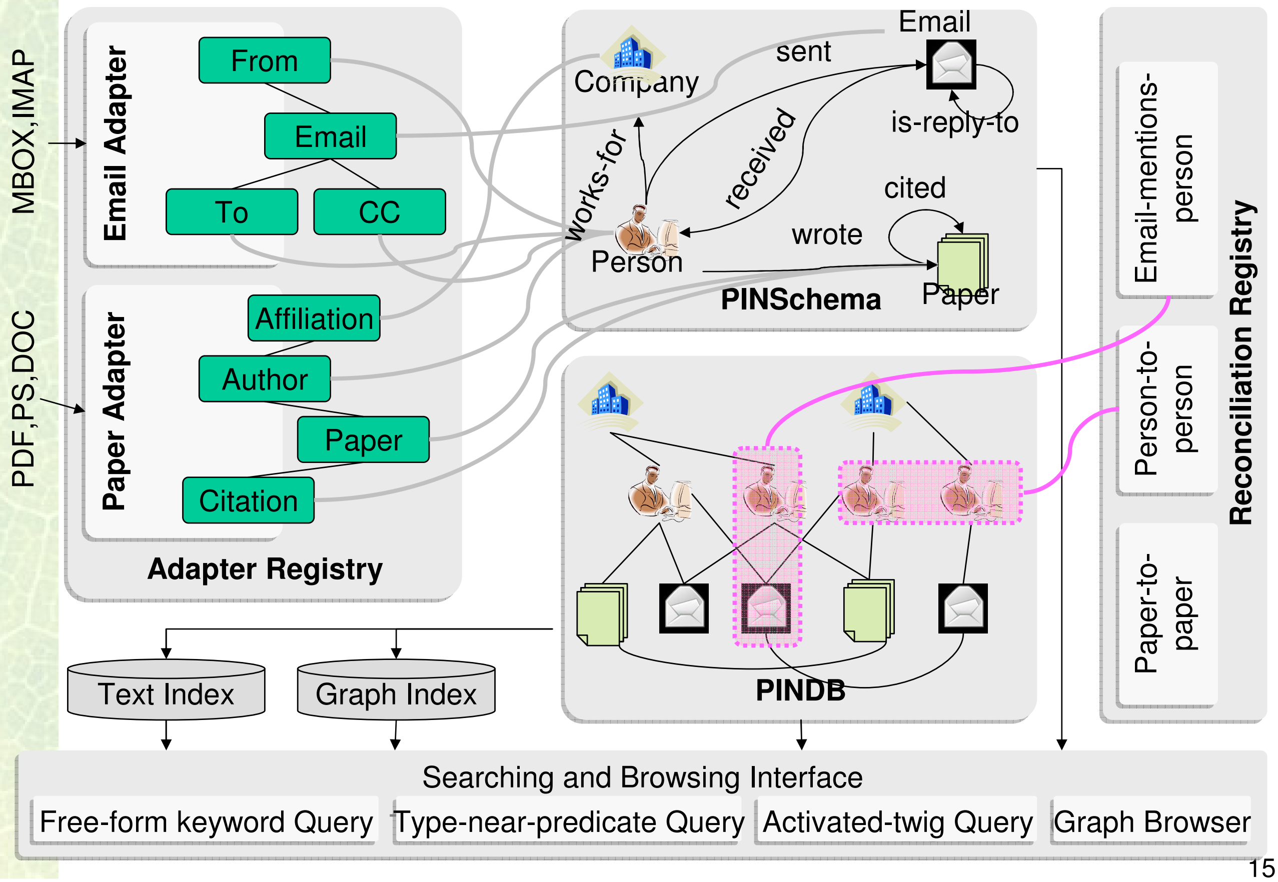
IR Baseline

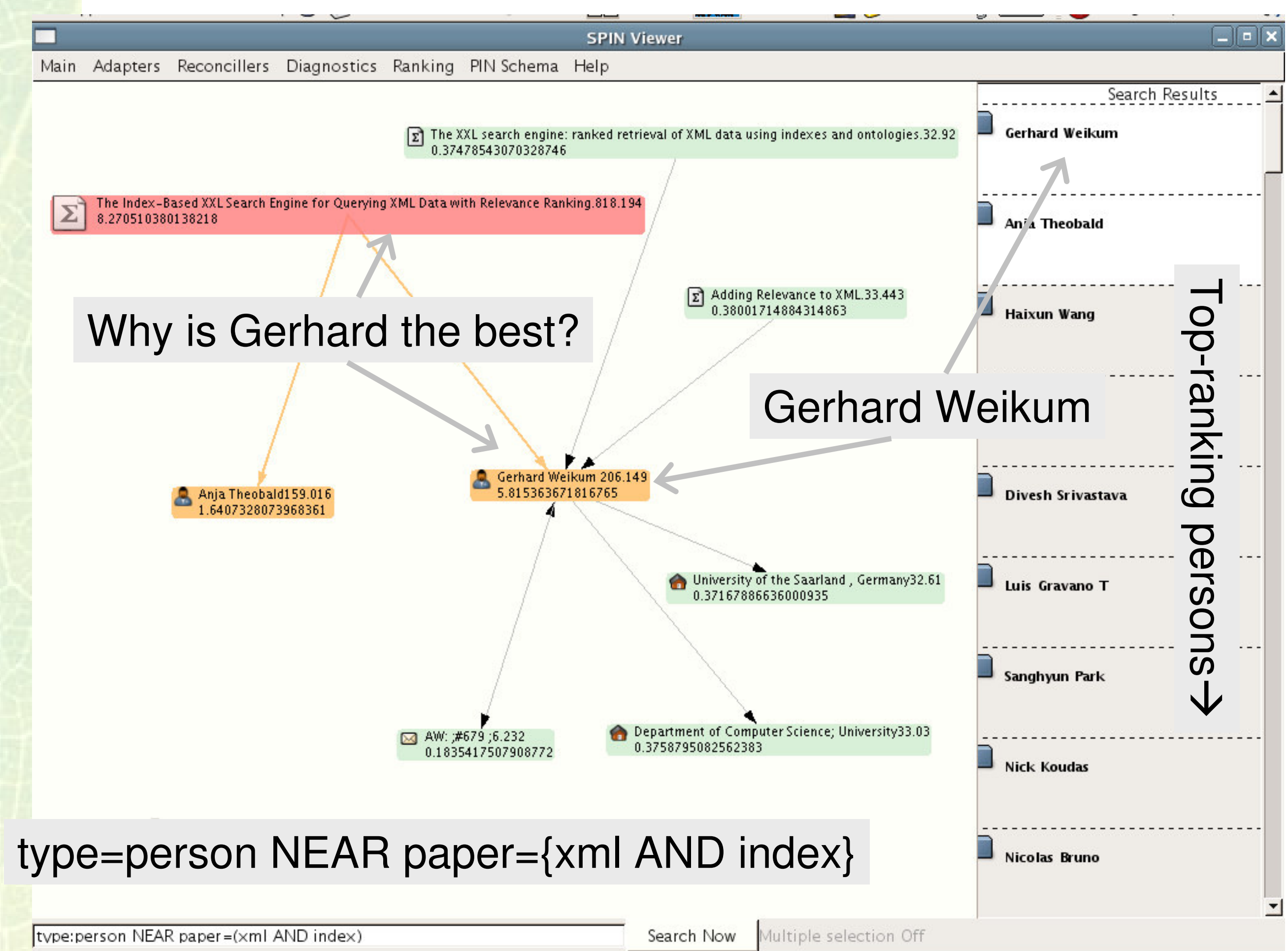
Train	Test	MRR
IR	2000	0.16
2001	2000	**0.29**

TREC year

Mean reciprocal rank: Average over questions, reciprocal of the first rank where an answer token was found (large good)

Searching personal information networks

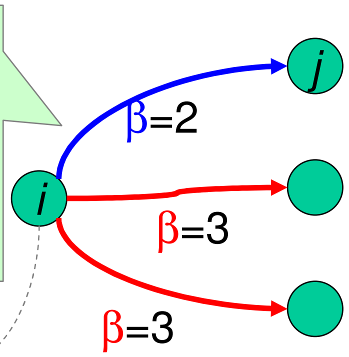




