Extracting, Searching and Mining Semantic Annotations on the Web

Tutorial T10-F, WWW 2009

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

April 15, 2009

High-level TOC

Annotation:

- Identifying token segments as entity mentions
- Ditto for relations
- Annotation vs. extraction
- Closed vs. open domain

Disambiguation:

- Supervised: against entities in a catalog
- Unsupervised: cluster mentions referring to same entity

Search:

- Relational
- XML, trees, twigs
- Proximity in text and general graphs

Timeline view

1992 Hearst patterns

1998 DIPRE (duality in pattern-relationship extraction)

2000 Snowball = DIPRE + confidence scores

2001 Turney's PMI

2001 SemTag and Seeker

2000–now Many systems for labeling token spans or 2d regions with entity types

2004 KnowItAll ≈ Hearst patterns + list extraction + a few more tricks

2007 TextRunner and ExDB

Supervision view

Detailed, domain-specific supervision: All sequential and tree labeling tasks

Limited domain, query by example:

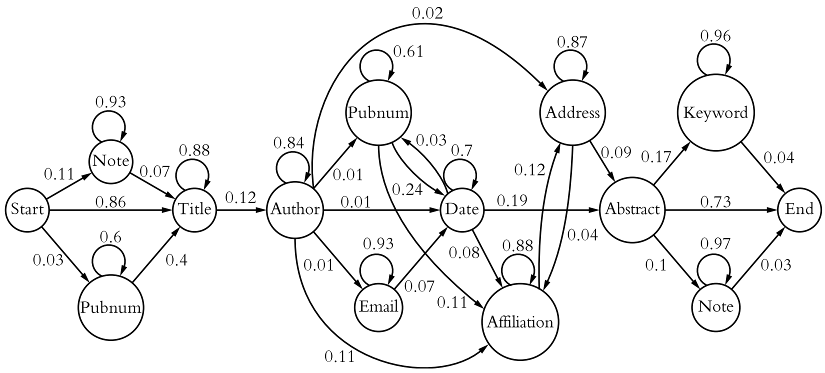
- Fixed is-a relation: KnowItAll [8]
- Broader set of binary relations: DIPRE [2], Snowball [3]

Open-domain, only linguistic supervision: Arbitrary unnamed relations: TextRunner [9]

Tagging entities

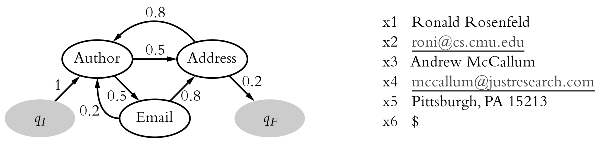
- Restricted set of types or domains
- E.g., person, place, organization, date, paper title, conference venue, journal name, page range, number and street name
- *Closed domain* means positive and negative examples available for each type
- Test data is unlabeled token sequences
- The job is to mark some token segments with one or the trained types

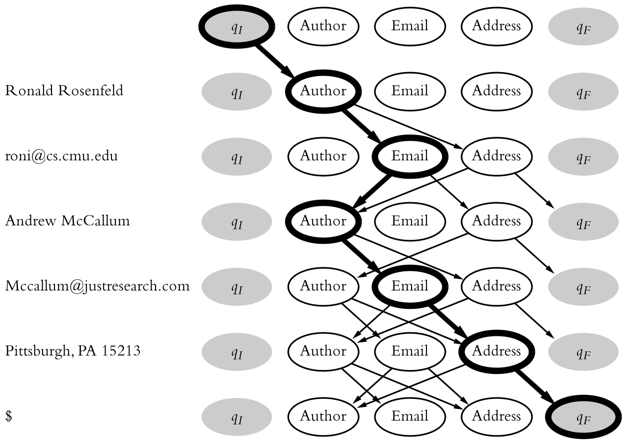
Markov models



- System is in one of M states y_t
- In each state, emits x_t one of N symbols
- Then moves to one of M states y_{t+1}
- Many state transitions disallowed

Viterbi decoding





Limitations of generative models

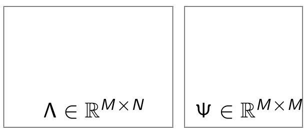
- Would like to think of x_t not as a symbol
- But as a feature vector
- `hasCap`, `isAllCap`, `isXxx`, `hasDigit`, `isAllDigits`, `isDDDD`
  - `isAllCap` $\Longrightarrow$ `hasCap`
  - `isDDDD` $\Longrightarrow$ `isAllDigits`
- Correlated features – same issue as naive Bayes vs. logistic regression
- Prefer to model $\Pr(y|x)$ rather than model $\Pr(x|y)$ and use Bayes rule

Markov sequence features

- $x = (x_t : t = 1, \dots, T)$ is the sequence of visible symbols
- Suppose there are N symbols
- $y = (y_t : t = 1, \dots, T)$ is the label/state sequence
- Suppose there are M states
- Define a feature vector $\phi(x, y) \in \mathbb{R}^d$
- From training data $x^i, y^i : i = 1, \dots, n$, learn a model $w \in \mathbb{R}^d$
- Given test sequence x, predict label sequence $\arg\max_y w^\top \phi(x, y)$

Designing $\phi(x, y)$

- N symbols, M states
- Think of $\phi(x, y)$ as a $M \times (N + M)$ matrix



- $\Lambda[m, n]$ is the number of times symbol n was emitted while in state m
- $\Psi[m, m']$ is the number of times a transition was made from state m to m'
- Model w also has $M(N + M)$ elements

Training w

- For each sequence x^i, there is (say) one correct labeling y^i; all other (exponentially many) labelings $y \neq y^i$ are incorrect
- Want to fit w such that $\forall i, \forall y \neq y^i: \; w^\top \phi(x^i, y^i) > w^\top \phi(x^i, y)$
- The worse y is compared to y^i the bigger we want the gap to be
- Loss function $\Delta(y^i, y) \geq 0$; $\Delta(y, y) = 0$
- E.g., Hamming loss
- $\forall i, \forall y \neq y^i: \; w^\top \phi(x^i, y^i) \geq w^\top \phi(x^i, y) + \Delta(y^i, y)$

B-I-O state model

- Our definition of Λ limits interaction between x and y to individual positions, e.g., x_t and y_t
- But this can be easily extended: we can define features between x_{t-1} and y_t, or x_{t+1} and y_t
- States none, person, place, epoch
  - Roy went to Boston in 1994
- "at" or "to" often precede place, "on" or "in" often precede epoch
- But the state of at, to, on, in are all 'none'

Pattern-relationship duality

- Start with a small table of known facts

Isaac Asimov	The Robots of Dawn
David Brin	Startide Rising
James Gleick	Chaos: Making a New Science
Charles Dickens	Great Expectations
William Shakespeare	The Comedy of Errors

- Find mentions of known authors and books in the corpus:

  > *The Robots of Dawn is a "whodunit" science fiction novel by Isaac Asimov, first published in 1983. It is part of Asimov's Robot series.*

- Induce and evaluate patterns on known data
  - `*prefix, author, middle, title, suffix*`

Pattern-relationship duality (2)

  - `<LI><B>title</B> by author (`
  - `<i>title</i> by author (`
- Find matches to patterns over corpus (scan/index?)
- Import confident extractions into database
- Rinse and repeat
- Brin bootstrapped as follows: 5 facts → 199 occurrences → 3 patterns → 4047 proposed facts → 105 more patterns → 9369 proposed facts
- Quality control needed

More pattern examples

Organization e_1	Location of headquarters e_2
Microsoft	Redmond
Exxon	Irving
IBM	Armonk
Boeing	Seattle
Intel	Santa Clara

- e_1's headquarters in e_2
- e_2-based e_1
- e_1, e_2

Generic bootstrapping pseudocode

1: input seed tuples $\{(e_{i1}, e_{i2}), i = 1, \ldots, n\}$
2: **while** database not big enough **do**
3: find snippets in corpus where seed tuples are mentioned
4: tag entities in snippets
5: generate new patterns L, t_1, C, t_2, R or L, t_2, C, t_1, R
6: apply new patterns over whole corpus
7: import newly extracted tuples into database

- Which patterns are sufficiently reliable and useful?
- Which extracted tuples are likely to be correct?

