Scalable Information Extraction and Integration

Eugene Agichtein — Microsoft Research → Emory University

Sunita Sarawagi — IIT Bombay

The Value of Text Data

- "Unstructured" text data is the primary source of human-generated information
  - Citeseer, comparison shopping, PIM systems, web search, data warehousing
- Managing and utilizing text: information extraction and integration
- Scalability: a bottleneck for deployment
- Relevance to data mining community

Example: Answering Queries Over Text

For years, Microsoft Corporation CEO Bill Gates was against open source. But today he appears to have changed his mind. "We can be open source. We love the concept of shared source," said Bill Veghte, a Microsoft VP. "That's a super-important shift for us in terms of code access.“

Richard Stallman, founder of the Free Software Foundation, countered saying…

```
Select  Name
From    PEOPLE
Where   Organization = 'Microsoft'
```

PEOPLE

Name	Title	Organization
Bill Gates	CEO	Microsoft
Bill Veghte	VP	Microsoft
Richard Stallman	Founder	Free Soft..

Bill Gates

Bill Veghte

(from William Cohen's IE tutorial, 2003)

Managing Unstructured Text Data

- **Information Extraction from text**
  - **Represent information in text data in a structured form**
  - Identify instances of entities and relationships
  - Main approaches and architectures
  - Scaling up to large collections of documents (e.g., web)

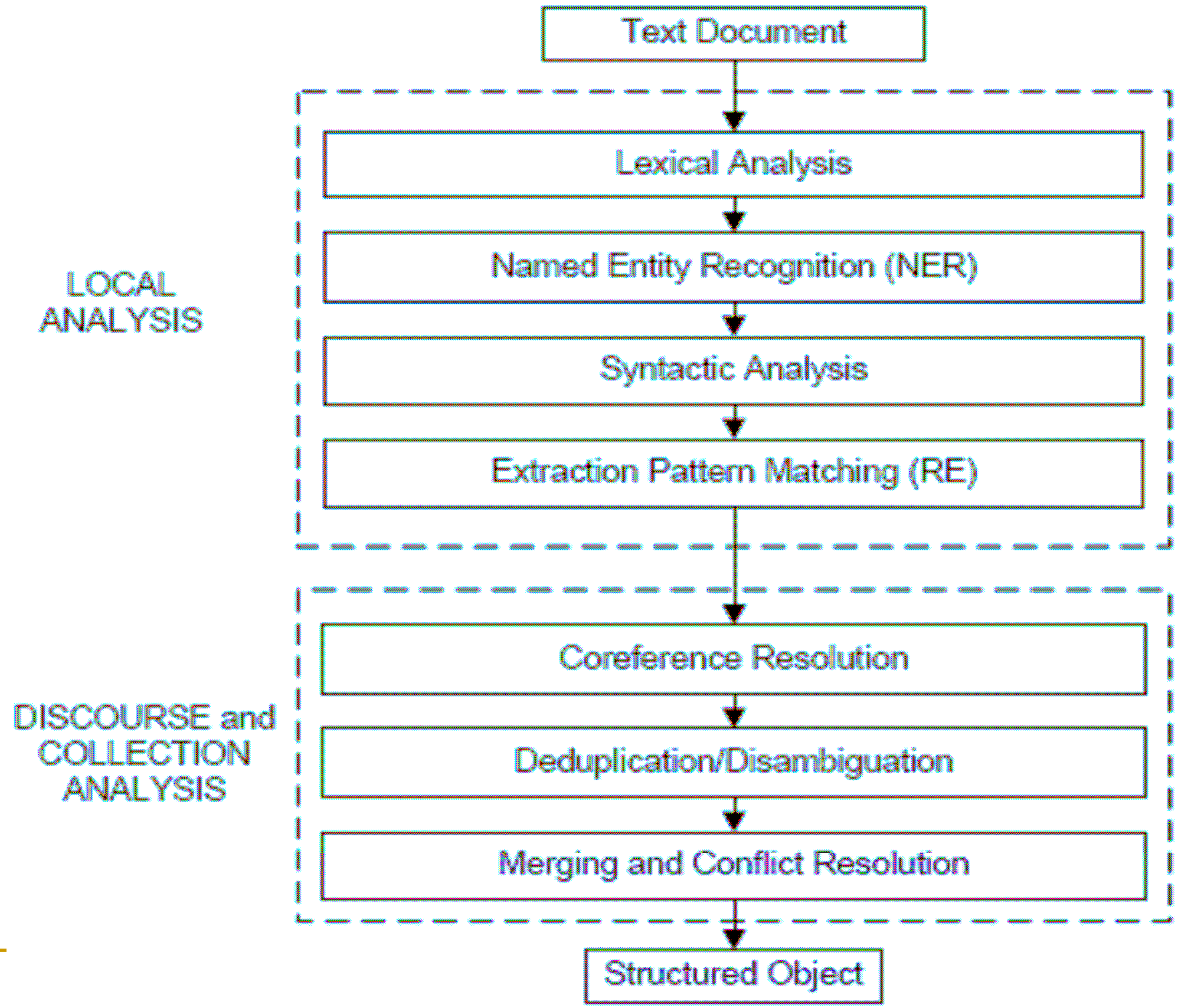
- **Information Integration**
  - **Combine/resolve/clean information about entities**
  - Entity Resolution & Deduplication
  - Scaling Up: Batch mode/algorithmic issues

- Connections between Information Extraction and Integration
  - Coreference Resolution
  - Deriving values from multiple sources
  - (Web) Question Answering

Part I: Tutorial Outline

- **Overview of Information Extraction**
  - Entity tagging
  - Relation extraction
- Scaling up Information Extraction
  - Focus on scaling up to large collections (where data mining and ML techniques shine)
  - Other dimensions of scalability

Information Extraction Components



Information Extraction Tasks

- Extracting entities and relations: this tutorial
  - Entities: named (e.g., Person) and generic (e.g., disease name)
  - Relations: entities related in a predefined way (e.g., Location of a Disease outbreak)
- Common extraction subtasks:
  - Preprocessing: sentence chunking, syntactic parsing, morphological analysis
  - Creating rules or extraction patterns: manual, machine learning, and hybrid
  - Applying extraction patterns to extract new information
- Postprocessing and complex extraction: not covered
  - Co-reference resolution
  - Combining Relations into Events and Facts

Related Tutorials

- Previous information extraction tutorials: consult for more details
  - R. Feldman, Information Extraction – Theory and Practice, ICML 2006
    http://www.cs.biu.ac.il/~feldman/icml_tutorial.html
  - W. Cohen, A. McCallum, Information Extraction and Integration: an Overview, KDD 2003
    http://www.cs.cmu.edu/~wcohen/ie-survey.ppt
  - A. Doan, R. Ramakrishnan, S. Vaithyanathan, Managing Information Extraction, SIGMOD'06
  - N. Koudas, D. Srivastava, S. Sarawagi, Record Linkage: Similarity Measures and Algorithms, SIGMOD 2006

Entity Tagging

- Identifying mentions of entities (e.g., person names, locations, companies) in text
  - MUC (1997): Person, Location, Organization, Date/Time/Currency
  - ACE (2005): more than 100 more specific types

- Hand-coded vs. Machine Learning approaches

- Best approach depends on entity type and domain:
  - Closed class (e.g., geographical locations, disease names, gene & protein names): hand coded + dictionaries
  - Syntactic (e.g., phone numbers, zipcodes): regexes
  - Others (e.g., person and company names): mixture of context, syntactic features, dictionaries, heuristics, etc.
  - “Almost solved” for common/typical entity types

- Non-syntactic entities computationally expensive

Example: Extracting Entities from Text

- Useful for data warehousing, data cleaning, web data integration

Address

House number	Building	Road	City	State	Zip
4089	Whispering Pines	Nobel Drive	San Diego	CA	92122

Citation

Ronald Fagin, Combining Fuzzy Information from Multiple Systems, Proc. of ACM SIGMOD, 2002

Segment(s_i)	Sequence	Label(s_i)
S_1	Ronald Fagin	Author
S_2	Combining Fuzzy Information from Multiple Systems	Title
S_3	Proc. of ACM SIGMOD	Conference
S_4	2002	Year

Hand-Coded Methods

- Easy to construct in many cases
  - e.g., to recognize prices, phone numbers, zip codes, conference names, etc.
- Easier to debug & maintain
  - Especially if written in a “high-level” language (as is usually the case): e.g., *[From Avatar]*

```
ContactPattern  ←  RegularExpression(Email.body,”can be reached at”)

PersonPhone     ←  Precedes(Person
                        Precedes(ContactPattern, Phone, D),
                        D)
```

- Easier to incorporate / reuse domain knowledge
- Can be quite labor intensive to write

Example of Hand-Coded Entity Tagger

[Ramakrishnan. G, 2005, Slides from Doan et al., SIGMOD 2006]

Rule 1 This rule will find person names with a salutation (e.g. Dr. Laura Haas) and two capitalized words

```
<token> INITIAL</token>
<token>DOT </token>
<token>CAPSWORD</token>
<token>CAPSWORD</token>
```

Rule 2 This rule will find person names where two capitalized words are present in a Person dictionary

```
<token>PERSONDICT, CAPSWORD </token>
<token>PERSONDICT, CAPSWORD</token>
```

CAPSWORD : Word starting with uppercase, second letter lowercase
E.g., DeWitt will satisfy it (DEWITT will not)
\p{Upper}\p{Lower}[\p{Alpha}]{1,25}

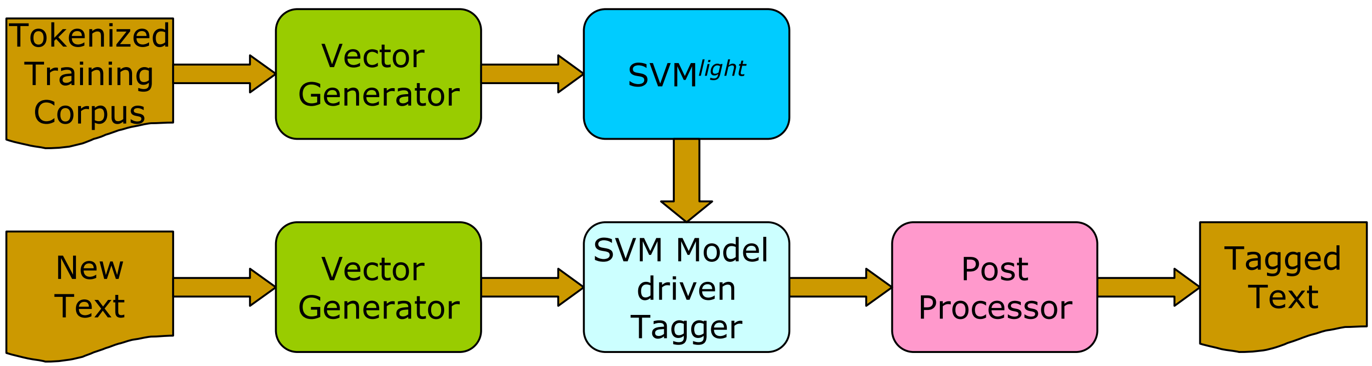
DOT : The character '.'