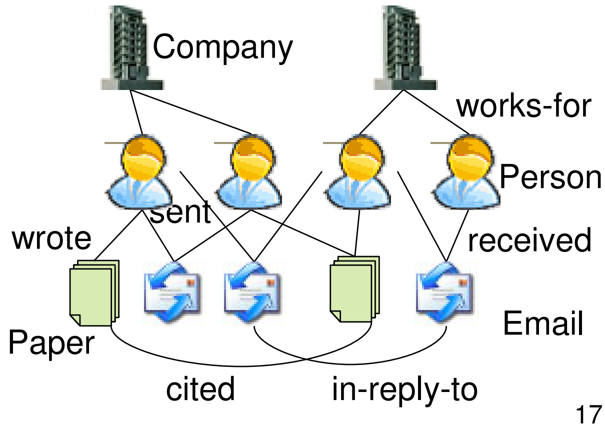
Ranking nodes in ER graphs

- Nodes have entity types: Person, Paper, Email, Company
- Edges have relation types: wrote, sent, cited, in-reply-to; edge *e* has type $t(e) \in \{1, \ldots T\}$
- Edge *i*→*j* of type *t* has *weight* $\beta(t)$ and *conductance* $C(i \rightarrow j)$…

Probability of following blue edge out of *i* is 2/(2+3+3)



Teleport?