How to represent patterns

Organization $e_1 \in t_1$	Headquarter location $e_2 \in t_2$
Microsoft	Redmond
Exxon	Irving
IBM	Armonk
Boeing	Seattle
Intel	Santa Clara

- $L = \{(\texttt{the}, 0.2)\}$
- $t_1 = \textit{organization}$
- $C = \{(\text{-}, 0.5), (\texttt{based}, 0.5)\}$
- $t_2 = \textit{location}$
- $R = \{\}$

How to induce and evaluate patterns

- Pattern generated by clustering contexts of known facts
- Keep a held-out set of verified true and false is-a statements
- Apply pattern P on held-out statements
- Use some notion like precision or F_1 as pattern confidence Conf(P), e.g.,

$$\text{Conf}(P) = \frac{n^+}{n^+ + n^-},$$

where n^+ (n^-) is the number of known true (false) statements matched by P

Pattern match

- Two patterns $P_1 = (L_1, a_1, C_1, b_1, R)$ and $P_2 = (L_2, a_2, C_2, b_2, R_2)$ have nonzero match only if $a_1 = a_2$ and $b_1 = b_2$
- In that case, the match score is

$$\mathrm{Match}(P_1, P_2) = L_1 \cdot L_2 + C_1 \cdot C_2 + R_1 \cdot R_2$$

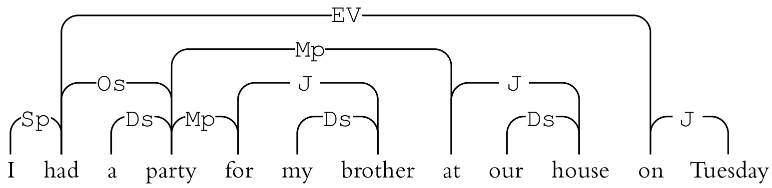
  i.e., sum of dot products of word bags
- Can scale to make it a belief between 0 and 1, or threshold it

Confidence of match

$$1 - \prod_P \big(1 - \mathrm{Conf}(P)\,\mathrm{Match}(P, S)\big)$$

- P is a pattern
- S is a snippet containing $e_1 \in t_1, e_2 \in t_2$
- A soft-or formulation
- Can iterate between match and pattern confidence using EM

Dependency parsing



- Links tokens in sentence
- Edges encode purpose of syntax
- Often used for slot filling [14]:
  - Which company bought out which company?
  - For how much money?
  - How much in stocks vs. cash?
  - Where did the accident happen?
  - How many were injured?

Shortest path kernel

Visual/Xpath extraction

High-level TOC

Annotation:

- Identifying token segments as entity mentions
- Ditto for relations
- Annotation vs. extraction
- Closed vs. open domain

Disambiguation:

- Supervised: against entities in a catalog
- Unsupervised: cluster mentions referring to same entity

Search:

- Relational
- XML, trees, twigs
- Proximity in text and general graphs

Hearst patterns

- "The bow lute, such as the Bambara ndang, is plucked and has an individual curved neck for each string."
  - What is a Bambara ndang?
  - Can answer without knowing what a bow lute is!
- Source pattern: `NP0 such as` $\{$`NP1, NP2 ... , (and|or)`$\}$ `NPn`
- Target inference: for all NPi, $i = 1, \ldots, n$, `hyponym(NP`i`, NP0)`

More patterns

- `such NP as NP, * (or|and) NP`
  - "... works by such authors as Herrick, Goldsmith, and Shakespeare."
  - Lets us infer `hyponym(''author'', ''Herrick'')`, `hyponym(''author'', ''Goldsmith'')`, `hyponym(''author'', ''Shakespeare'')`
- `NP , NP* , (or|and) other NP`
- `NP , (especially|including) NP ,*(or|and) NP`
- `NP1 is a NP2`

Caveats

- Garth Brooks is a country
- gift such as wall
- person like Paris

Caveats

- Garth Brooks is a country singer
- gift such as wall
- person like Paris

Caveats

- Garth Brooks is a country singer
- gift such as wall clock
- person like Paris

Caveats

- Garth Brooks is a country singer
- gift such as wall clock
- person like Paris Hilton

Caveats

- Garth Brooks is a country singer
- gift such as wall clock
- person like Paris Hilton

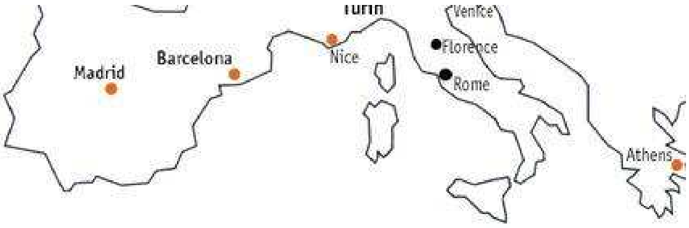
- Need noun phrase chunking, e.g. "country singer" or "wall clock"
- Also need disambiguation, because "Hilton" may not be mentioned explicitly
- Cannot depend on simple phrase queries on a text index

KnowItAll

- Specific binary relation: is-a
- Use Hearst patterns to get high-confidence facts
- Pick random subsets (say, of size 3–4)
- Ask Web query with subset
- Locate HTML lists mentioning most elements of subset
- Judge if list is of entities of same type (how?)
- Extract other candidate entities from list
- Use global stats to validate candidates

List/table extraction

airport because it is near the city center. For the intercontinental arrivals, the airport offers direct flights to the main European Hubs:



Direct Flights from/to Torino - Caselle International Airport

City	Flight hours	Company	Frequency	Fare (Euro)
Amsterdam	2.00	KLM	Daily	350/670
Barcelona	2.00	Iberia	Daily	300/860
Brussels	1.40	Sabena	Daily	300/750
Dusseldorf	2.00	Lufthansa	Daily	320/810
Frankfurt	1.15	Lufthansa	Daily	320/760
Lisbon	3.30	Air Portugal	No Saturday	390/1050
London	2.00	Ryan Air	Daily	Only on line
Madrid	2.10	Iberia	Daily	300/630
Munich	2.00	Lufthansa	Daily	300/690

Open information extraction[1]

- Traditional heavily-supervised IE is narrow
- Applied to small, homogeneous corpora
- On the Web
  - No parser is sufficiently accurate
  - No pre-trained named-entity taggers
  - Supervised learning is impractical
- Semisupervised learning needs a few hand-labeled examples per concept
- Concepts themselves are pre-specified

[1] From Etzioni slides at WSDM 2008

TextRunner[2]

- Use a simple model of relationships in English to label extractions
- Bootstrap a general model of relationships in English sentences, encoded as a CRF
- Using a shallow parser, decompose each sentence into one or more (`NP1`, `VP`, `NP2`) chunks
- Use CRF model to retain relevant parts of each NP and VP

[2] From Etzioni slides at WSDM 2008

TextRunner example[3]

- Extract Triple representing binary relation (`Arg1`, `Relation`, `Arg2`) from sentence.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- RDF-like triple (`Ebay`, `Founded by`, `Pierre Omidyar`)

[3] From Etzioni slides at WSDM 2008

TextRunner example[3]

- Extract Triple representing binary relation (`Arg1`, `Relation`, `Arg2`) from sentence.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- RDF-like triple (`Ebay`, `Founded by`, `Pierre Omidyar`)

[3]From Etzioni slides at WSDM 2008

TextRunner example[3]

- Extract Triple representing binary relation (`Arg1`, `Relation`, `Arg2`) from sentence.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- RDF-like triple (`Ebay`, `Founded by`, `Pierre Omidyar`)

[3]From Etzioni slides at WSDM 2008

TextRunner example[3]