Hand Coded Rule Example: Conference Name

```
# These are subordinate patterns
$wordOrdinals="(?:first|second|third|fourth|fifth|sixth|seventh|eighth|ninth|tenth|eleventh|twelfth|thirtee
teenth|fifteenth)";
my $numberOrdinals="(?:\\d?(?:1st|2nd|3rd|1th|2th|3th|4th|5th|6th|7th|8th|9th|0th))";
my $ordinals="(?:$wordOrdinals|$numberOrdinals)";
my $confTypes="(?:Conference|Workshop|Symposium)";
my $words="(?:[A-Z]\\w+\\s*)"; # A word starting with a capital letter and ending with 0 or more spaces
my $confDescriptors="(?:international\\s+|[A-Z]+\\s+)"; # .e.g "International Conference ...' or the confere
name for workshops (e.g. "VLDB Workshop ...")
my $connectors="(?:on|of)";
my $abbreviations="(?:\\([A-Z]\\w\\w+[\\W\\s]*?(?:\\d\\d+)?\\))"; # Conference abbreviations like "(SIGMOD
# The actual pattern we search for.  A typical conference name this pattern will find is
# "3rd International Conference on Blah Blah Blah (ICBBB-05)"
my
$fullNamePattern="((?:$ordinals\\s+$words*|$confDescriptors)?$confTypes(?:\\s+$connectors\\s+.*?|\\s
reviations?)(?:\\n|\\r|\\.|<)";
################################ ##################################
# Given a <dbworldMessage>, look for the conference pattern
################################################################
lookForPattern($dbworldMessage, $fullNamePattern);
#########################################################
# In a given <file>, look for occurrences of <pattern>
# <pattern> is a regular expression
#########################################################
sub lookForPattern {
   my ($file,$pattern) = @_;
```

Gene & Protein Tagger: AliBaba

- Extract gene names from PubMed abstracts
- Use Classifier (Support Vector Machine - SVM)



- Corpus of 7500 sentences
  - 140.000 non-gene words
  - 60.000 gene names
- SVM^{light} on different feature sets
- Dictionary compiled from Genbank, HUGO, MGD, YDB
- Post-processing for compound gene names

Some Hand Coded Entity Taggers

- FRUMP [DeJong 82]
- CIRCUS / AutoSlog [Riloff 93]
- SRI FASTUS [Appelt, 1996]
- **MITRE Alembic** (available for use)
- **Alias-I LingPipe** (available for use)
- OSMX [Embley, 2005]
- DBLife [Doan et al, 2006]
- Avatar [Jayram et al, 2006]

Machine Learning Methods

- Can work well when training data is easy to construct and is plentiful
- Can capture complex patterns that are hard to encode with hand-crafted rules
  - e.g., determine whether a review is positive or negative
  - extract long complex gene names

[From AliBaba]

> ***The human T cell leukemia lymphotropic virus type 1 Tax protein represses MyoD-dependent transcription by inhibiting MyoD-binding to the KIX domain of p300.“***

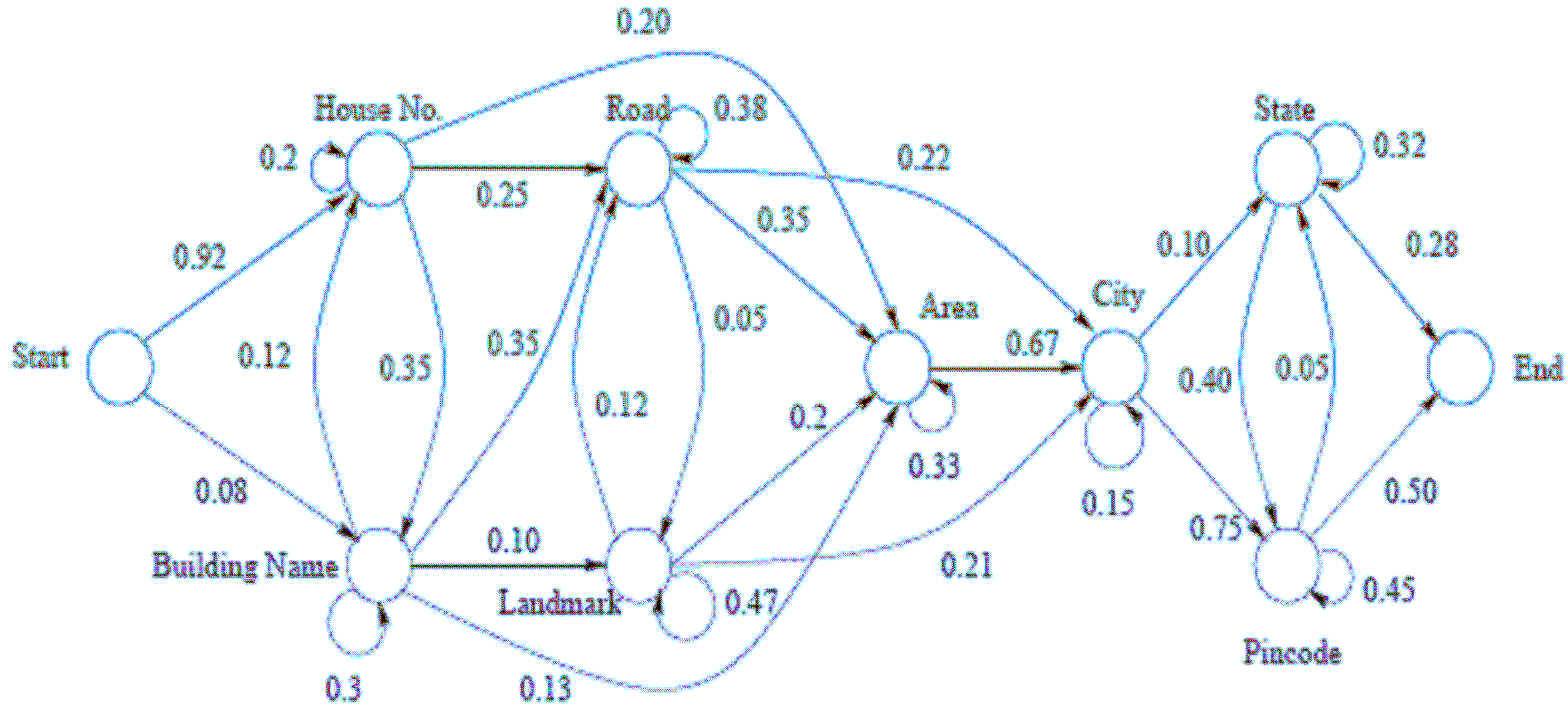
- Can be labor intensive to construct training data
  - Question: how much training data is sufficient?

Popular Machine Learning Methods for IE

- Naive Bayes
- SRV [Freitag-98], Inductive Logic Programming
- Rapier [Califf & Mooney-97]
- Hidden Markov Models [Leek, 1997]
- Maximum Entropy Markov Models [McCallum et al, 2000]
- Conditional Random Fields [Lafferty et al, 2000]
  - Implementations available:
    - Mallet (Andrew McCallum)
    - crf.sourceforge.net (Sunita Sarawagi)
    - MinorThird minorthird.sourceforge.net (William Cohen)

For details: [Feldman, 2006 and Cohen, 2004]

Example of State-based ML Method



Extracted Entities: Resolving Duplicates

Document 1: *The Justice Department has officially ended its inquiry into the assassinations of* **John F. Kennedy** *and Martin Luther King Jr., finding ``no persuasive evidence'' to support conspiracy theories, according to department documents. The House Assassinations Committee concluded in 1978 that* **Kennedy** *was ``probably'' assassinated as the result of a conspiracy involving a second gunman, a finding that broke from the* ***Warren Commission*** *'s belief that Lee Harvey Oswald acted alone in* ***Dallas*** *on Nov. 22, 1963.*

Document 2: *In 1953, Massachusetts* **Sen. John F. Kennedy** *married Jacqueline Lee Bouvier in Newport, R.I. In 1960, Democratic presidential candidate* **John F. Kennedy** *confronted the issue of his Roman Catholic faith by telling a Protestant group in Houston, ``I do not speak for my church on public matters, and the church does not speak for me.''*

Document 3: David Kennedy *was born in Leicester, England in 1959. …***Kennedy** *co-edited The New Poetry (Bloodaxe Books 1993), and is the author of New Relations: The Refashioning Of British Poetry 1980-1994 (Seren 1996).*

[From Li, Morie, & Roth, AI Magazine, 2005]

Important Problem, Addressed in Part II

- Appears in numerous real-world contexts
- Plagues many applications
  - Citeseer, DBLife, AliBaba, Rexa, etc.

Outline

- Overview of Information Extraction
  - Entity tagging
  - **Relation extraction**
- Scaling up Information Extraction
  - Focus on scaling up to large collections (where data mining and ML techniques shine)
  - Other dimensions of scalability

Relation Extraction: Disease Outbreaks

- Extract structured relations from text

May 19 1995, Atlanta -- The Centers for Disease Control and Prevention, which is in the front line of the world's response to the deadly **Ebola** epidemic in **Zaire**, is finding itself hard pressed to cope with the crisis…

Information Extraction System (e.g., NYU's Proteus)

Disease Outbreaks in ***The New York Times***

Date	*Disease Name*	*Location*
Jan. 1995	Malaria	Ethiopia
July 1995	Mad Cow Disease	U.K.
Feb. 1995	Pneumonia	U.S.