Edge conductance

Dead-end

Ordinary walk step

$$C(j,i) = \begin{cases} 0, & i \neq d, j \neq d, i \in \mathrm{leaf}(V) \\ \alpha \frac{\beta(t(i,j))}{\sum_{j'} \beta(t(i,j'))}, & i \neq d, j \neq d, i \notin \mathrm{leaf}(V) \\ 1, & i \neq d, j = d, i \in \mathrm{leaf}(V) \\ 1-\alpha, & i \neq d, j = d, i \notin \mathrm{leaf}(V) \\ \boxed{r_j} & i = d, j \neq d \\ 0, & i = d, j = d \end{cases}$$

Teleport from dummy node to ordinary nodes

Teleport from ordinary nodes to dummy node

Constrained design of conductance

- Hard constraints

Scaling all β preserves **p**, so we can demand all β(t)≥1

$$\min_{\beta \geq \mathbf{1}} \mathrm{ModelCost}(\beta)$$

subject to :

$$\mathbf{p} = \mathbf{C}(\beta)\,\mathbf{p}$$

$$p_i \leq p_j \text{ for all } i \prec j$$

Must break this recurrence between **p** and **p** to solve directly

- Most parsimonious model?
  - All β(*t*)=1:

$$\mathrm{ModelCost}(\beta) = \sum_t (\beta(t) - 1)^2$$

  - All β(*t*) equal:

$$\mathrm{ModelCost}(\beta) = \sum_{t \neq t'} (\beta(t) - \beta(t'))^2$$

No margin!? –Because an arbitrary margin (say 1) may never be attainable by deviating from the parsimonious model and scaling β (unlike RankSVM)

Breaking the **p=Cp** recurrence

- Pagerank is usually approximated by the Power Method: $\mathbf{p} \approx \mathbf{C}^H \mathbf{p}^0$ where
  - *H* is a large enough horizon to give convergence
  - $\mathbf{p}^0$ is an initial distribution over nodes, usually uniform
- Compute alongside Pagerank (chain rule):

$$\frac{\partial}{\partial \beta_t}(C^0 p^0)_i = 0 \quad \text{for all } t \text{ and } i,$$

and for $h = 1, \ldots, H$:

$$(C^h p^0)_i = \sum_j C(i,j)\,(C^{h-1} p^0)_j$$

$$\frac{\partial}{\partial \beta_t}(C^h p^0)_i = \sum_j \left[\frac{\partial C(i,j)}{\partial \beta_t}(C^{h-1}p^0)_j + C(i,j)\frac{\partial}{\partial \beta_t}(C^{h-1}p^0)_j \right]$$

Setting up the optimization

- Objective

$$\min_{\beta \geq \mathbf{1}} \sum_{t \neq t'} (\beta(t) - \beta(t'))^2 + B \sum_{i \prec j} \text{huber}\left((C^H p^0)_i - (C^H p^0)_j\right)$$

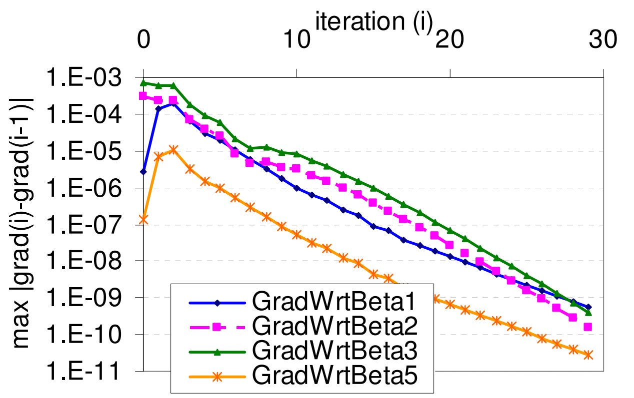
- Gradient of the loss part

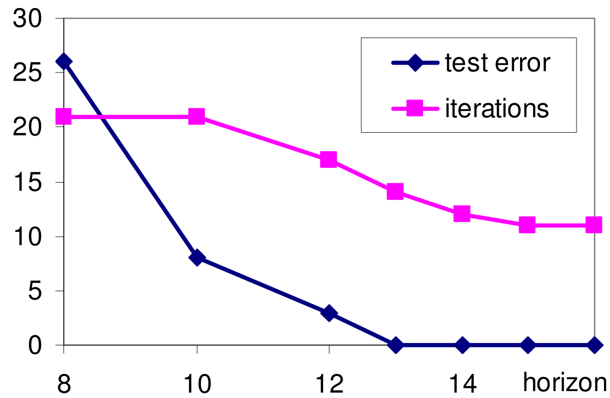
$$\sum_{i \prec j} \text{huber}'\left((C^H p^0)_i - (C^H p^0)_j\right)\left(\frac{\partial (C^H p^0)_i}{\partial \beta(t)} - \frac{\partial (C^H p^0)_j}{\partial \beta(t)}\right)$$

- Polynomial ratios and products—surface not monotonic or unimodal, need some grid search

The effect of a limited horizon

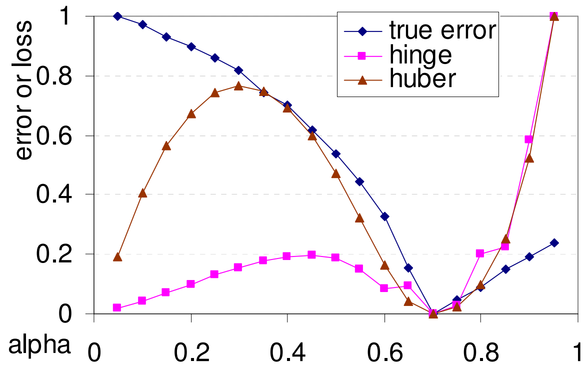
- Gradients also converge, residuals decrease exponentially
  - Not surprising
  - Can perhaps prove given some assumptions
- As H increases
  - More CPU time needed
  - Gradient is more accurate, low test error
  - Fewer Newton iterations needed