- Extract Triple representing binary relation (`Arg1`, `Relation`, `Arg2`) from sentence.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- Internet powerhouse, EBay, was originally founded by Pierre Omidyar.
- RDF-like triple (`Ebay`, `Founded by`, `Pierre Omidyar`)

[3] From Etzioni slides at WSDM 2008

High-level TOC

Annotation:

- Identifying token segments as entity mentions
- Ditto for relations
- Annotation vs. extraction
- Closed vs. open domain

Disambiguation:

- Supervised: against entities in a catalog
- Unsupervised: cluster mentions referring to same entity

Search:

- Relational
- XML, trees, twigs
- Proximity in text and general graphs

Entity disambiguation

- ... book by Mike Jordan on graphical models ...
- ... chance to see Michael Jordan play without Dean Smith ...
- `http://en.wikipedia.org/wiki/Michael_Jordan` or `http://en.wikipedia.org/wiki/Michael_I._Jordan` or ...?
- Which entity catalog to use? (Wikipedia, TAP, OpenCYC, WordNet, ...)
- What about the many Mike Jordans not in the catalog?
- Different from anaphora: Every dog has *its* day

Some distinctions from WSD

- Word sense disambiguation (WSD) is largely about common words, not references to specific entities
  - 42 senses of "run" in WordNet
  - Part of speech helps a fair bit
- Entity catalog typically richer info source than dictionary
  - Broader category system
  - Part of speech is largely "proper noun" and not as helpful

Why annotate?

- Make raw text look like Wikipedia with definitional and informational links (most systems)
  - Annotate first occurrence only
  - Annotate only on-topic entities
  - Use discretion to avoid "hyperlink fatigue"
- Index the annotations to enable advanced search (our focus)
  - Exhaustive annotation
  - Make no whole-document topic judgment

Notation

- book by Mike Jordan on graphical models and chance to see Michael Jordan play without Dean Smith are spots s
- Mike Jordan and Michael Jordan are mentions, other tokens form context
- γ is an entity ID or label from the catalog
- Goals:
  - Identify that a sequence of tokens is a potential mention
  - Capture suitable context around to form spot s
  - Assign s to a suitable entity γ in catalog
  - Or claim that there is no suitable γ
- n spots on a page

Notation (2)

- y_s is the entity label assigned to spot s
- y is a vector of n label variables
- Γ_s is the set of labels admissible for spot s

Catalog representation

- Pattern after WordNet, Wikipedia, TAP, ...
- Each entity γ as an associated description
- Descriptions link to other related entities γ'
- Entities belong to one or more categories
- Categories (physicist) are subcategories of others (scientist)
- Links may be "incidental"
- Categories and super-categories may be noisy: *Machine learning researcher* more meaningful than *Living people* or *Year of birth missing*
- Cycles in is-a "hierarchy"?

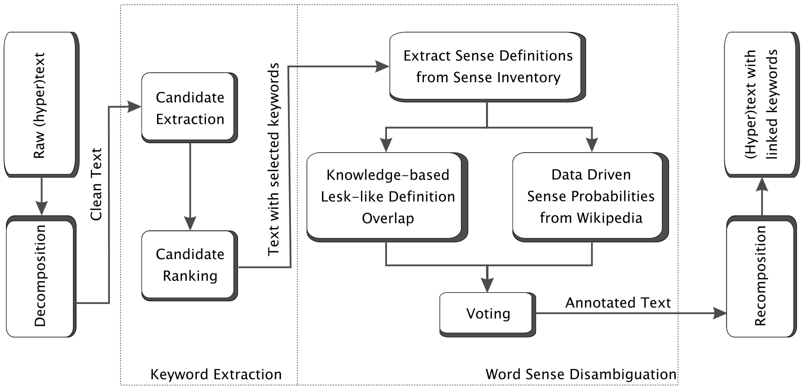
Local compatibility between s and γ

- Feature vector $f_s(\gamma) \in \mathbb{R}^d$ expresses local textual compatibility between (context of) spot s and candidate label γ
- One element of $f_s(\gamma)$ might be the TFIDF cosine similarity between tokens from the context of spot s (say ± 10 tokens) and whole page of description for entity γ
- Another element may be derived of "anchor text" match:
  - Find all links to γ from within Wikipedia
  - Collect anchor text from all these links in a bag of words
  - Find TFIDF cosine similarity between this bag and the spot context s

Node score

- Node scoring model $w \in \mathbb{R}^d$
- Node score defined as $w^\top f_s(\gamma)$
- w is trained to give suitable relative weights to different compatibility measures and aggregate the evidence
- During test time, greedy choice local to s would be $\arg\max_{\gamma \in \Gamma_s} w^\top f_s(\gamma)$
- Early algorithms are variations on this theme

Wikify!



- Two-phase process
- First identify token spans "worthy of annotation"
- Then choose entity labels

Sample annotations

In 1834, Sumner was admitted to the [[bar (law)|bar]] at the age of twenty-three, and entered private practice in Boston.

It is danced in 3/4 time (like most waltzes), with the couple turning approx. 180 degrees every [[bar (music)|bar]].

Vehicles of this type may contain expensive audio players, televisions, video players, and [[bar (counter)|bar]]s, often with refrigerators.

Jenga is a popular beer in the [[bar (establishment)|bar]]s of Thailand

---

This is a disturbance on the water surface of a river or estuary, often cause by the presence of a [[bar (landform)|bar]] or dune on the riverbed.

Choosing token spans to annotate

- Wikify! follows the Wikipedia philosophy
- Use some score to rank candidate spans
- TFIDF of a token in a document
- χ^2 test:

count of token in doc	count of all other tokens in doc
count of token in other docs	count of all other tokens in other docs

- "Keyphraseness" — In how many Wikipedia documents is the same term made a link anchor?
- (They only consider as candidates words which appear at least five times in Wikipedia)

Disambiguation

Wikify! compares two local techniques:

- "Knowledge-based approach" — similarity between Wikipedia page text of entity γ and context words in spot s
- "Data-driven approach" — similarity between context of known links to γ and context words in spot s
- "Context" consists of ± 3 words around mention, their parts of speech, salient words chosen from whole document

Results

- "Data-driven" better than "knowledge-based"
- Consensus (agreement) has highest precision

Method	Words		Evaluation		
	(A)	(C)	(P)	(R)	(F)
	Baselines				
Random baseline	6,517	4,161	63.84	56.90	60.17
Most frequent sense	6,517	5,672	87.03	77.57	82.02
	Word sense disambiguation methods				
Knowledge-based	6,517	5,255	80.63	71.86	75.99
Feature-based learning	6,517	6,055	92.91	**83.10**	**87.73**
Combined	5,433	5,125	**94.33**	70.51	80.69

Relatedness info from entity catalog

- How related are two entities γ, γ' in Wikipedia?
- Embed γ in some space using $g : \Gamma \to \mathbb{R}^c$
- Define relatedness $r(\gamma, \gamma') = g(\gamma) \cdot g(\gamma')$ or related
- Cucerzan's proposal: c = number of categories; $g(\gamma)[\tau] = 1$ if γ belongs to category τ, 0 otherwise

$$r(\gamma, \gamma') = \frac{g(\gamma)^\top g(\gamma')}{\sqrt{g(\gamma)^\top g(\gamma)}\sqrt{g(\gamma')^\top g(\gamma')}},$$

(standard cosine)

Relatedness info from entity catalog (2)

- Milne and Witten's proposal: c = number of Wikipedia pages; $g(\gamma)[p] = 1$ if page p links to page γ, 0 otherwise