Example: Protein Interactions

„We show that CBF-A and CBF-C interact with each other to form a CBF-A-CBF-C complex and that CBF-B does not interact with CBF-A or CBF-C individually but that it associates with the CBF-A-CBF-C complex."

$$\text{CBF-A} \overset{\text{interact}}{\underset{\text{complex}}{\longleftrightarrow}} \text{CBF-C}$$

$$\text{CBF-B} \xrightarrow{\text{associates}} \text{CBF-A-CBF-C complex}$$

Relation Extraction

- Typically require Entity Tagging as preprocessing

- Knowledge Engineering
  - Rules defined over lexical items
    - “<company> located in <location>”
  - Rules defined over parsed text
    - “((Obj <company>) (Verb located) (*) (Subj <location>))”
  - Proteus, GATE, …

- Machine Learning-based
  - Learn rules/patterns from examples
    Dan Roth 2005, Cardie 2006, Mooney 2005, …
  - Partially-supervised: bootstrap from “seed” examples
    Agichtein & Gravano 2000, Etzioni et al., 2004, …

- Recently, hybrid models [Feldman2004, 2006]

Example Extraction Rule [NYU Proteus]

```
;;; For <company> appoints <person> <position>

(defpattern appoint
   "np-sem(C-company)?  rn?  sa?  vg(C-appoint) np-sem(C-person) ´,´?
    to-be?  np(C-position) to-succeed?:
    company-at=1.attributes, sa=3.span, lv=4.span, person-at=5.attribute
    position-at=8.attributes |
...
```

```
(defun when-appoint (phrase-type)
    (let ((person-at (binding ´person-at))
        (company-entity (entity-bound ´company-at))
        (person-entity (essential-entity-bound ´person-at ´C-person))
        (position-entity (entity-bound ´position-at))
        (predecessor-entity (entity-bound ´predecessor-at))
        new-event)
     (not-an-antecedent position-entity)
     ;; if no company is specified for position, use agent
...
```

Example Extraction Patterns: Snowball [AG2000]

ORGANIZATION {<'s 0.7> <in 0.7> <headquarters 0.7>} LOCATION

LOCATION {<- 0.75> <based 0.75>} ORGANIZATION

Accuracy of Information Extraction

Information Type	Accuracy
Entities	90-98%
Attributes	80%
Facts	60-70%
Events	50-60%

[Feldman, ICML 2006 tutorial]

- Errors cascade (error in entity tag → error in relation extraction)

- This estimate is optimistic:
  - Holds for well-established tasks
  - Many specific/novel IE tasks exhibit lower accuracy

Outline

- Overview of Information Extraction
  - Entity tagging
  - Relation extraction
- **Scaling up Information Extraction**
  - Focus on scaling up to large collections (where data mining and ML techniques shine)
  - Other dimensions of scalability

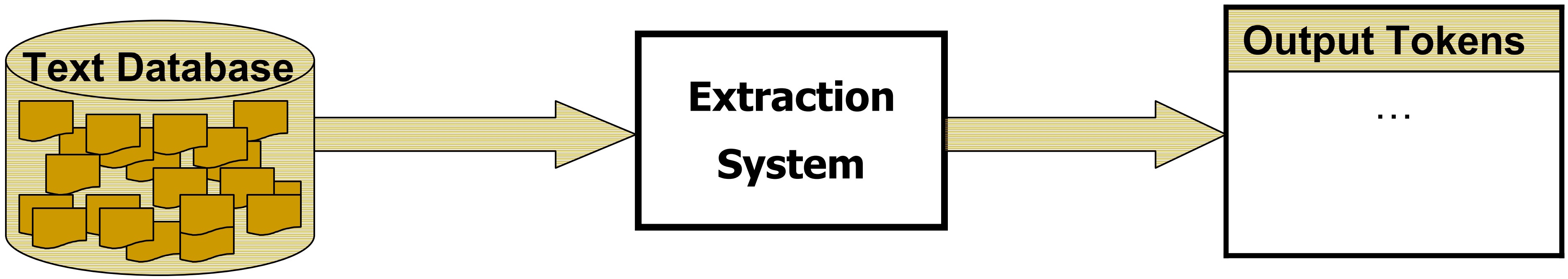
Dimensions of Scalability

- Efficiency/corpus size
  - Years to process a large collections (centuries for Web)
- Heterogeneity/diversity of information sources
  - Requires many rules (expensive to apply)
  - Many sources/conventions (expensive to maintain rules)
- Accessing required documents
  - Hidden Web databases are not crawlable
- Number of Extraction Tasks (not covered)
  - Many patterns/rules to develop and maintain
  - Open research area

Scaling Up Information Extraction

- Scan-based extraction
  - Classification/filtering to avoid processing documents
  - Sharing common tags/annotations
- General (keyword) index-based techniques
  - QXtract, KnowItAll
- Specialized indexes
  - BE/KnowItNow, Linguist’s Search Engine
- Parallelization/Adaptive Processing
  - IBM WebFountain, Google’s Map/Reduce
- Application: Question Answering
  - AskMSR, Arranea, Mulder

Scan



1. Retrieve docs from database
2. Process documents
3. Extract output tokens

- **Scan** retrieves and processes documents sequentially (***until reaching target recall***)

Execution time = |Retrieved Docs| · (R + P)

Time for retrieving a document

Time for processing a document

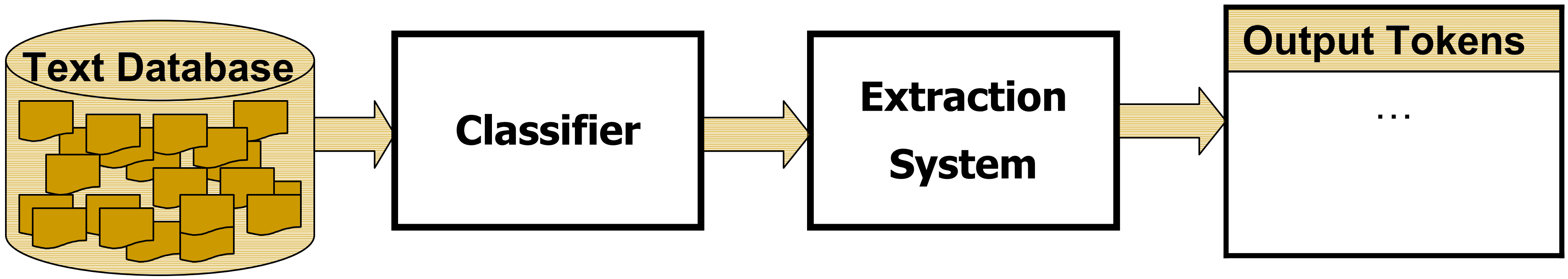
Efficient Scanning for Information Extraction

- 80/20 rule: use few simple rules to capture majority of the cases [PRH2004]

- Train a classifier to discard irrelevant documents without processing [GHY2002]

- Share base annotations (entity tags) across multiple tasks

Filtered Scan



1. Retrieve docs from database
2. Filter documents
3. Process **filtered** documents
4. Extract output tokens

- **Scan** retrieves and processes all documents (***until reaching target recall***)
- **Filtered Scan** uses a classifier to identify and process only promising documents *(e.g., the Sports section of NYT is unlikely to describe disease outbreaks)*

Execution time = |Retrieved Docs| * (R + F + σ P)

Time for retrieving a document

Time for filtering a document

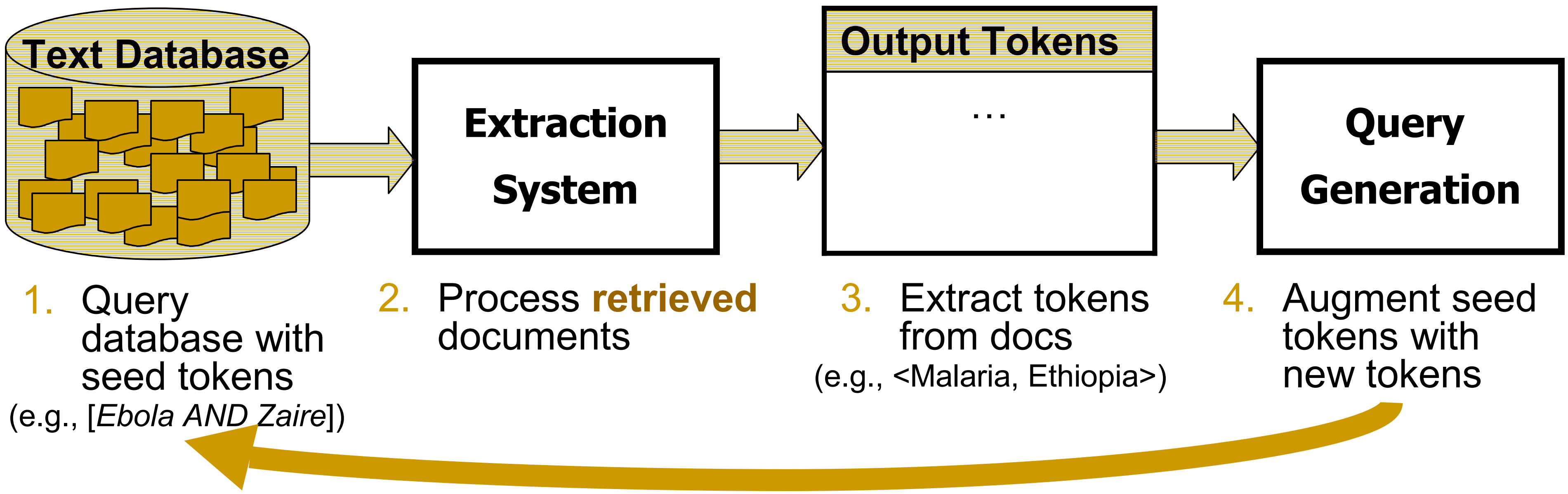
Classifier selectivity (σ≤1)

Time for processing a document

Exploiting Keyword and Phrase Indexes

- Generate queries to retrieve only relevant documents
- Data mining problem!
- Some methods in literature:
  - Traversing Query Graphs [AIG2003]
  - Iteratively refine queries [AG2003]
  - Iteratively partition document space [Etzioni et al., WWW 2004]
- Case studies: *QXtract*, KnowItAll

Simple Strategy: Iterative Set Expansion



1. Query database with seed tokens (e.g., [*Ebola AND Zaire*])
2. Process **retrieved** documents
3. Extract tokens from docs (e.g., <Malaria, Ethiopia>)
4. Augment seed tokens with new tokens