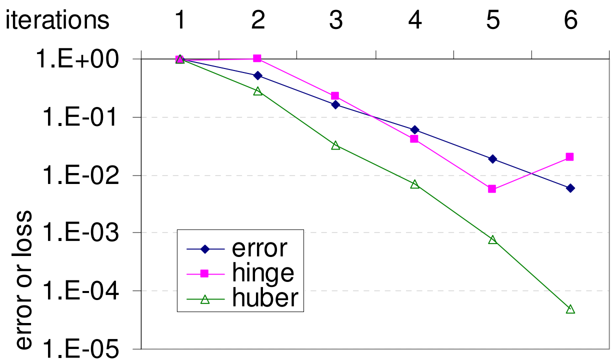




Appropriateness of loss approximation

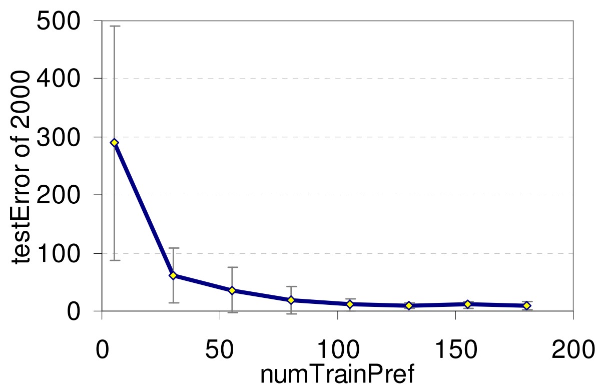
- Less reliable than true error (as usual)
- Hinge loss is even worse than Huber
- "In practice"…
  - β optimization never seems to get trapped in local minima
  - α optimization is started from a 0:0.1:1 grid
- Need better understanding of the optimization surface

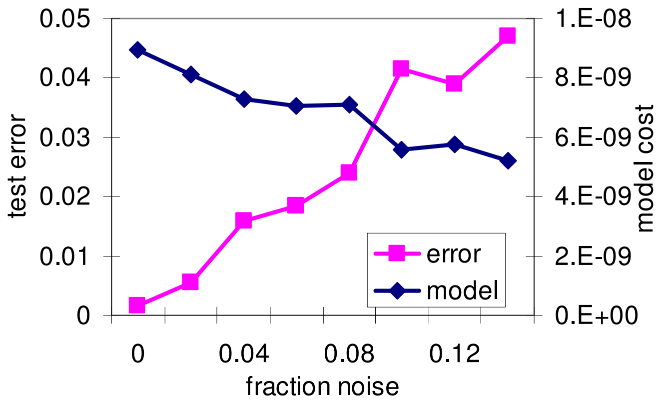




Learning rate and robustness

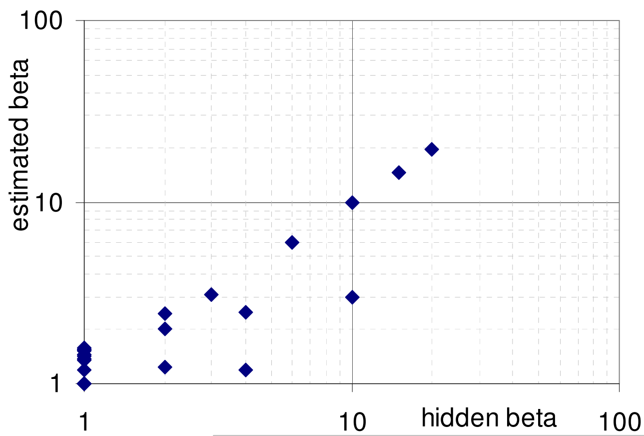
- 20000-node, 120000-edge graph
  - 100 pairwise training preferences enough to cut down test error to 11 out of 2000
  - Careful! Training and test preferences were made node-disjoint
- 20% random reversal of train pairs → 5% increase in test error
  - Model cost reduces