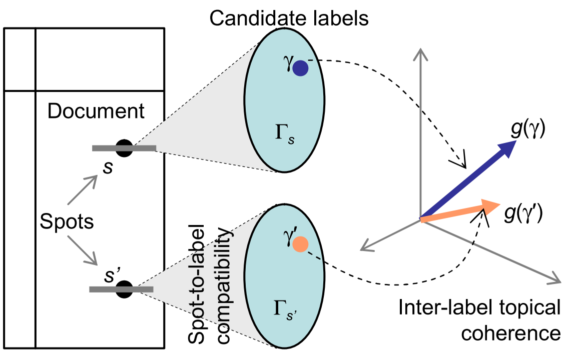
$$r(\gamma, \gamma') = \frac{\log \dfrac{|g(\gamma) \cap g(\gamma')|}{|g(\gamma) \cup g(\gamma')|}}{\log \dfrac{c}{\min\{|g(\gamma)|, |g(\gamma')|\}}}$$

- Related to Jaccard
- With voice of small inlink sets attenuated

Leave-one-out disambiguation

- Let Γ_0 be all possible entity disambiguations for all spots on a page
- Precompute the average vector $g(\Gamma_0) = \sum_{\gamma \in \Gamma_0} g(\gamma)$
- Score of candidate label γ for spot s depends on two factors multiplied together
- The local compatibility score as before
- $g(\gamma)^\top g(\Gamma_0 \setminus \{\gamma\}) = g(\gamma)^\top \sum_{\gamma' \in \Gamma_0 \setminus \gamma} g(\gamma')$
- Note that $\Gamma_0 \setminus \gamma$ still contains contributions from entities that cannot be used simultaneously to label the page
- $g(\Gamma_0 \setminus \gamma)$ may not be a representative feature vector

The need for collective disambiguation



- Premise: coherent doc refers to entities about related categories
- Optimize wrt y an objective with two parts:
  - Local compatibility between s and y_s
  - Global coherence between y_s and $y_{s'}$ for all spot pairs

Two-part objective to maximize

Node potential:

$$\mathrm{NP}(y) = \prod_s \mathrm{NP}_s(y_s) = \prod_s \exp\left(w^\top f_s(y_s)\right)$$

Clique potential:

$$\mathrm{CP}(y) = \exp\left(\sum_{s \neq s'} g(y_s)^\top g(y_{s'})\right)$$

After taking logs and rescaling terms

$$\frac{1}{|S_0|} \sum_s w^\top f_s(y_s) + \frac{1}{\binom{|S_0|}{2}} \sum_{s \neq s'} g(y_s)^\top g(y_{s'})$$

Two-part objective to maximize

Node potential:

$$\mathrm{NP}(y) = \prod_s \mathrm{NP}_s(y_s) = \prod_s \exp\left(w^\top f_s(y_s)\right)$$

Clique potential:

$$\mathrm{CP}(y) = \exp\left(\sum_{s \neq s'} g(y_s)^\top g(y_{s'})\right)$$

After taking logs and rescaling terms

$$\frac{1}{|S_0|} \sum_s w^\top f_s(y_s) + \frac{1}{\binom{|S_0|}{2}} \sum_{s \neq s'} g(y_s)^\top g(y_{s'})$$

High-level TOC

Annotation:

- Identifying token segments as entity mentions
- Ditto for relations
- Annotation vs. extraction
- Closed vs. open domain

Disambiguation:

- Supervised: against entities in a catalog
- Unsupervised: cluster mentions referring to same entity

Search:

- Relational
- XML, trees, twigs
- Proximity in text and general graphs

Two extreme search paradigms

Searching a RDBMS

- Complex data model: tables, rows, columns, data types
- Expressive, powerful query language
- Need to know schema to query
- Answer = unordered set of rows
- Ranking: afterthought

Information Retrieval

- Collection = set of documents, document = sequence of terms
- Terms and phrases present or absent
- No (nontrivial) schema to learn
- Answer = sequence of documents
- Ranking: central to IR

Convergence?

SQL→XML search

- Trees, reference links
- Labeled edges
- Nodes may contain
  - Structured data
  - Free text fields

  Data vs. document
- Query involves node data and edge labels
  - Partial knowledge of schema ok
- Answer = set of paths

Web search←IR

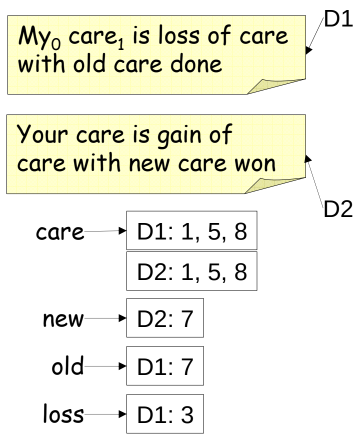
- Documents are nodes in a graph
- Hyperlink edges have important but unspecified semantics
  - Google, HITS
- Query language remains primitive
  - No data types
  - No use of tag-tree
- Answer = URL list

Outline of this tutorial

- Review of text indexing and information retrieval (IR)
- Support for text search and similarity join in relational databases with text columns
- Text search features in major XML query languages (and what's missing)
- A graph model for semi-structured data with "free-form" text in nodes
- Proximity search formulations and techniques; how to rank responses
- Folding in user feedback
- Trends and research problems

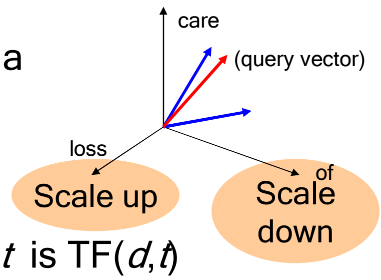
Text indexing basics

- "Inverted index" maps from term to document IDs
- Term offset info enables phrase and proximity ("near") searches
- Document boundary and limitations of "near" queries
- Can extend inverted index to map terms to
  - Table names, column names
  - Primary keys, RIDs
  - XML DOM node IDs



Information retrieval basics

- Stopwords and stemming
- Each term t in lexicon gets a dimension in vector space
- Documents and the query are vectors in term space



- Component of d along axis t is TF(d,t)
  - Absolute term count or scaled by max term count
- Downplay frequent terms: IDF(t) = $\log(1+|D|/|D_t|)$
  - Better model: document vector d has component TF(d,t) IDF(t) for term t
- Query is like another "document"; documents ranked by cosine similarity with query

Map

		Data model	
		Relational	XML-like
IR support	None	SQL, Datalog	XML-QL, Xquery
	Schema	WHIRL	ELIXIR, XIRQL
	No schema	DBXplorer, BANKS, DISCOVER	EasyAsk, Mercado, DataSpot, BANKS

- "None" = nothing more than string equality, containment (substring), and perhaps lexicographic ordering
- "Schema": Extensions to query languages, user needs to know data schema, IR-like ranking schemes, no implicit joins
- "No schema": Keyword queries, implicit joins

WHIRL (Cohen 1998)

place(univ,state) and job(univ,dept)

- Ranked retrieval from a RDBMS:
  - select univ from job where dept ~ 'Civil'
- Ranked similarity join on text columns:
  - select state, dept from place, job where place.univ ~ job.univ
- Limit answer to best *k* matches only
- Avoid evaluating full Cartesian product
  - "Iceberg" query
- Useful for data cleaning and integration

WHIRL scoring function

A where-clause in WHIRL is a

- Boolean predicate as in SQL (age=35)
  - Score for such clauses are 0/1
- Similarity predicate (job ~ 'Web design')
  - Score = cosine(job, 'Web design')
- Conjunction or disjunction of clauses
  - Sub-clause scores interpreted as probabilities
  - $\text{score}(B_1 \wedge \ldots \wedge B_m;\ \theta) = \Pi_{1 \le i \le m} \text{score}(B_i, \theta)$
  - $\text{score}(B_1 \vee \ldots \vee B_m;\ \theta) = 1 - \Pi_{1 \le i \le m} (1 - \text{score}(B_i, \theta))$