Execution time = |Retrieved Docs| * (R + P) + |Queries| * Q

R: Time for retrieving a document

P: Time for processing a document

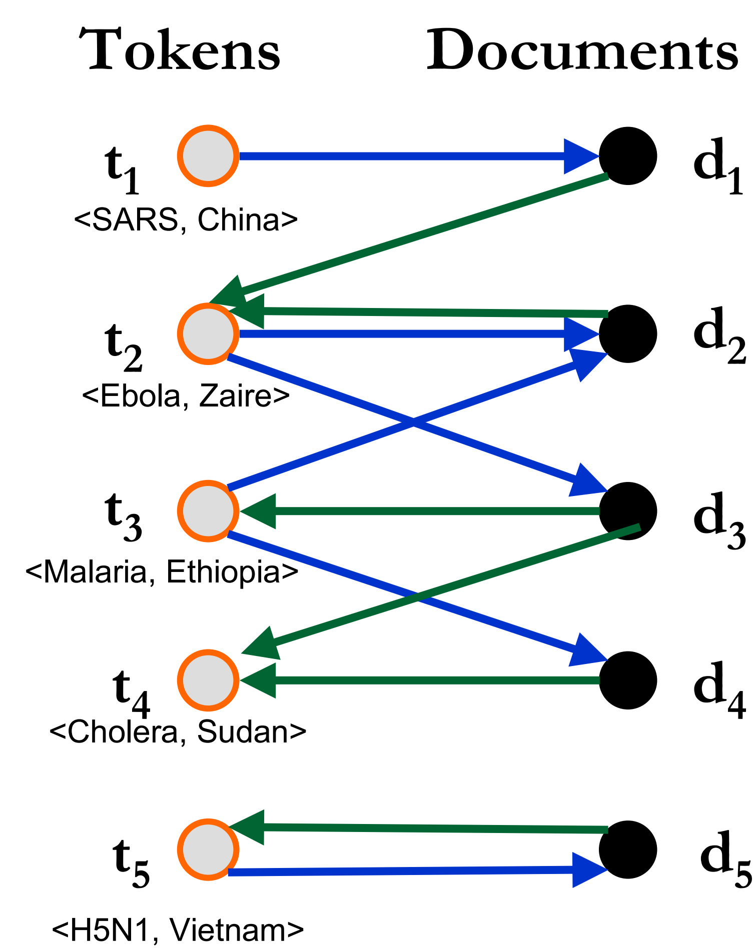
Q: Time for answering a query

Querying Graph

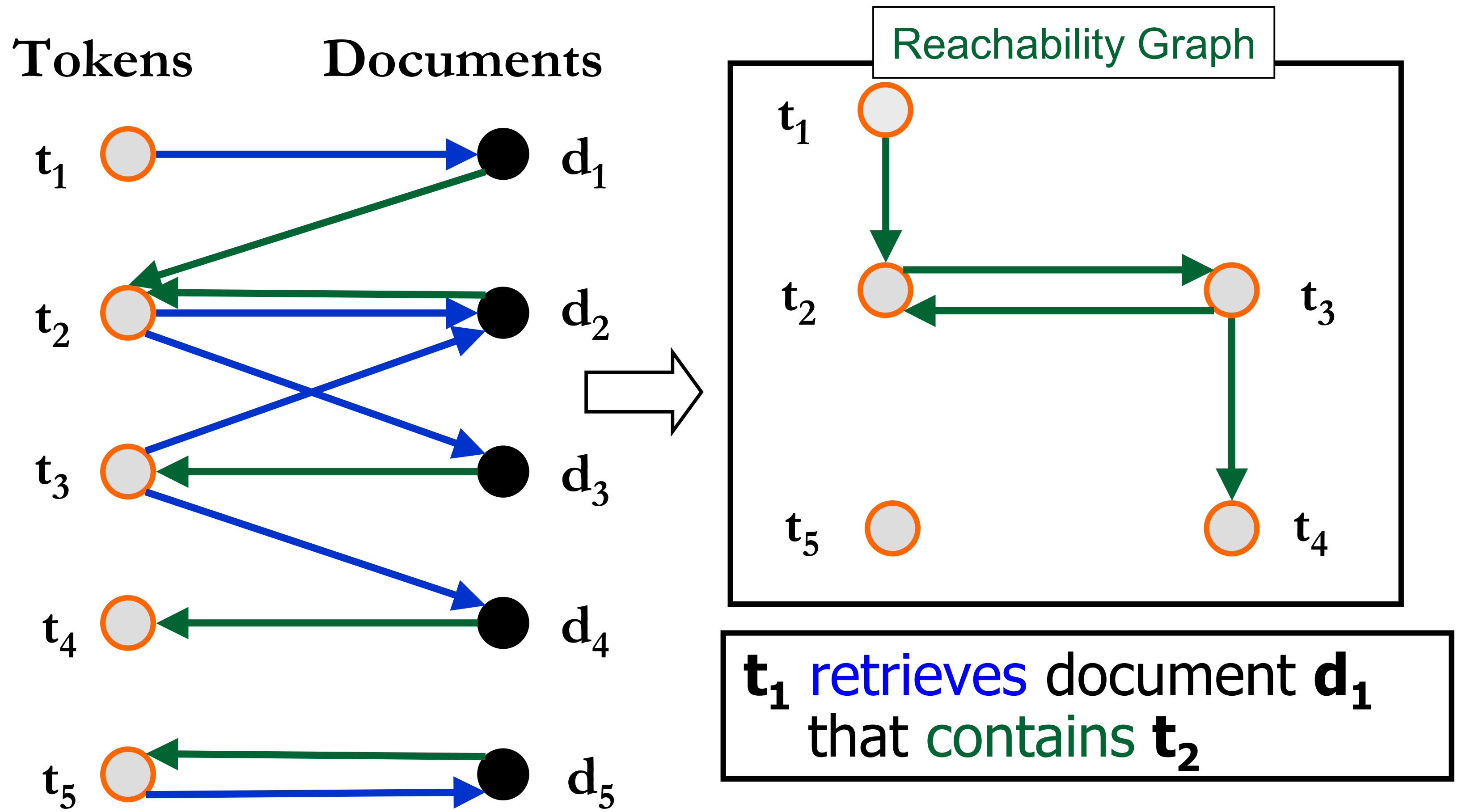
[AIG2003]

- The querying graph is a bipartite graph, containing tokens and documents

- Each token (transformed to a **keyword** query) **retrieves** documents
- Documents **contain** tokens