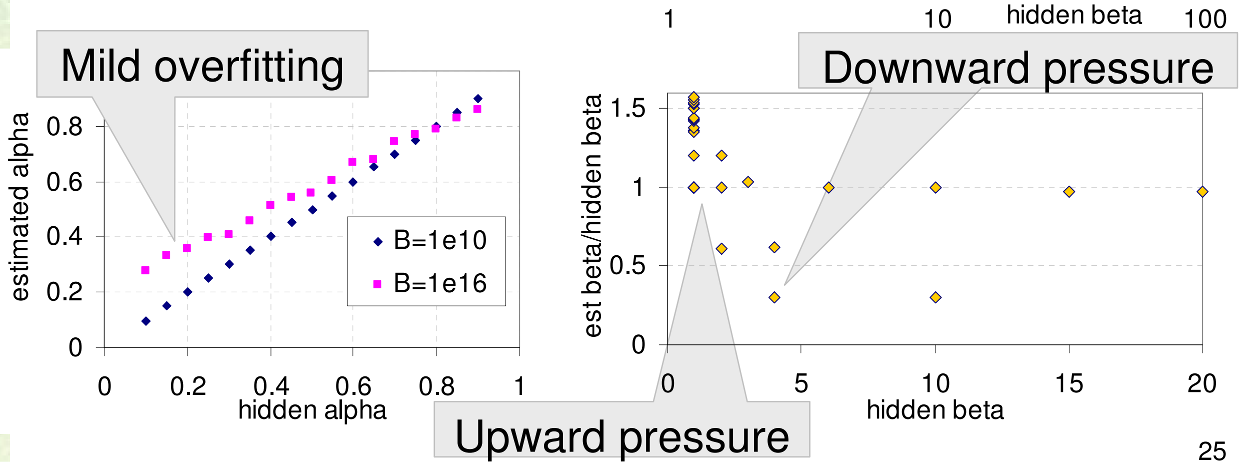




Discovering hidden edge weights

- Assign hidden edge weights to edge types
- Compute weighted Pagerank and sample $\prec$
- See if our algorithm can recover hidden weights
- Likewise with α





Part 1 summary

- Inner product of weights with feature vector
  - A very simple scoring model
  - Still, TFIDF and BM25 evolved over decades
  - Learning weights: very recent, still evolving
- Ranking in graphs increasingly important
  - Pagerank and friends are just version 0.1
- Next step: entity-relationship graphs
  - Nodes and edges have associated types
  - Nodes (possibly edges) have associated text
- Bootstrap ranking wisdom via learning

Part 2: Indexing for Proximity Search

Part-2: Workload-driven indexing

- Type hierarchies are large and deep
  - 18000 internal and 80000 leaf types in WordNet
- Runtime atype expansion time-intensive
  - Even WordNet knows 650 scientists, 860 cities…
- Index each token as all generalizations
  - Sagan → physicist, scientist, person, living thing
  - Large index space bloat

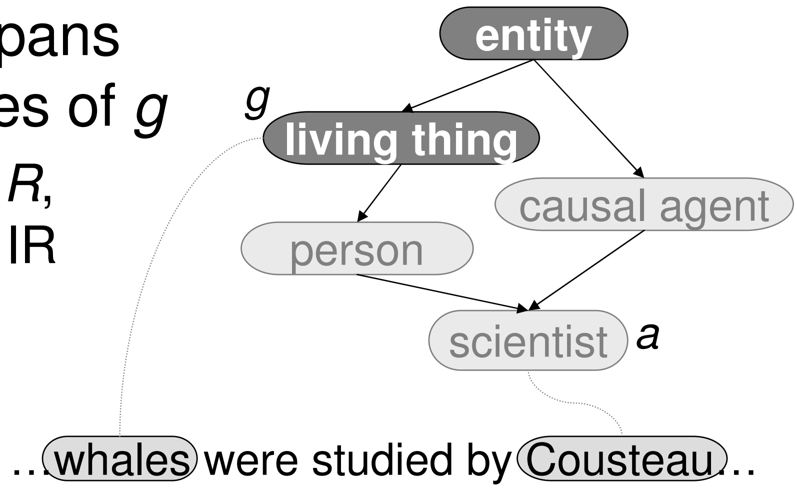
Index a subset of atypes

Corpus/Index	Gbytes
Original corpus	5.72
Gzipped corpus	1.33
Stem index	0.91
Full type index	4.30

Pre-generalize (and post-filter)

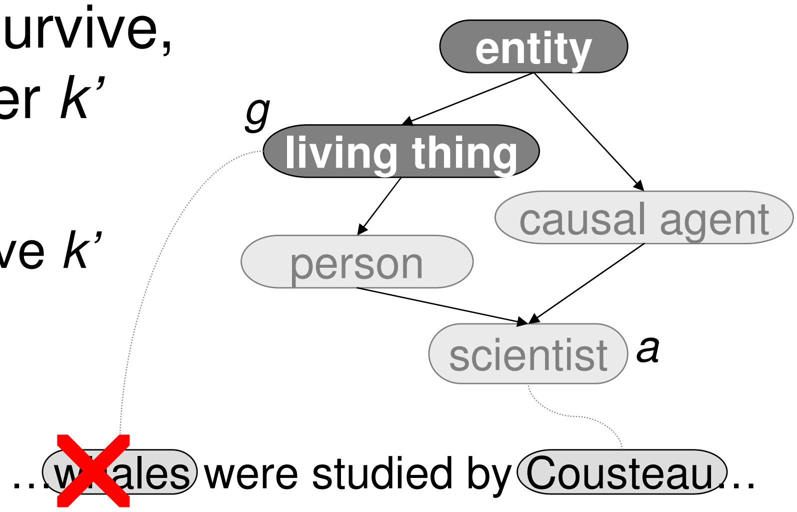
- Full set of "atypes" (answer types) is A
- Index only a "registered" subset R of A
- Say query has atype a; want k answers
- Find a's "best" generalization $g \in R$
- Get best $k' > k$ spans that are instances of g
  - Given index on R, this is standard IR (see paper)

"Which scientist studied whales?