Query execution strategy

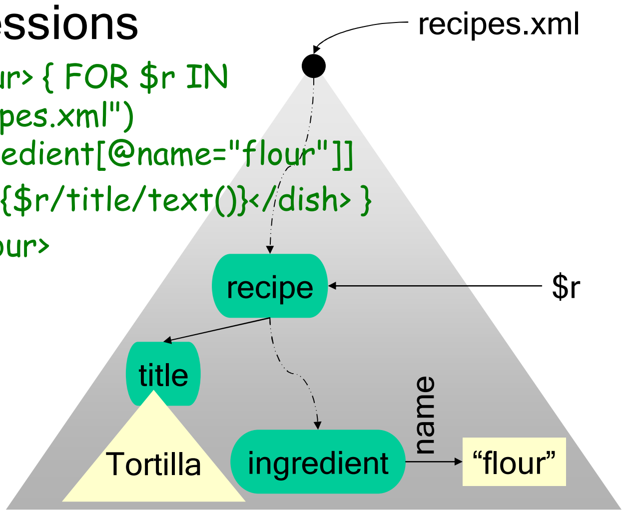
```
select state, dept from place, job
        where place.univ ~ job.univ
```

- Start with place(U1,S) and job(U2,D) where U1, U2, S and D are “free”
  - Any binding of these variables to constants is associated with a score
- Greedily extend the current bindings for maximum gain in score
- Backtrack to find more solutions

XQuery

- Quilt + Lorel + YATL + XML-QL
- Path expressions

```
<dishes_with_flour> { FOR $r IN
   document("recipes.xml")
   //recipe[//ingredient[@name="flour"]]
   RETURN <dish>{$r/title/text()}</dish> }
</dishes_with_flour>
```



Early text support in XQuery

- Title of books containing some para mentioning both “sailing” and “windsurfing”

```
FOR $b IN document("bib.xml")//book
WHERE SOME $p IN $b//paragraph SATISFIES
   (contains($p,"sailing") AND
     contains($p,"windsurfing"))
RETURN $b/title
```

- Title and text of documents containing at least three occurrences of “stocks”

```
FOR  $a IN view("text_table") WHERE
   numMatches($a/text_document,"stocks") > 3
RETURN
   <text>{$a/text_title}{$a/text_document}</>
```


Tutorial outline

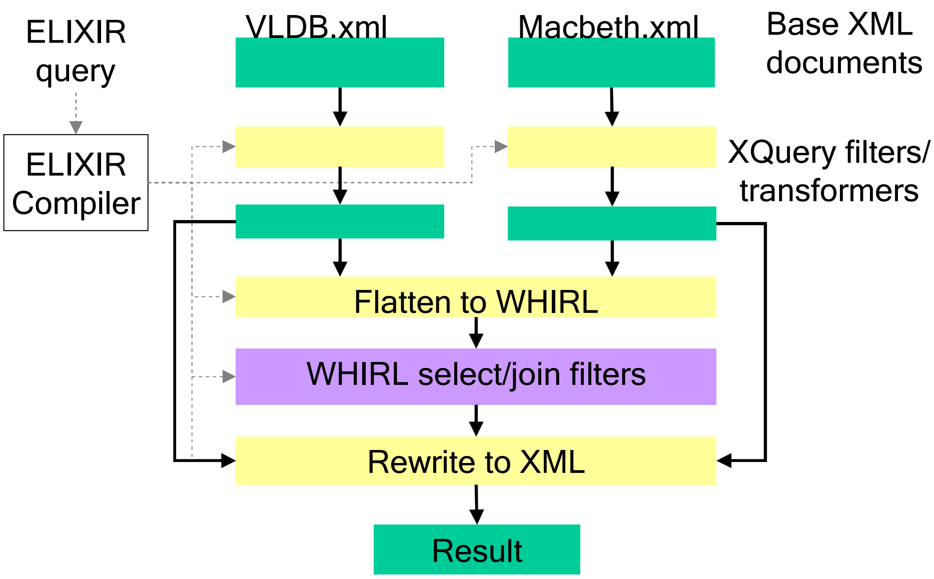
		Data model	
		Relational	XML-like
IR support	None	SQL,Datalog	~~XML-QL~~, Xquery
	Schema	WHIRL	ELIXIR, XIRQL
	No schema	DBXplorer, BANKS, DISCOVER	EasyAsk, Mercado, DataSpot, BANKS

- Review of text indexing and information retrieval
- Support for text search and similarity join in relational databases with text columns (WHIRL)
- Adding IR-like text search features to XML query languages (Chinenyanga et al. Führ et al. 2001)

ELIXIR: Adding IR to XQuery

- Ranked select
```
for $t in document("db.xml")/items/(book|cd)
where $t/text() ~ "Ukrainian recipe"
return <dish>$t</dish>
```
- Ranked similarity join: find titles in recent VLDB proceedings similar to speeches in Macbeth
```
for $vi in
        document("vldb.xml")/issue[@volume>24],
      $si in document("macbeth.xml")//speech
where $vi//article/title ~ $si
return <similar><title>$vi//article/title</>
                  <speech>$si</></similar>
```


How ELIXIR works



A more detailed view

VLDB.xml

```
<issue><volume>10</>
 <article>…</> …</>
<issue><volume>25</>
 <article><title>Size
 separation spatial join</>…</></>
```

```
<q21>{ for $at in
document("VLDB.xml")//issue
   [volume > 24]//title
return
<tuple><title>{ $at }</></tuple> }</q21>
```

q21.xml

```
<q21><tuple><title>Size separation
spatial join</title></tuple></q21>
```

Macbeth.xml

```
<act number="…">
<scene number="…">
 <speech>To Ireland, I; our
 separated fortune.</>
</></>
```

```
<q22>{ for $as in
document("Macbeth.xml")//act/scene/speech
return <tuple><line>{ $as }</></tuple> }</q22>
```

q22.xml

```
<q22><tuple><line>To Ireland,
I; our separated fortune.
</line></tuple></q22>
```

```
q3($title,$line) :- q21($title), q22($line), $title ~ $line
```

WHIRL query

```
<similar>{ for $row in q3/tuple return $row }</>
```

Result

Observations

- SQL/XQuery + IR-like result ranking
- Schema knowledge remains essential
  - "Free-form" text vs. tagged, typed field
  - Element hierarchy, element names, IDREFs
- Typical Web search is two words long
  - End-users don't type SQL or XQuery
  - Possible remedy: HTML form access
  - Limitation: restricted views and queries

Using proximity without schema

- General, detailed representation: XML
- Lowest common representation
  - Collection, document, terms
  - Document = node, hyperlink = edge
- Middle ground
  - Graph with text (or structured data) in nodes
  - Links: element, subpart, IDREF, foreign keys
  - All links hint at unspecified notion of proximity

Exploit structure where available, but do not impose structure by fiat

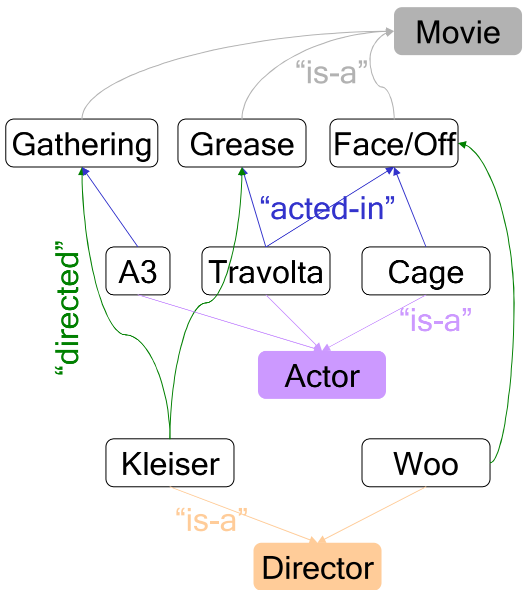
Two paradigms of proximity search

- A single node as query response
  - Find node that matches query terms…
  - …or is "near" nodes matching query terms

  (Goldman et al., 1998)
- A connected subgraph as query response
  - Single node may not match all keywords
  - No natural "page boundary"

Single-node response examples

- Travolta, Cage
  - Actor, Face/Off
- Travolta, Cage, Movie
  - Face/Off
- Kleiser, Movie
  - Gathering, Grease
- Kleiser, Woo, Actor
  - Travolta