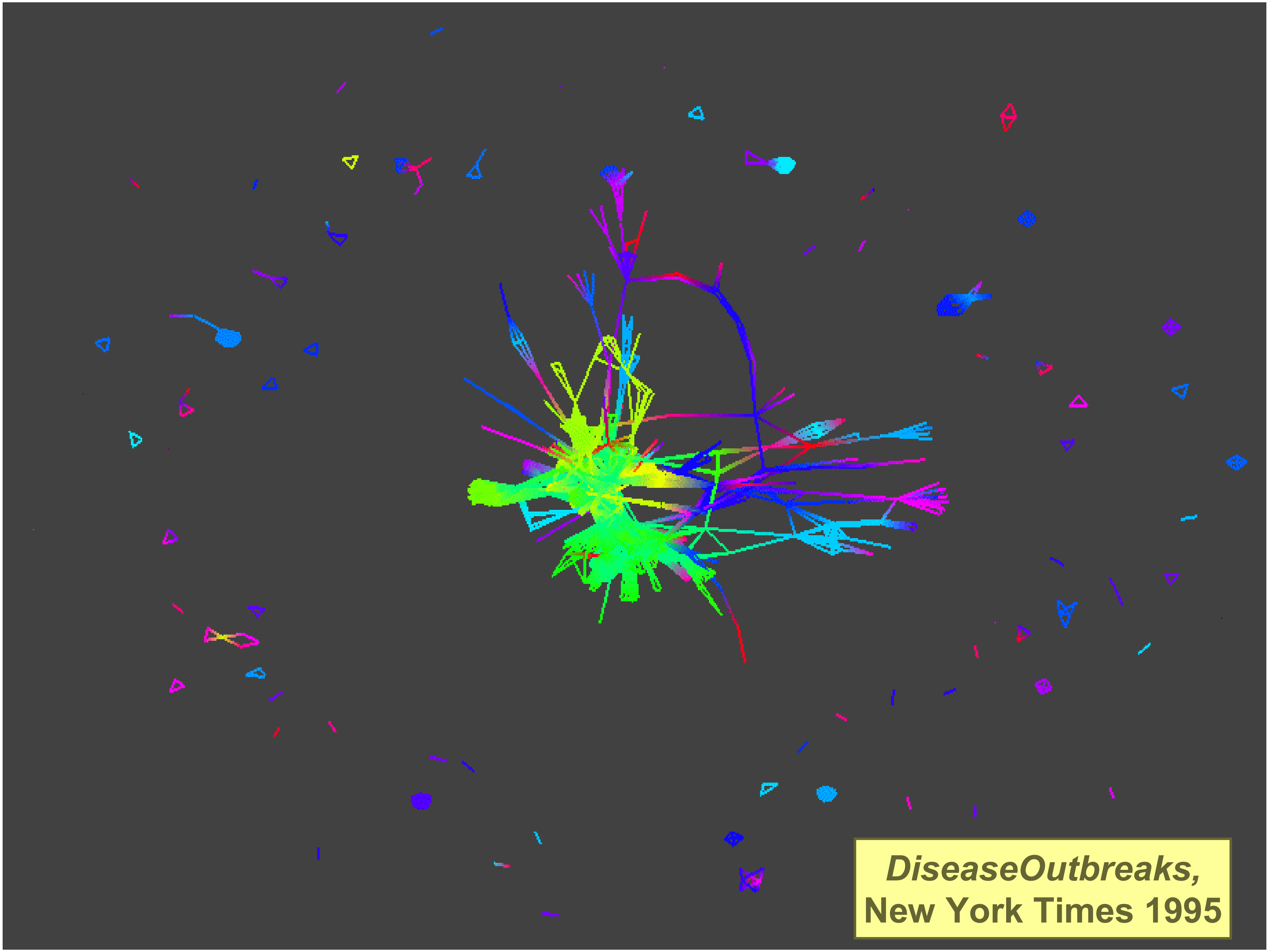
Recall Limit: Reachability Graph



t_1 retrieves document **d_1** that contains **t_2**

Upper recall limit: determined by the size of the biggest connected component

Reachability Graph for DiseaseOutbreaks

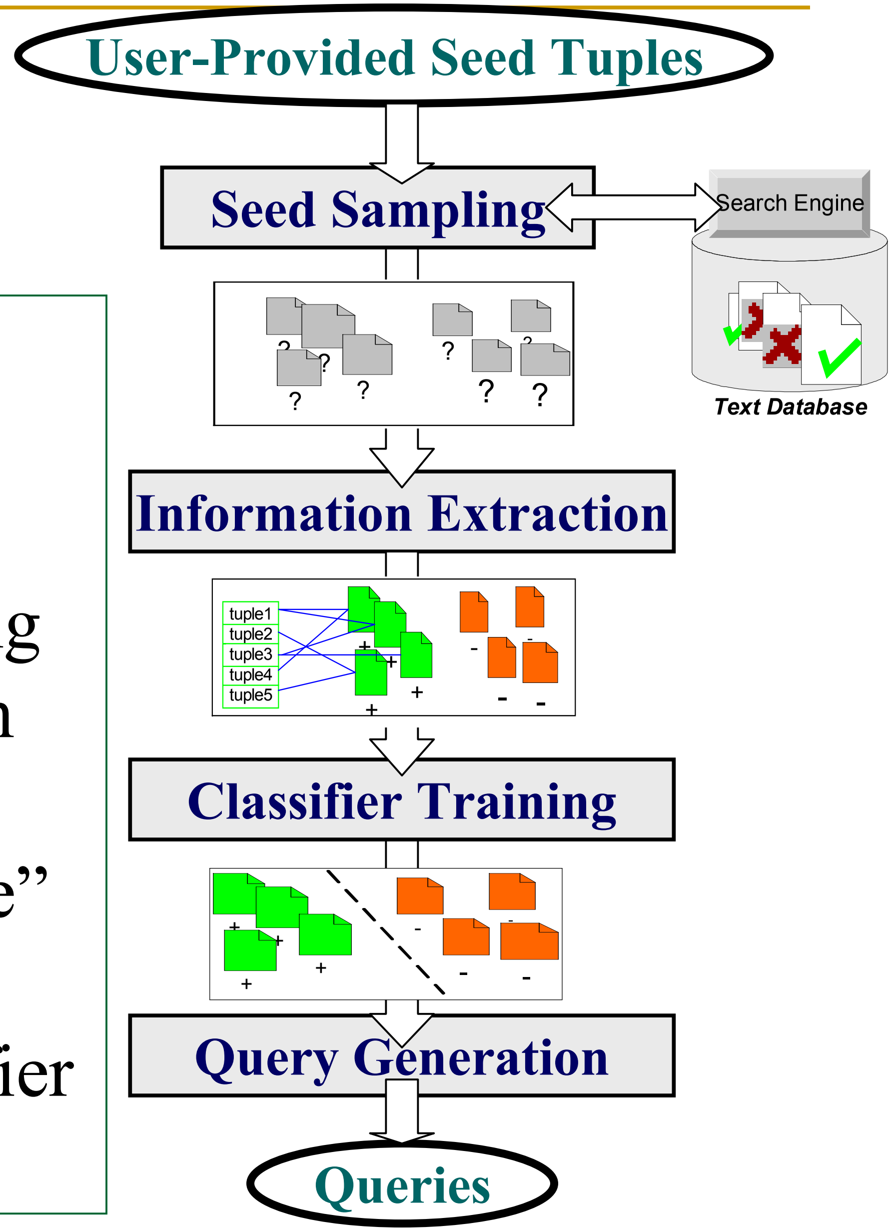

DiseaseOutbreaks,
New York Times 1995

Getting Around Reachability Limit

- KnowItAll:
  - Add keywords to partition documents into retrievable disjoint sets
  - Submit queries with parts of extracted instances
- QXtract
  - General queries with many matching documents
  - Assumes many documents retrievable per query

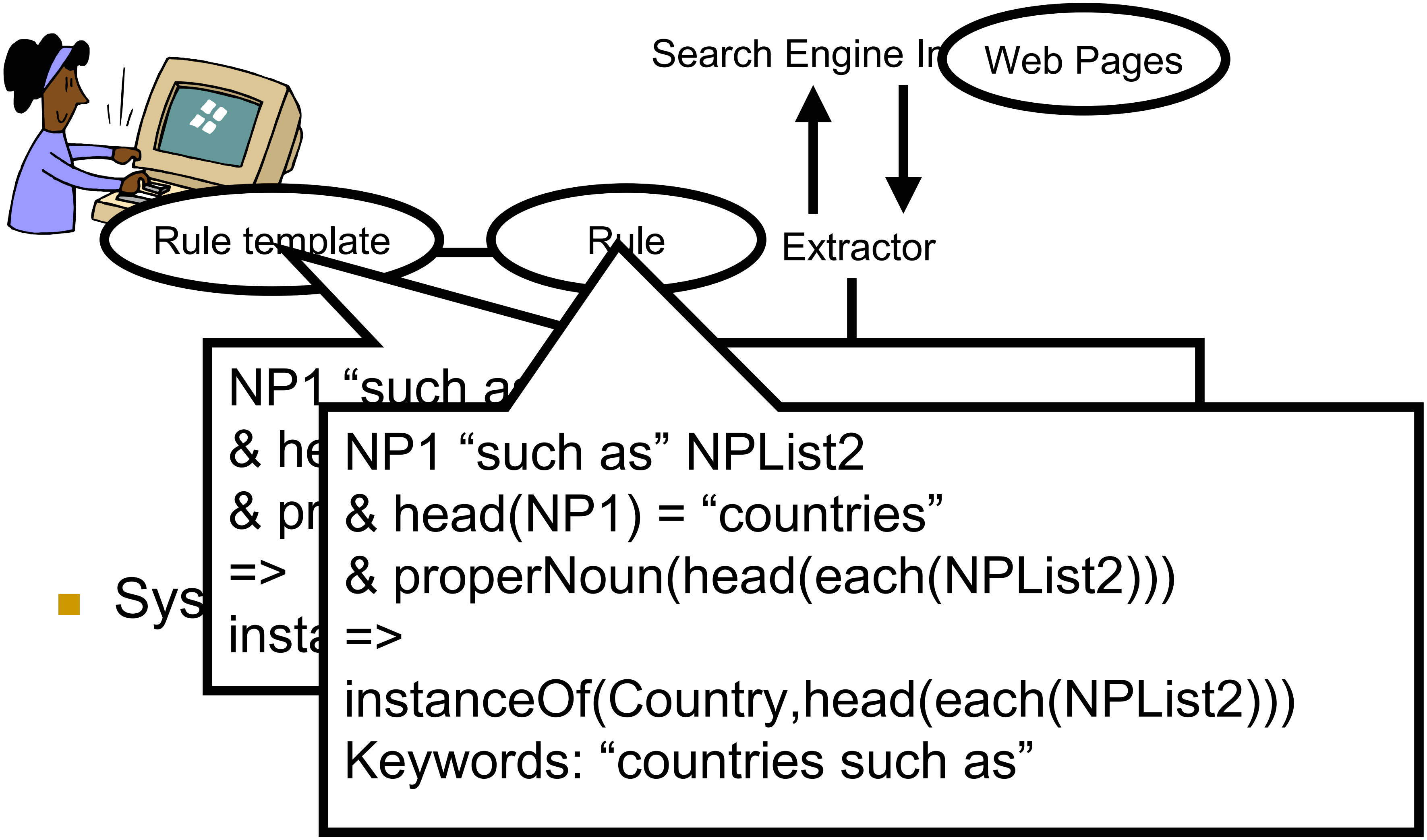
QXtract [AG2003]

1. Get document sample with "likely negative" and "likely positive" examples.
2. Label sample documents using information extraction system as "oracle."
3. Train classifiers to "recognize" useful documents.
4. Generate queries from classifier model/rules.

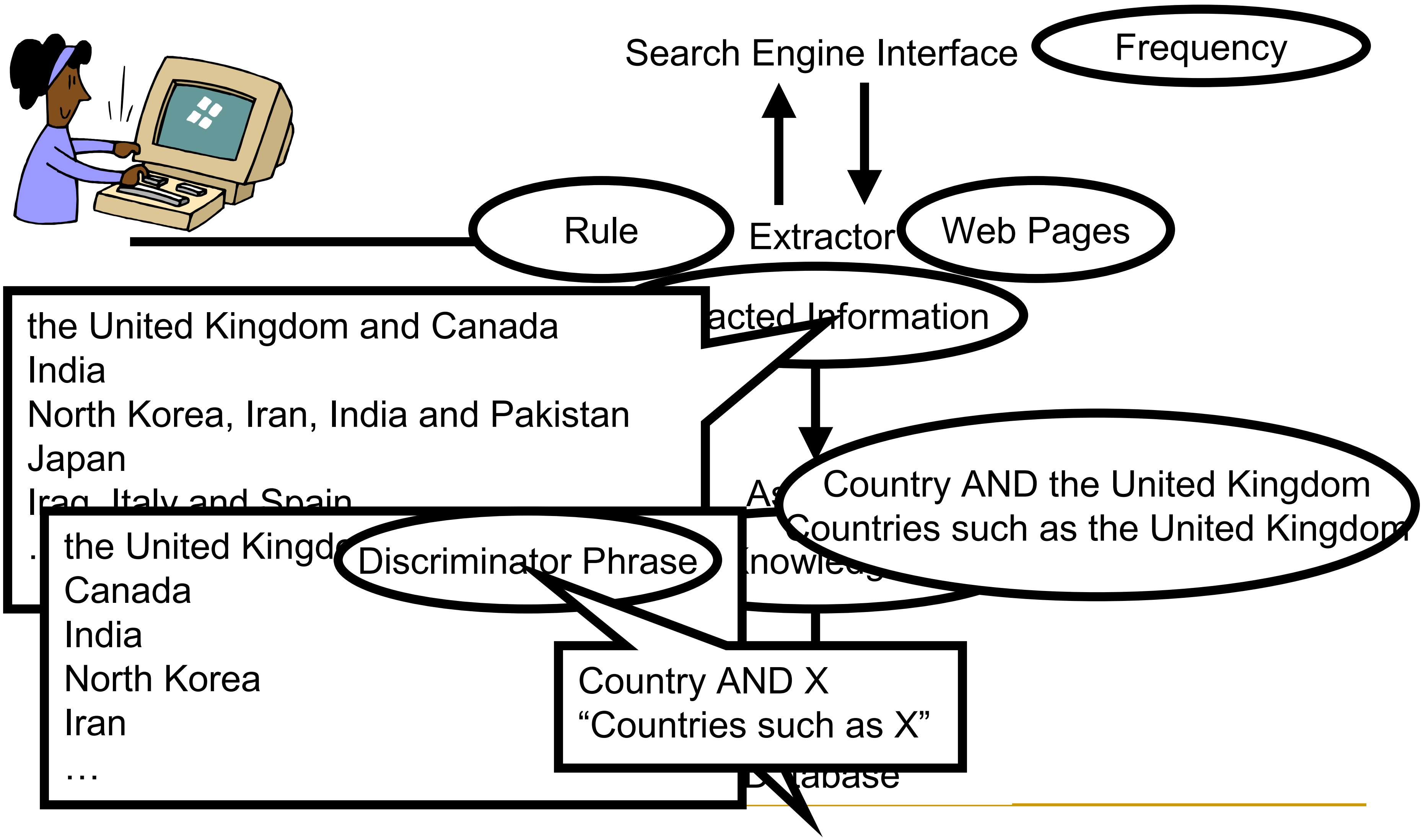


KnowItAll Architecture

Slides: [Zheng Shao, UIUC]



KnowItAll Architecture (Cont.)



Using Generic Indexes: Summary

- Order of magnitude scale-up in corpus size
- Indexes are approximate (queries not precise)
- Require many documents to retrieve

- Can we do better?

Index Structures for Information Extraction

- **Bindings Engine [CE2005]**
- Indexes of entities: [CGHX2006], [IBM Avatar]
- Other systems (not covered)
  - Linguist's search engine (P. Resnik et al.): indexes syntactic structures
  - FREE: Indexing regular expressions: J. Cho et al.