(Pre-generalize and) post-filter

- Fetch each high-scoring span w
- Check if w is-a a
  - Fast compact "forward index" (doc,offset)→token
  - Fast small "reachability index", common in XML
- If fewer than k survive, restart with larger k'
  - Expensive
  - Pick conservative k'



Estimates needed by optimizer

- If we index token ancestors in *R* as against ancestors in all of *A*, how much index space will we save?
  - Cannot afford to try out and see for many *R*s
- If query atype *a* is not found in *R* and we must generalize to *g*, what will be the bloat factor in query processing time?
  - Need to average over a representative workload

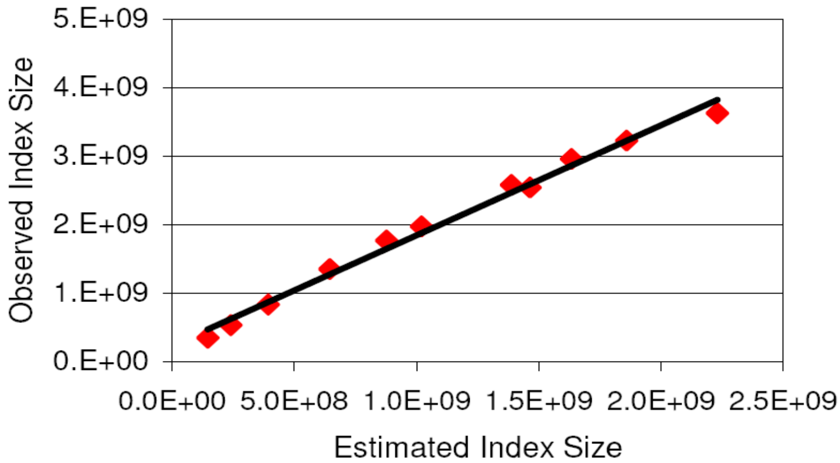
Index space estimate given R

- Each token occurrence leads to one posting entry
- Assume index compression is a constant factor
- Then total estimated index size is proportional to

$$\sum_{r \in R} corpusCount(r)$$

Number of tokens in corpus that connect up to *r*

- Surprisingly accurate!



Processing time bloat for one query

- If R=A, query takes time approximated by

$$t_{\text{scan}}\ \mathit{corpusCount}(a)$$

Time to score one candidate position while scanning postings

Number of occurrences of descendants of type a

- If a cannot be found in R, the price paid for generalization to g consists of
  - Scanning more posting entries: $t_{\text{scan}}\ \mathit{corpusCount}(g)$
  - Post-filtering k' responses: $k'\,t_{\text{filter}}$

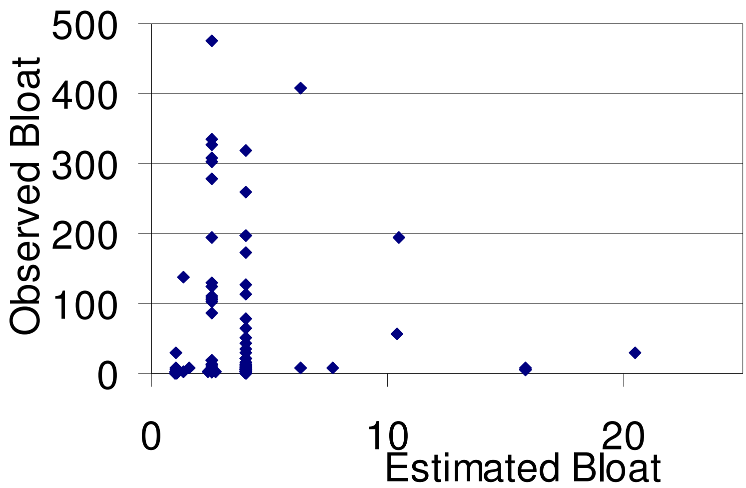
Time to check if answer is instance of a as well

- Therefore, overall bloat factor is

$$\mathit{queryBloat}(a,R) = \frac{t_{\text{scan}}\,\mathit{corpusCount}(g) + k'\,t_{\text{filter}}}{t_{\text{scan}}\,\mathit{corpusCount}(a)}$$

Query time bloat—results

- Observed bloat fit not as good as index space estimate



- While observed::estimated ratio for one query is noisy, average over many queries is much better

Expected bloat over many queries

$$\sum_{a \in A} queryProb(a)\ queryBloat(a, R)$$

Prob of new query having atype *a*

Already estimated

- Maximum likelihood estimate

$$queryProb_{\text{Train}}(a) = \frac{queryCount_{\text{Train}}(a)}{\sum_{a' \in A} queryCount_{\text{Train}}(a')}$$

- Many *a*'s get zero training probability
→ Optimizer does not register *g* close to *a*
- Low-prob atypes appear in test → huge bloat
- Collectively matter a lot (heavy-tailed distrib)