Basic search strategy

- Node subset A activated because they match query keyword(s)
- Look for node near nodes that are activated
- Goodness of response node depends
  - Directly on degree of activation
  - Inversely on distance from activated node(s)

Ranking a single node response

- Activated node set *A*
- Rank node *r* in "response set" *R* based on proximity to nodes *a* in *A*
  - Nodes have relevance ρ_R and ρ_A in [0,1]
  - Edge costs are "specified by the system"
- $d(a,r)$ = cost of shortest path from *a* to *r*
- Bond between *a* and *r*

$$b(a,r) = \frac{\rho_A(a)\rho_R(r)}{d(a,r)^t}$$

- Parameter *t* tunes relative emphasis on distance and relevance score
- Several ad-hoc choices

Scoring single response nodes

- Additive

$$\text{score}(r) = \sum_{a \in A} b(a,r)$$

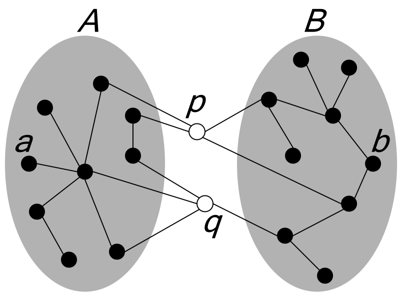
- Belief

$$\text{score}(r) = 1 - \prod_{a \in A} \left(1 - b(a,r)\right)$$

- Goal: list a limited number of find nodes with the largest scores
- Performance issues
  - Assume the graph is in memory?
  - Precompute all-pairs shortest path ($|V|^3$)?
  - Prune unpromising candidates?

Hub indexing

- Decompose APSP problem using sparse vertex cuts
  - |*A*|+|*B* | shortest paths to *p*
  - |*A*|+|*B* | shortest paths to *q*
  - *d*(*p*,*q*)
- To find *d*(*a*,*b*) compare
  - *d*(*a*→*p*→*b*) not through *q*
  - *d*(*a*→*q*→*b*) not through *p*
  - *d*(*a*→*p*→*q*→*b*)
  - *d*(*a*→*q*→*p*→*b*)
- Greatest savings when |A|≈|B|
- Heuristics to find cuts, e.g. large-degree nodes



Connected subgraph as response

- Single node may not match all keywords
- No natural "page boundary"
- Two scenarios
  - Keyword search on relational data
    - Keywords spread among normalized relations
  - Keyword search on XML-like or Web data
    - Keywords spread among DOM nodes and subtrees

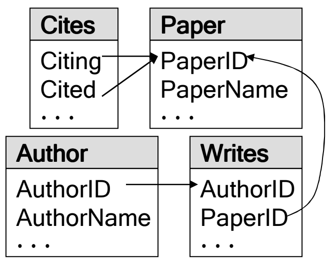
Tutorial outline

		Data model	
		Relational	XML-like
IR support	None	SQL,Datalog	XML-QL, Xquery
	Schema	~~WHIRL~~	ELIXIR, XIRQL
	No schema	DBXplorer, BANKS, DISCOVER	EasyAsk, Mercado, DataSpot, BANKS

- Adding IR-like text search features to XML query languages
- A graph model for relational data with “free-form” text search and implicit joins
- Generalizing to graph models for XML

Keyword search on relational data

- Tuple = node
- Some columns have text
- Foreign key constraints = edges in schema graph→
- Query = set of terms
- No natural notion of a document
  - Normalization
  - Join may be needed to generate results
  - Cycles may exist in schema graph: ‘Cites’



AuthorID	PaperID
A1	P1
A2	P2
A3	P2

AuthorID	AuthorName
A1	Chaudhuri
A2	Sudarshan
A3	Hulgeri

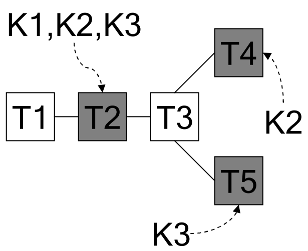
Citing	Cited
P2	P1

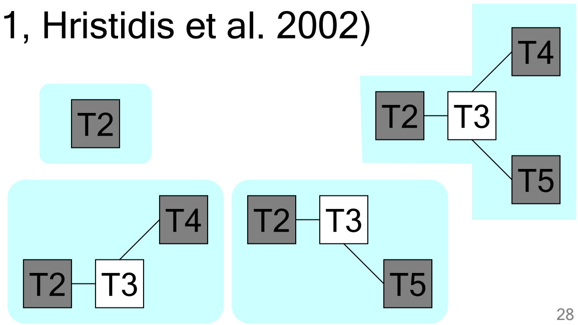
PaperID	PaperName
P1	DBXplorer
P2	BANKS

DBXplorer and DISCOVER

- Enumerate subsets of relations in schema graph which, when joined, may contain rows which have *all* keywords in the query
  - "Join trees" derived from schema graph
- Output SQL query for each join tree
- Generate joins, checking rows for matches

(Agrawal et al. 2001, Hristidis et al. 2002)





Discussion

- Exploits relational schema information to contain search
- Pushes final extraction of joined tuples into RDBMS
- Faster than dealing with full data graph directly

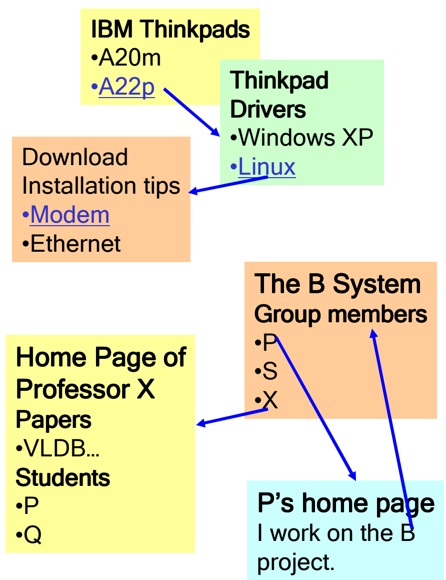
- Coarse-grained ranking based on schema tree
- Does not model proximity or (dis) similarity of individual tuples
- No recipe for data with less regular (e.g. XML) or ill-defined schema

Generalized graph proximity

- General data graph
  - Nodes have text, can be scored against query
  - Edge weights express dissimilarity
- Query is a set of keywords as before
- Response is a connected subgraph of the database
- Each response graph is scored using
  - Node weights which reflect match, maximize
  - Edge weights which reflect lack of proximity, minimize

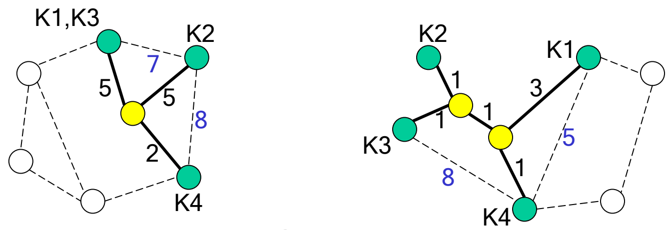
Motivation from Web search

- "Linux modem driver for a Thinkpad A22p"
  - Hyperlink path matches query collectively
  - Conjunction query would fail
- Projects where X and P work together
  - Conjunction may retrieve wrong page
- General notion of graph proximity

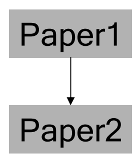


“Information unit” (Lee et al., 2001)

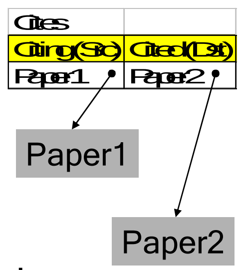
- Generalizes join trees to arbitrary graph data
- Connected subgraph of data without cycles
- Includes at least one node containing each query keyword
- Edge weights represent price to pay to connect all keyword-matching nodes together
- May have to include non-matching nodes



Setting edge weights

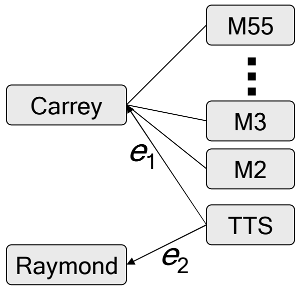


- Edges are generally directed
  - Foreign to primary key in relational data
  - Containing to contained element in XML
  - IDREFs have clear source and target
- Consider the RDMS scenario
- Forward edge weight for edge (u,v)
  - u, v are tuples in tables $R(u)$, $R(v)$
  - Weight $s(R(u),R(v))$ between tables
    - Configured heuristically based on semantics
    - $w_F(u,v)=s(R(u),R(v))$ all such tuple pairs u, v
- Proximity search must traverse edges in *both* directions … what should $w_B(u,v)$ be?