Bindings Engine (BE) [Slides: Cafarella 2005]

- 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

BE Query Support

cities such as <*NounPhrase*>

President Bush <*Verb*>

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

reach me at <*phone-number*>

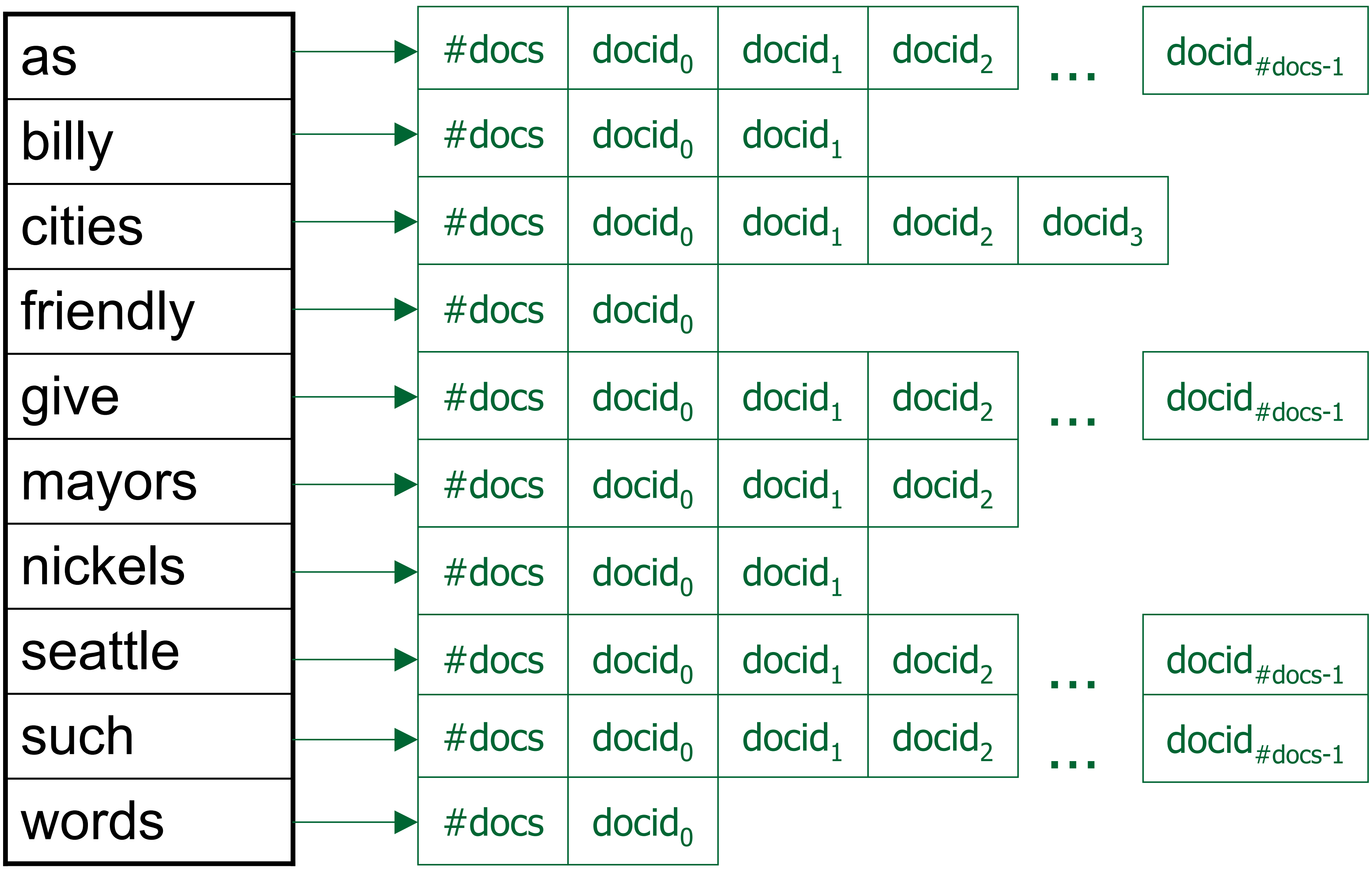
- Any sequence of concrete terms and typed variables
- NEAR is insufficient
- Functions (e.g., "*head(<NounPhrase>)*")

BE Operation

- Like a generic 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

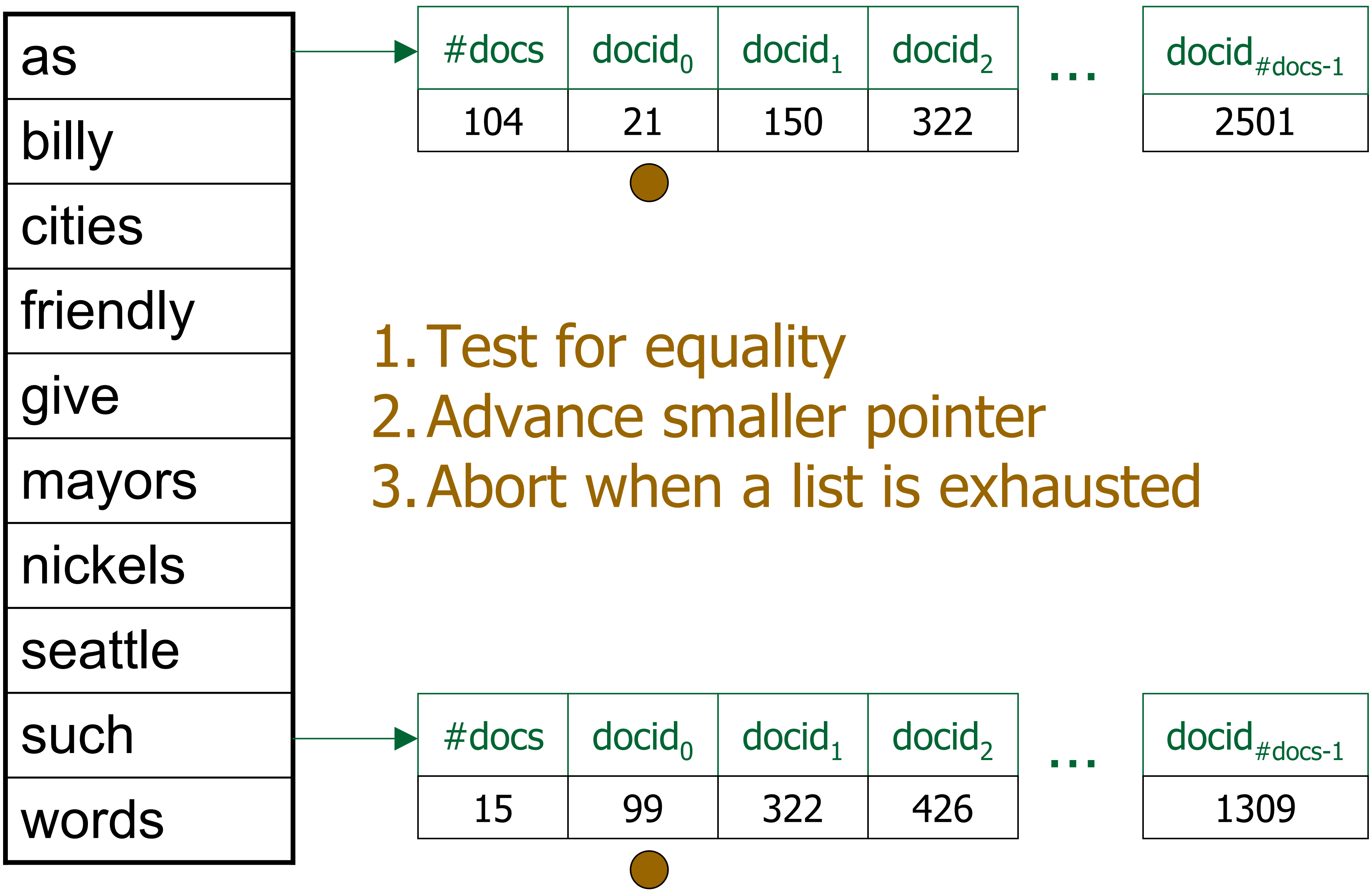
Index design

- 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



as
billy
cities
friendly
give
mayors
nickels
seattle
such
words
#docs docid0 docid1 docid2 ... docid#docs-1
#docs docid0 docid1
#docs docid0 docid1 docid2 docid3
#docs docid0
#docs docid0 docid1 docid2 ... docid#docs-1
#docs docid0 docid1 docid2
#docs docid0 docid1
#docs docid0 docid1 docid2 ... docid#docs-1
#docs docid0 docid1 docid2 ... docid#docs-1
#docs docid0