Smoothing low-probability atypes

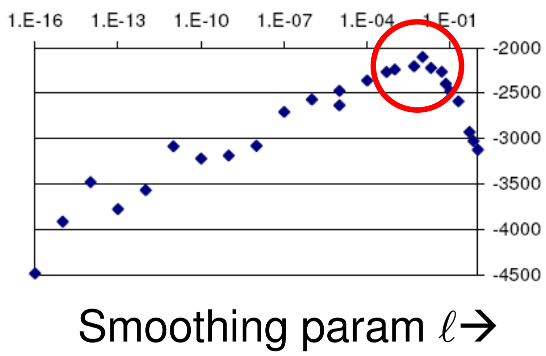
- Lidstone smoothing:

$$queryProb_{\text{Train}}(a) = \frac{queryCount_{\text{Train}}(a) + \ell}{\sum_{a' \in A} \left(queryCount_{\text{Train}}(a') + \ell\right)}$$

- Smoothing param ℓ fit by maximizing log-likelihood of held-out data:

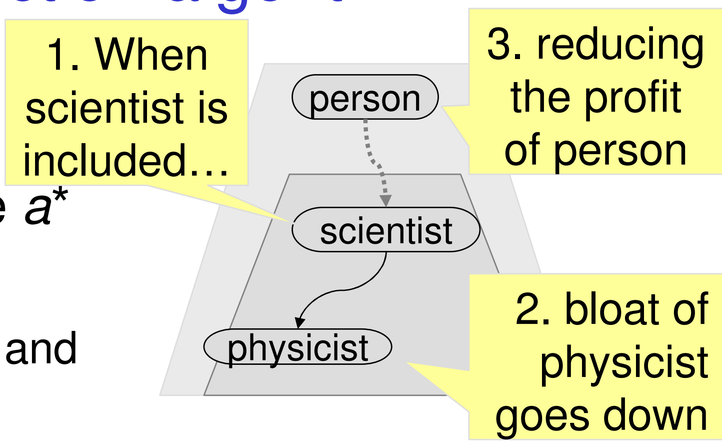
$$\sum_{a \in \text{HeldOut}} queryCount_{\text{HeldOut}}(a)\ \log(queryProb_{\text{Train}}(a))$$

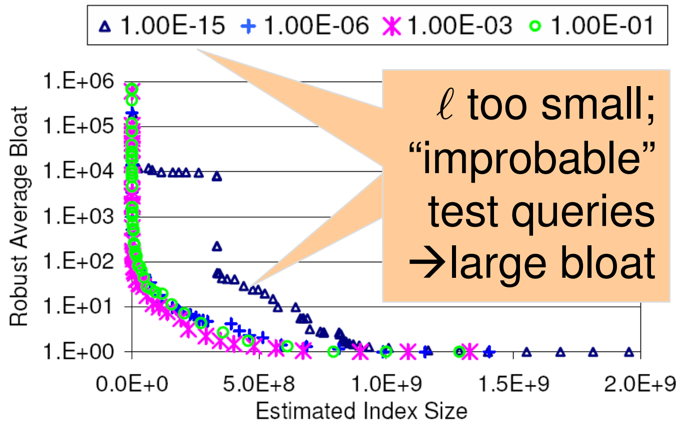
- Clear range of good fits for ℓ
- Can probably do better