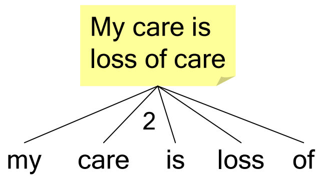
Backward edge weights

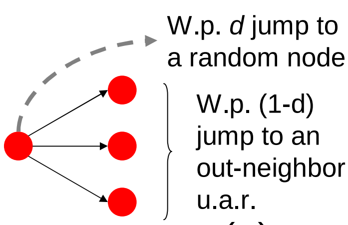
- "Distance" between a pair of nodes is asymmetric in general
  - Ted Raymond acted only in The Truman Show, which is 1 of 55 movies for Jim Carrey
  - $w(e_1)$ should be larger than $w(e_2)$ (think "resistance" on the edge)
- For every edge (u,v) that exists, $w_B(u,v)=s(R(v),R(u))\ .\ \mathrm{IN}_v(u)$
  - $\mathrm{IN}_v(u)$ is the #edges from $R(v)$ to u
- $w(u,v) = \min\{w_F(u,v),\ w_B(u,v)\}$
- More general edge weight models possible, e.g., R→S→T relation path-based weights



Node weight = relevance + prestige

- Relevance w.r.t. keyword(s)
  - 0/1: node contains term or it does not
  - Cosine score in [0,1] as in IR
  - Uniform model: a node for each keyword (e.g. DataSpot)
- Popularity or prestige
  - E.g. "mohan transaction"
  - Indegree
  - PageRank





$$p(v) = \frac{d}{N} + (1-d)\sum_{u \to v} \frac{p(u)}{\mathrm{OutDegree}(u)}$$

Trading off node and edge weights

- A high-scoring answer *A* should have
  - Large node weight
  - Small edge weight
- Weights must be normalized to extreme values
- N(*v*)=node weight of *v*
- Overall NodeScore = $\dfrac{\sum_{v \in A} \log\left(1+ {N(v)}/{N_{\max}}\right)}{\#\text{nodes}}$
- Overall EdgeScore = $\dfrac{1}{1+\sum_{e \in A} \log\left(1+ {w(e)}/{w_{\min}}\right)}$
- Overall score = EdgeScore $\times$ NodeScore$^{\lambda}$
  - λ tunes relative contribution of nodes and edges
- Ad-hoc, but guided by heuristic choices in IR

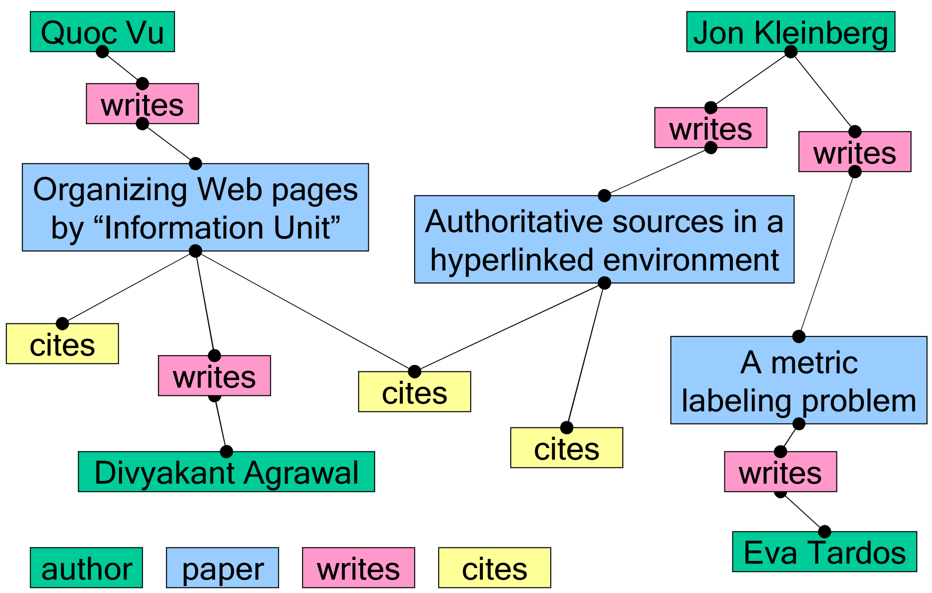
Data structures for search

- Answer = tree with at least one leaf containing each keyword in query
  - Group Steiner tree problem, NP-hard
- Query term *t* found in source nodes S_t
- Single-source-shortest-path SSSP iterator
  - Initialize with a source (near-) node
  - Consider edges backwards
  - getNext() returns next nearest node
- For each iterator, each visited node *v* maintains for each *t* a set $v.R_t$ of nodes in S_t which have reached *v*

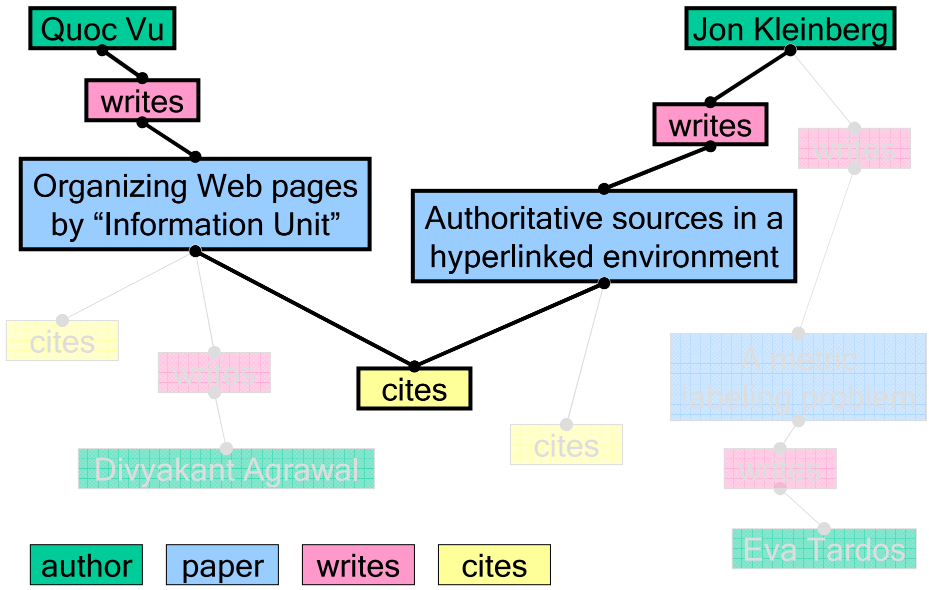
Generic expanding search

- Near node sets S_t with $S = \cup_t S_t$
- For all source nodes $\sigma \in S$
  - create a SSSP iterator with source σ
- While more results required
  - Get next iterator and its next-nearest node v
  - Let t be the term for the iterator's source s
  - crossProduct = $\{s\} \times \Pi_{t' \neq t} v.R_{t'}$
  - For each tuple of nodes in crossProduct
    - Create an answer tree rooted at v with paths to each source node in the tuple
  - Add s to $v.R_t$

Search example ("Vu Kleinberg")