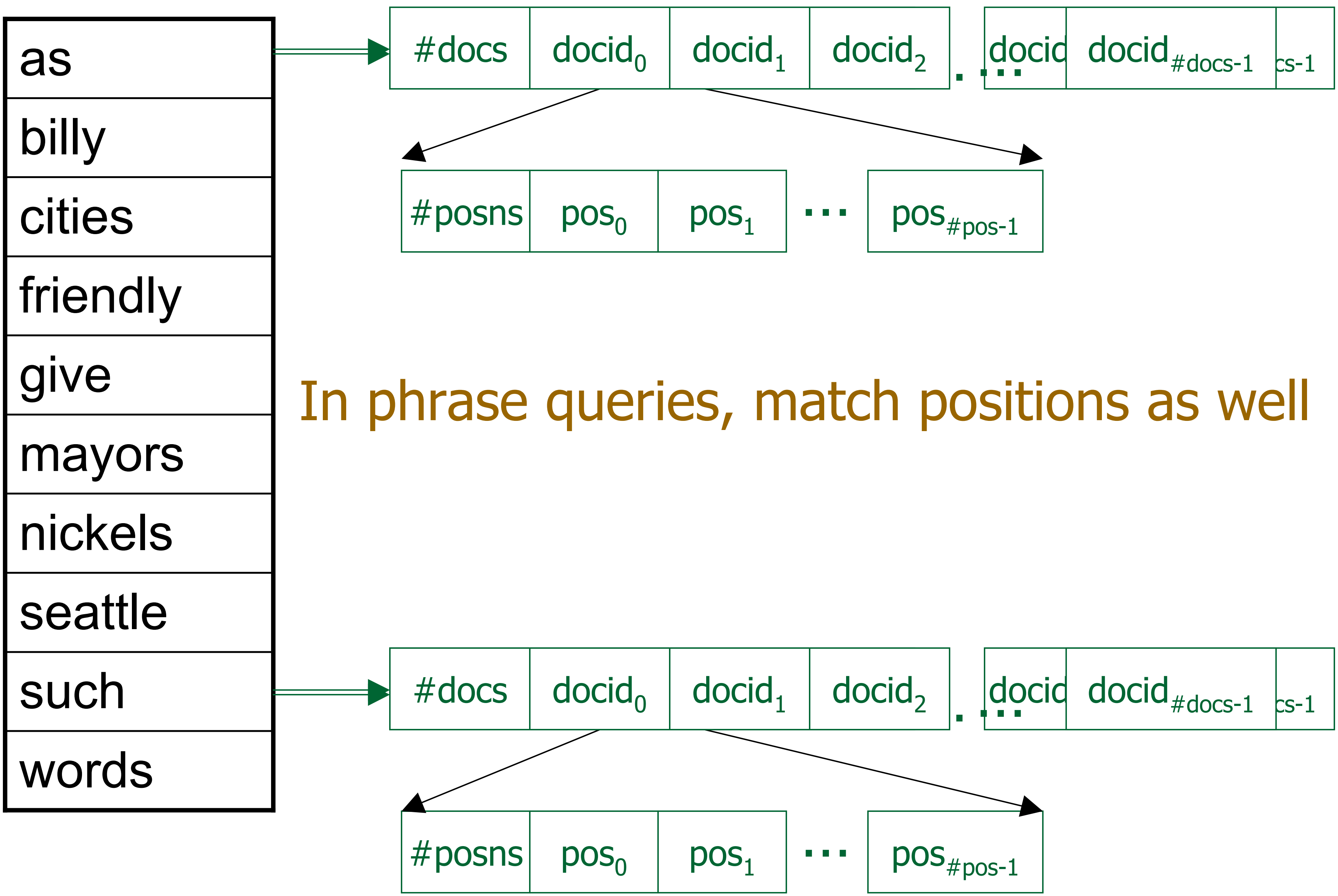
Query: **such as**



1. Test for equality
2. Advance smaller pointer
3. Abort when a list is exhausted

Returned docs: 322

"such as"



In phrase queries, match positions as well

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 Philadelphia.”

Left	Right
	AdjT: “love”

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 Philadelphia.”

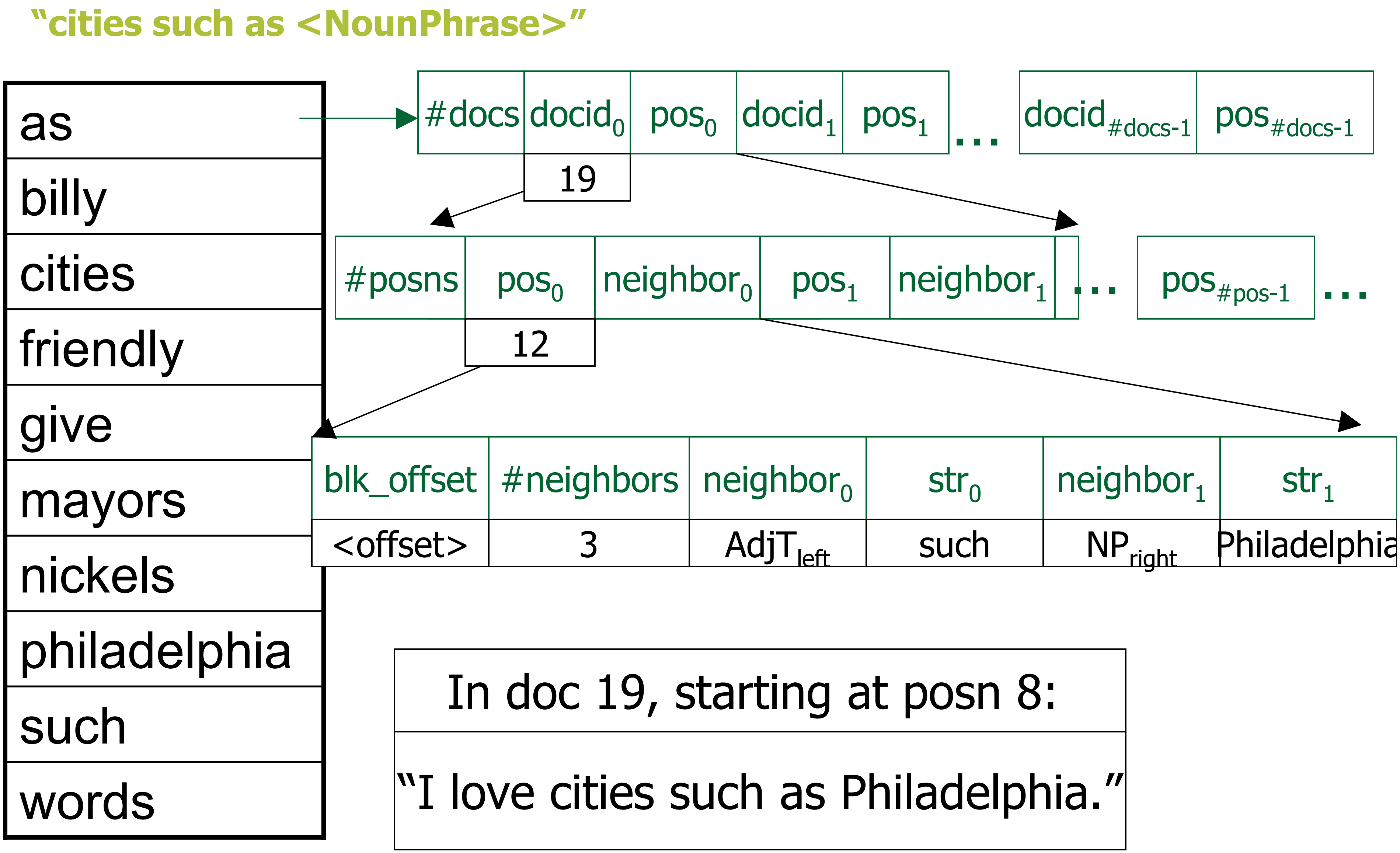
Left	Right
AdjT: “I” NP: “I”	AdjT: “cities” NP: “cities”

Neighbor Index

Query: “cities such as <NounPhrase>”

“I love cities such as Philadelphia.”

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



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

Asymptotic Efficiency 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	$O(k + B)$	$O(N)$

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

Experiment 2: KnowItAll on BE

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

Experiment 2: KnowItAll on BE

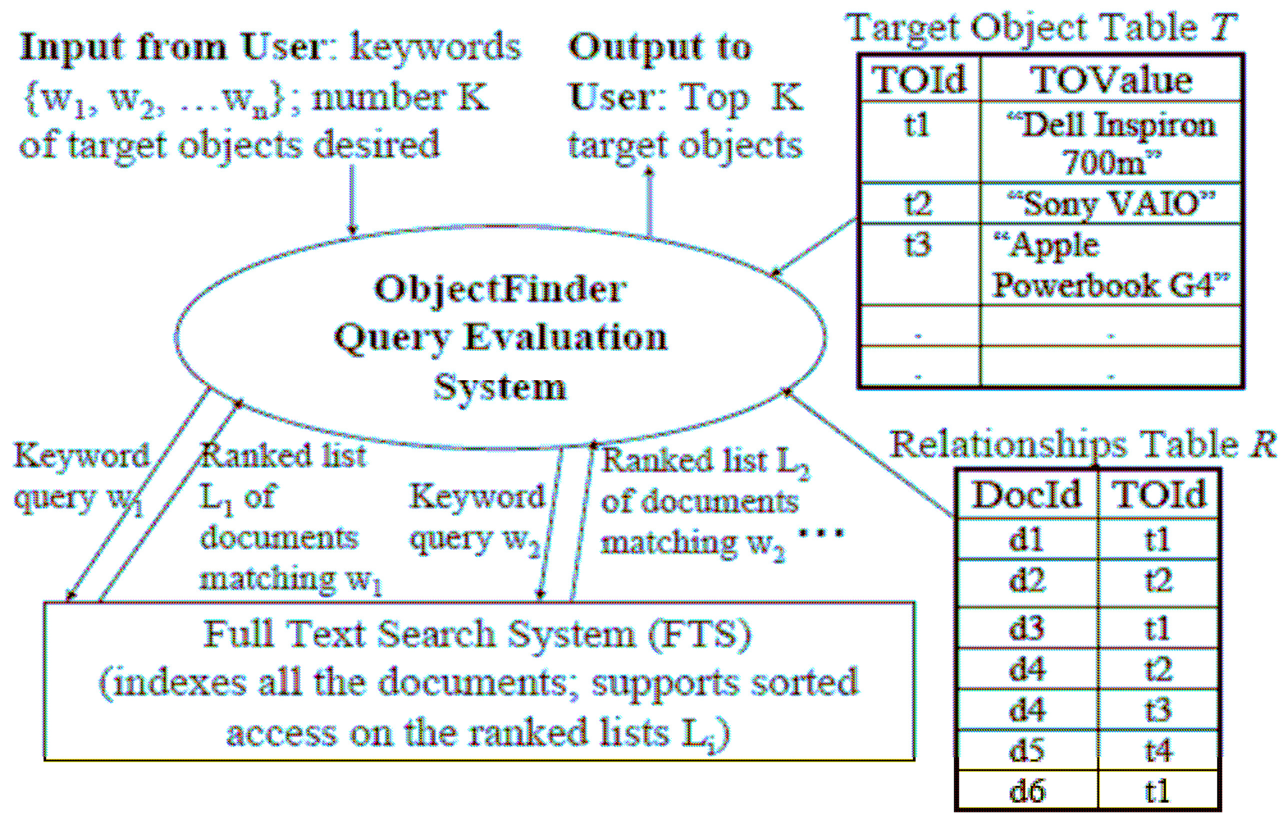
Num Extractions	Std Imp/ Google	BE	Speedup
10k	5,976s	95s	63x
50k	29,880s	95s	314x
150k	89,641s	N/A	N/A

BE Summary

- Significant improvement over generic indexes
- Index size grows linearly with number of types
- Some ML-based patterns (e.g., HMMs, CRFs, character models) not supported
- Can we use it for general QA, RE tasks?