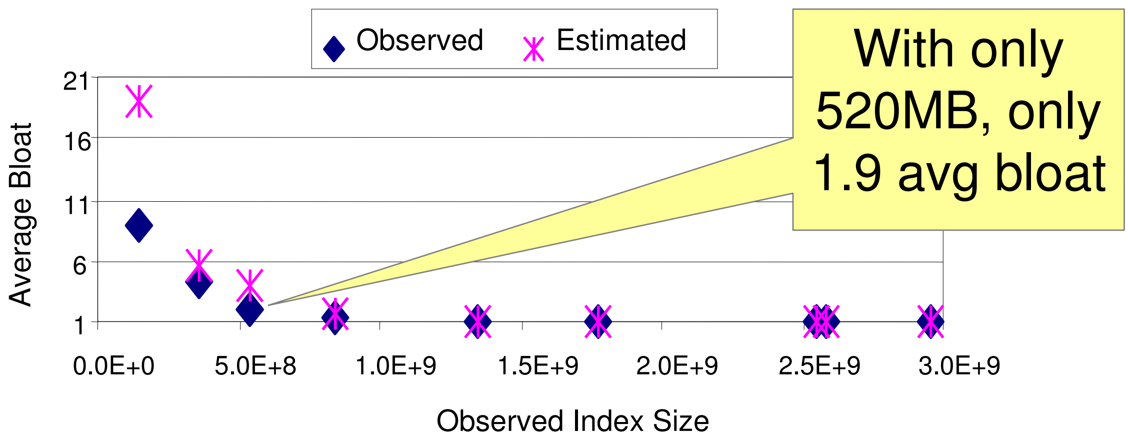
The *R* selection algorithm

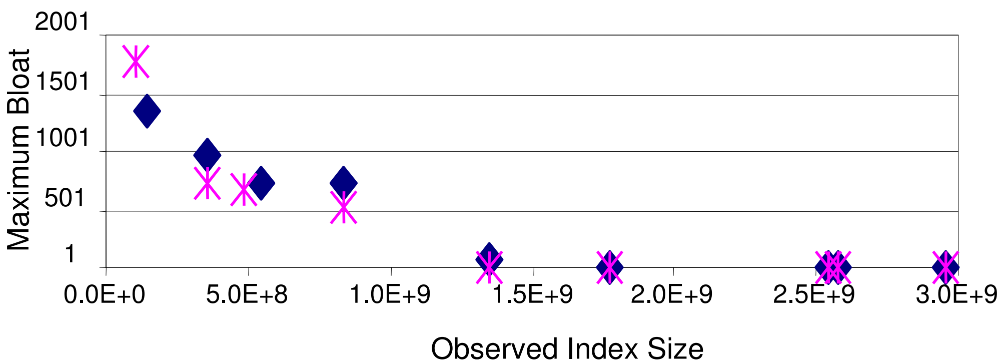
- $R \leftarrow$ roots of A
- Greedily add the "most profitable" atype a^*
- Profit = ratio of
  - reduction in bloat of a^* and its descendants to
  - increase in index space
- Downward and upward traversals and updates
- Gives a tradeoff between index space and query bloat





Optimized space-time tradeoff





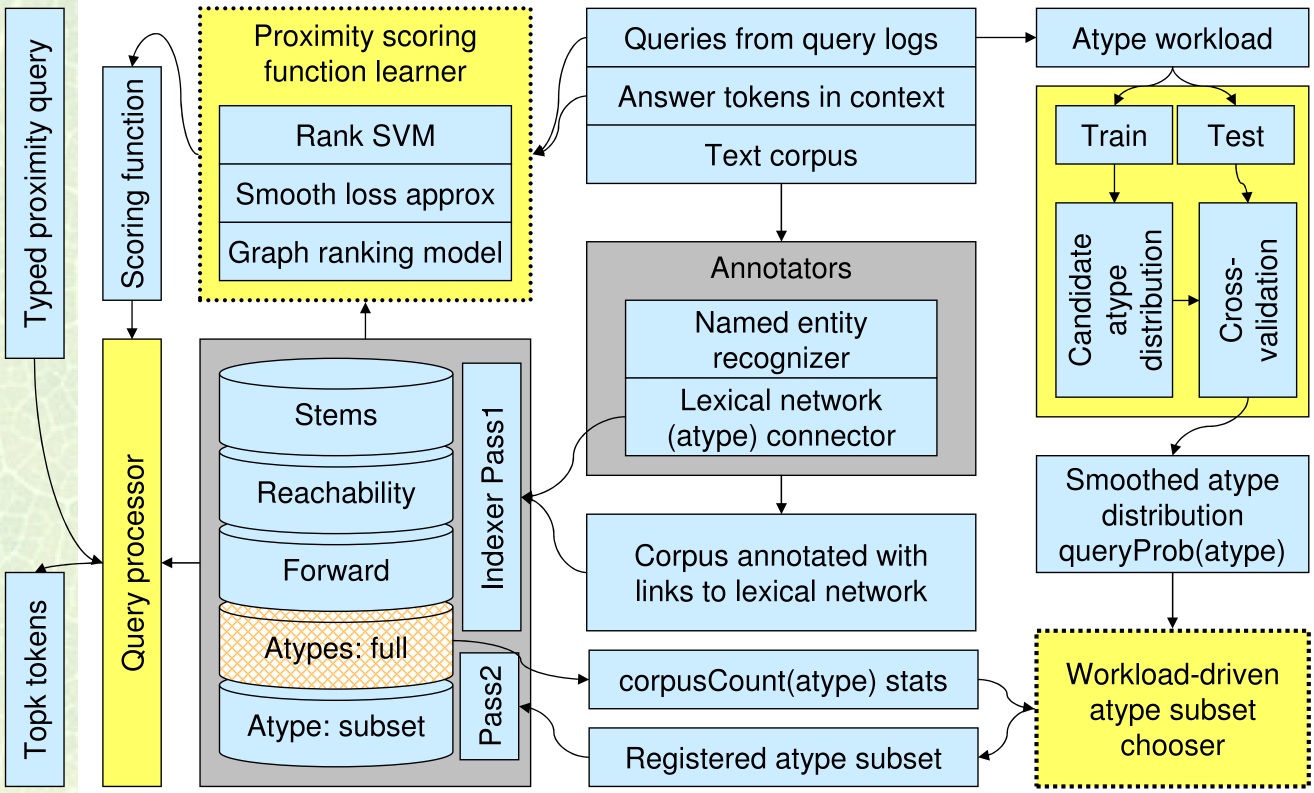
Optimized index sizes

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

Part 2 summary

- Working prototype around Lucene and UIMA
  - Annotators attach tokens to type taxonomy
  - Query atype workload help compact index
  - Ranking function learnt from preference data
  - NL queries translated into atype+selectors
- Ongoing work
  - Indexing and searching relations other than is-a
  - More general notions of graph proximity
- Email soumen@cse.iitb.ac.in for code access

The big picture



Email soumen@cse.iitb.ac.in for code access

Conclusion

- Perform limited pre-structuring of corpus
  - Difficult to anticipate all query needs
  - Attach graph structure where possible
  - Do not insist on specific schema
  - Partial structure and raw text coexist
- Exploit statistical tools on graph models
  - Models of influence along links getting clearer
  - What is a query? What is a response?
  - Ranking in graphs still under-explored
  - Scalable indexing and “top-k” query execution are major challenges