First response

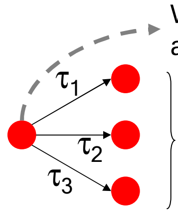


Folding in user feedback

- As in IR systems, results may be imperfect
  - Unlike SQL or XQuery, no exact control over matching, ranking and answer graph form
  - Ad-hoc choices for node and edge weights
- Per-user and/or per-session
  - By graph/path/node type, e.g. “want author *citing* author,” not “author *coauthoring with* author”
- Across users
  - Modifying edge costs to favor nodes (or node types) liked by users

Random walk formulations

- Generalize PageRank to treat outlinks differently
  - τ(u,v) is the "conductance" of edge *u*→*v*
- *p*(*v*) is a function of τ(*u*,*v*) for all in-neighbors *u* of *v*
  - p_{guess}(v) … at convergence
  - p_{user}(v) … user feedback



W.p. *d* jump to a random node

W.p. 1-*d* = $\tau_1+\tau_2+\tau_3$ jump to an out-neighbor

$$p(v) = \frac{d}{N} + \sum_{u \to v} p(u)\ \tau(u,v)$$

$$\frac{\partial p(v)}{\partial \tau(u,v)} = p(u)$$

Gradient ascent/descent:

- For each *u*→*v*, set (with learning rate η):
  $$\tau(u,v) \leftarrow \tau(u,v) + \eta\ \mathrm{sgn}\big(p_{user}(v) - p_{guess}(v)\big)\frac{p(u)}{\sum_{u' \to v} p(u')}$$
- Re-iterate to convergence

Prototypes and products

- DTL DataSpot → Mercado Intuifind www.mercado.com/
- EasyAsk www.easyask.com/
- ELIXIR www.smi.ucd.ie/elixir/
- XIRQL ls6-www.informatik.uni-dortmund.de/ir/projects/hyrex/
- Microsoft DBXplorer
- BANKS www.cse.iitb.ac.in/banks/

Summary

- Confluence of structured and free-format, keyword-based search
  - Extend SQL, XQuery, Web search, IR
  - Many useful applications: product catalogs, software libraries, Web search
- Key idiom: proximity in a graph representation of textual data
  - Implicit joins on foreign keys
  - Proximity via IDREF and other links
- Several working systems
- Not enough consensus on clean models

References

[1] M. Hearst, "Automatic acquisition of hyponyms from large text corpora," in *International Conference on Computational Linguistics*, vol. 14, 1992, pp. 539–545.

[2] S. Brin, "Extracting patterns and relations from the World Wide Web," in *WebDB Workshop*, ser. LNCS, P. Atzeni, A. O. Mendelzon, and G. Mecca, Eds., vol. 1590. Valencia, Spain: Springer, Mar. 1998, pp. 172–183.

References (2)

[3] E. Agichtein and L. Gravano, "Snowball: Extracting relations from large plain-text collections," in *ICDL*. ACM, 2000, pp. 85–94.

[4] P. D. Turney, "Mining the Web for synonyms: PMI-IR versus LSA on TOEFL," in *ECML*, 2001.

[5] S. Dill *et al.*, "SemTag and Seeker: Bootstrapping the semantic Web via automated semantic annotation," in *WWW Conference*, 2003.

References (3)

[6] J. Lafferty, A. McCallum, and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *ICML*, 2001, pp. 282–289.

[7] J. Zhu, Z. Nie, B. Zhang, and J.-R. Wen, "Dynamic hierarchical Markov random fields and their application to Web data extraction," in *ICML*, 2007, pp. 1175–1182. http://www.machinelearning.org/proceedings/icml2007/papers/215.pdf

References (4)

[8] O. Etzioni, M. Cafarella, *et al.*, "Web-scale information extraction in KnowItAll," in *WWW Conference*. New York: ACM, 2004. http://www.cs.washington.edu/research/knowitall/papers/www-paper.pdf

[9] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, "Open information extraction from the Web," in *IJCAI*, M. M. Veloso, Ed., 2007, pp. 2670–2676. http://www.ijcai.org/papers07/Papers/IJCAI07-429.pdf

References (5)

[10] M. J. Cafarella, C. Re, D. Suciu, O. Etzioni, and M. Banko, "Structured querying of web text: A technical challenge," in *CIDR*, 2007, pp. 225–234. http://www-db.cs.wisc.edu/cidr/cidr2007/papers/cidr07p25.pdf

[11] D. Freitag and A. McCallum, "Information extraction using HMMs and shrinkage," in *Papers from the AAAI-99 Workshop on Machine Learning for Information Extraction*, 1999, pp. 31–36.

References (6)

[12] I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun, "Large margin methods for structured and interdependent output variables," *JMLR*, vol. 6, no. Sep, pp. 1453–1484, 2005. http://ttic.uchicago.edu/~altun/pubs/TsoJoaHofAlt-JMLR.pdf

[13] F. Sha and F. Pereira, "Shallow parsing with conditional random fields," in *HLT-NAACL*, 2003, pp. 134–141. http://acl.ldc.upenn.edu/N/N03/N03-1028.pdf

References (7)

[14] R. C. Bunescu and R. J. Mooney, "A shortest path dependency kernel for relation extraction," in *EMNLP Conference*. Association for Computational Linguistics, 2005, pp. 724–731. http://acl.ldc.upenn.edu/H/H05/H05-1091.pdf

[15] R. Mihalcea and A. Csomai, "Wikify!: linking documents to encyclopedic knowledge," in *CIKM*, 2007, pp. 233–242. http://portal.acm.org/citation.cfm?id=1321440.1321475

References (8)

[16] R. Bunescu and M. Pasca, "Using encyclopedic knowledge for named entity disambiguation," in *EACL*, 2006, pp. 9–16. http://www.cs.utexas.edu/~ml/papers/encyc-eacl-06.pdf

[17] S. Cucerzan, "Large-scale named entity disambiguation based on Wikipedia data," in *EMNLP Conference*, 2007, pp. 708–716. http://www.aclweb.org/anthology/D/D07/D07-1074

References (9)

[18] S. Kulkarni, A. Singh, G. Ramakrishnan, and S. Chakrabarti, "Collective annotation of Wikipedia entities in Web text," in *SIGKDD Conference*, 2009.

[19] G. Bhalotia, A. Hulgeri, C. Nakhe, S. Chakrabarti, and S. Sudarshan, "Keyword searching and browsing in databases using BANKS," in *ICDE*. IEEE, 2002.

References (10)

[20] S. Chakrabarti, "Dynamic personalized PageRank in entity-relation graphs," in *WWW Conference*, Banff, May 2007. http://www.cse.iitb.ac.in/~soumen/doc/netrank/

[21] H. Garcia-Molina, R. Goldman, N. Shivakumar, and S. Venkatasubramanian, "Proximity search in databases," in *24th International Conference on Very Large Databases (VLDB)*, 1998.

References (11)

[22] D. Florescu, D. Kossman, and I. Manolescu, "Integrating keyword searches into XML query processing," in *WWW Conference*, Amsterdam, 2000, pp. 119–135. http://www9.org/w9cdrom/324/324.html

[23] T. T. Chinenyanga and N. Kushmerick, "An expressive and efficient language for XML information retrieval," *JASIST*, vol. 53, no. 6, pp. 438–453, 2002. citeseer.ist.psu.edu/andn01expressive.html

References (12)

[24] N. Fuhr and K. Grosjohann, "XIRQL: A query language for information retrieval in XML documents," in *Research and Development in Information Retrieval*, 2001, pp. 172–180. citeseer.ist.psu.edu/article/fuhr01xirql.html

[25] S. Agrawal, S. Chaudhuri, and G. Das, "DBXplorer: A system for keyword-based search over relational databases," in *ICDE*. San Jose, CA: IEEE, 2002.

References (13)

[26] A. Balmin, V. Hristidis, and Y. Papakonstantinou, "Authority-based keyword queries in databases using ObjectRank," in *VLDB Conference*, Toronto, 2004.

[27] P. Sarkar, A. W. Moore, and A. Prakash, "Fast incremental proximity search in large graphs," in *ICML*, 2008, pp. 896–903. http://icml2008.cs.helsinki.fi/papers/565.pdf