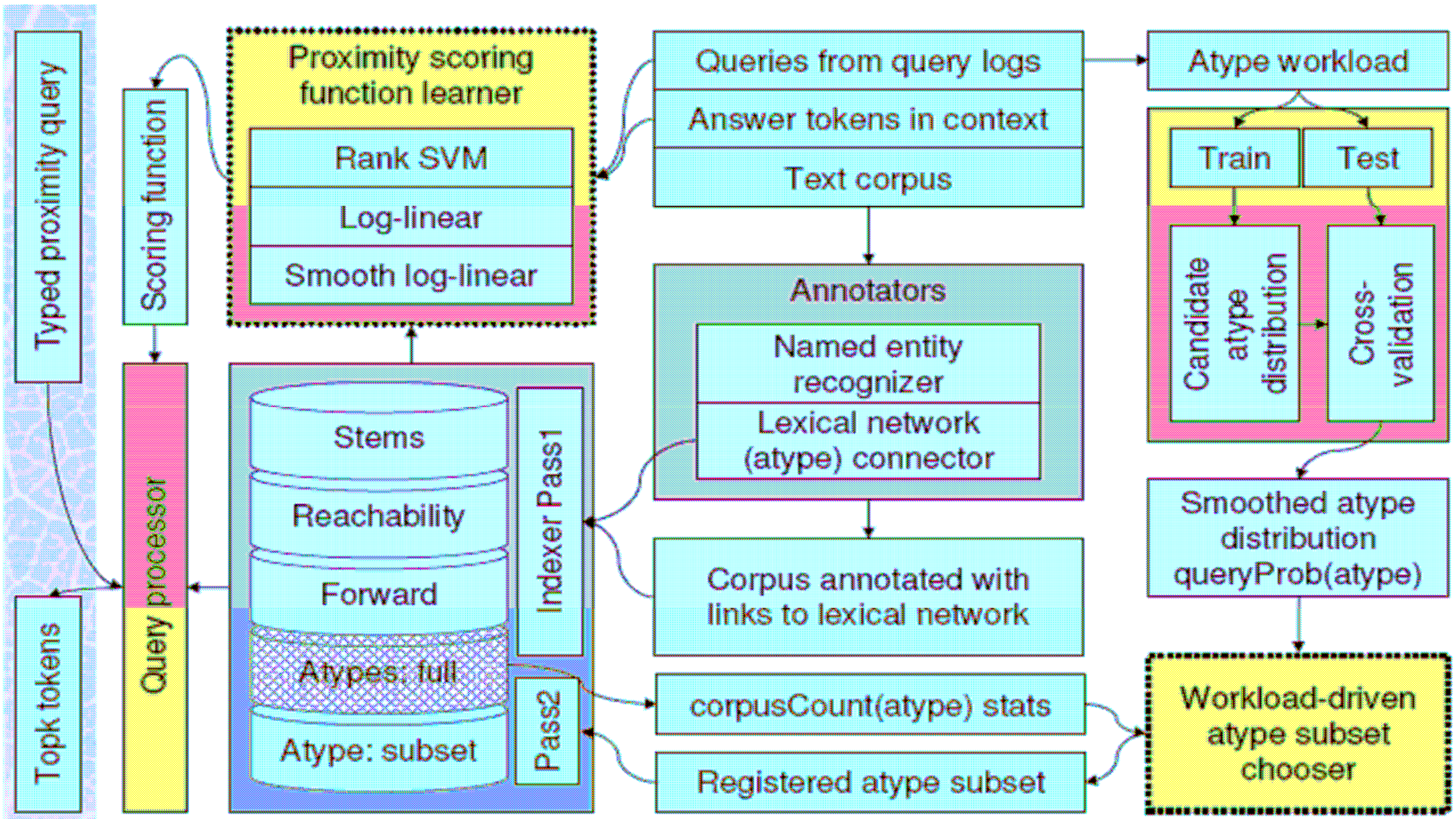
Similar Approach: [CGHX2006]

- Support “relationship” keyword queries over indexed entities
- Top-K support for early termination



Indexing Thousands of Entity Types

[Slides from Chakrabarti et al., WWW 2006]



Workload-Driven Indexing

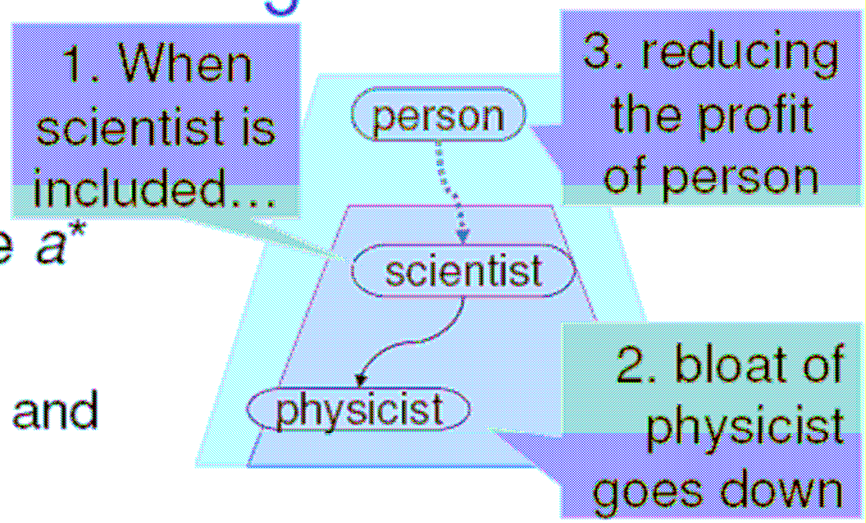
- Type hierarchies are large and deep
  - 18000 internal and 80000 leaf types in WordNet
- Runtime atype expansion time-intensive
  - Even WordNet knows 650 scientists, 860 cities…
- Index each token as all generalizations
  - Sagan → physicist, scientist, person, living thing
  - Large index space bloat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

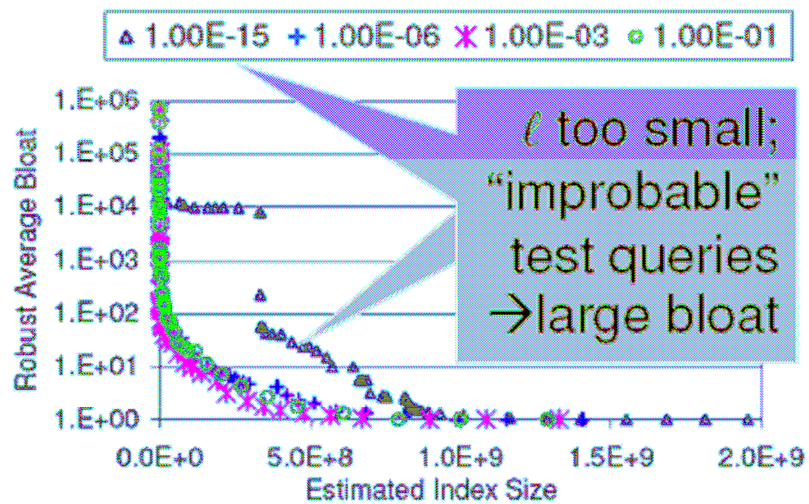
Corpus/Index	Gbytes
Original corpus	5.72
Gzipped corpus	1.33
Stem index	0.91
Full type index	4.30

Selecting Types to Index

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



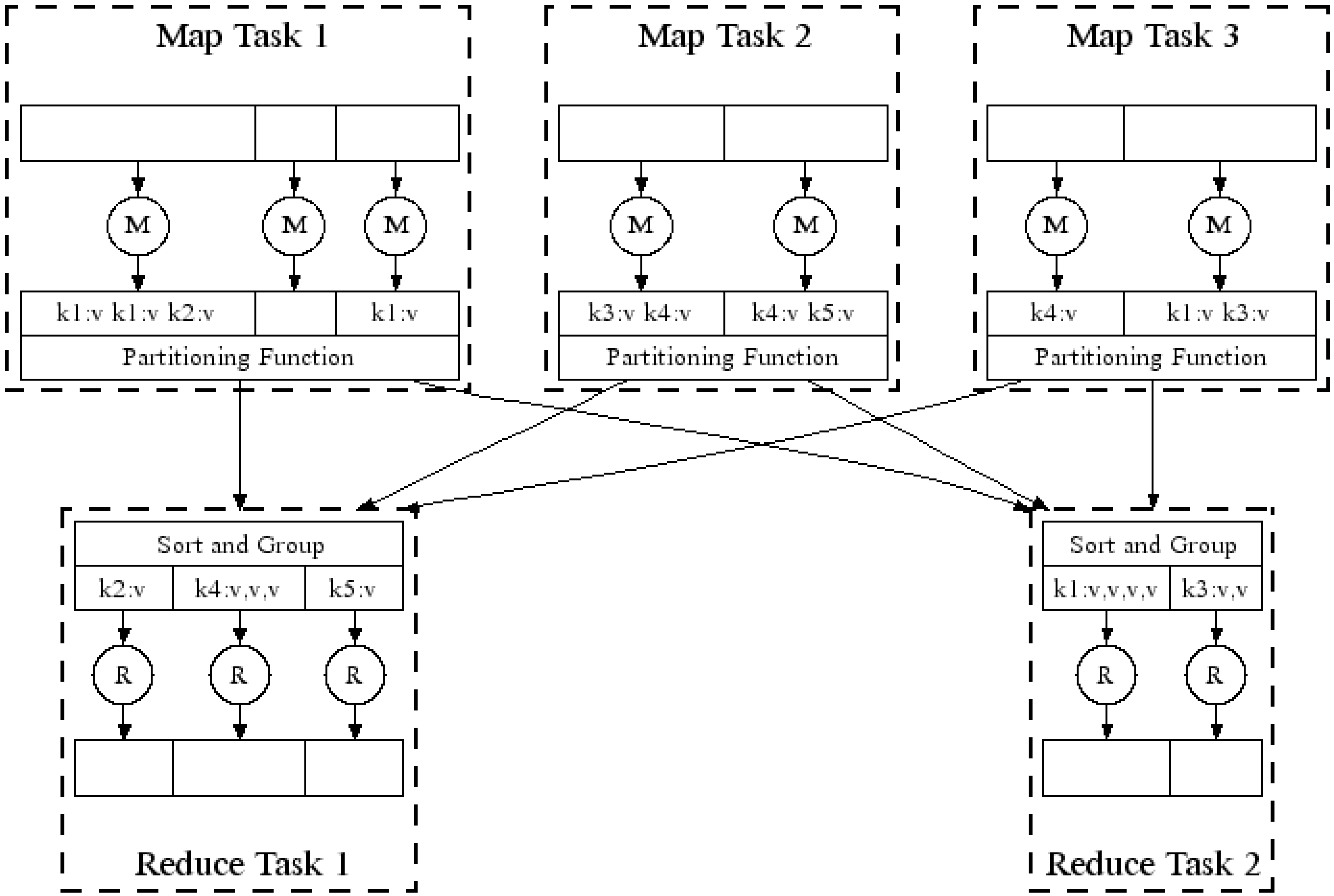


Parallelization/Adaptive Processing

- Parallelize processing:
  - IBM WebFountain [GCG+2004]
  - Google’s Map/Reduce

- Select most efficient access strategy
  - Cost Estimation and Optimization [IAJG2006]

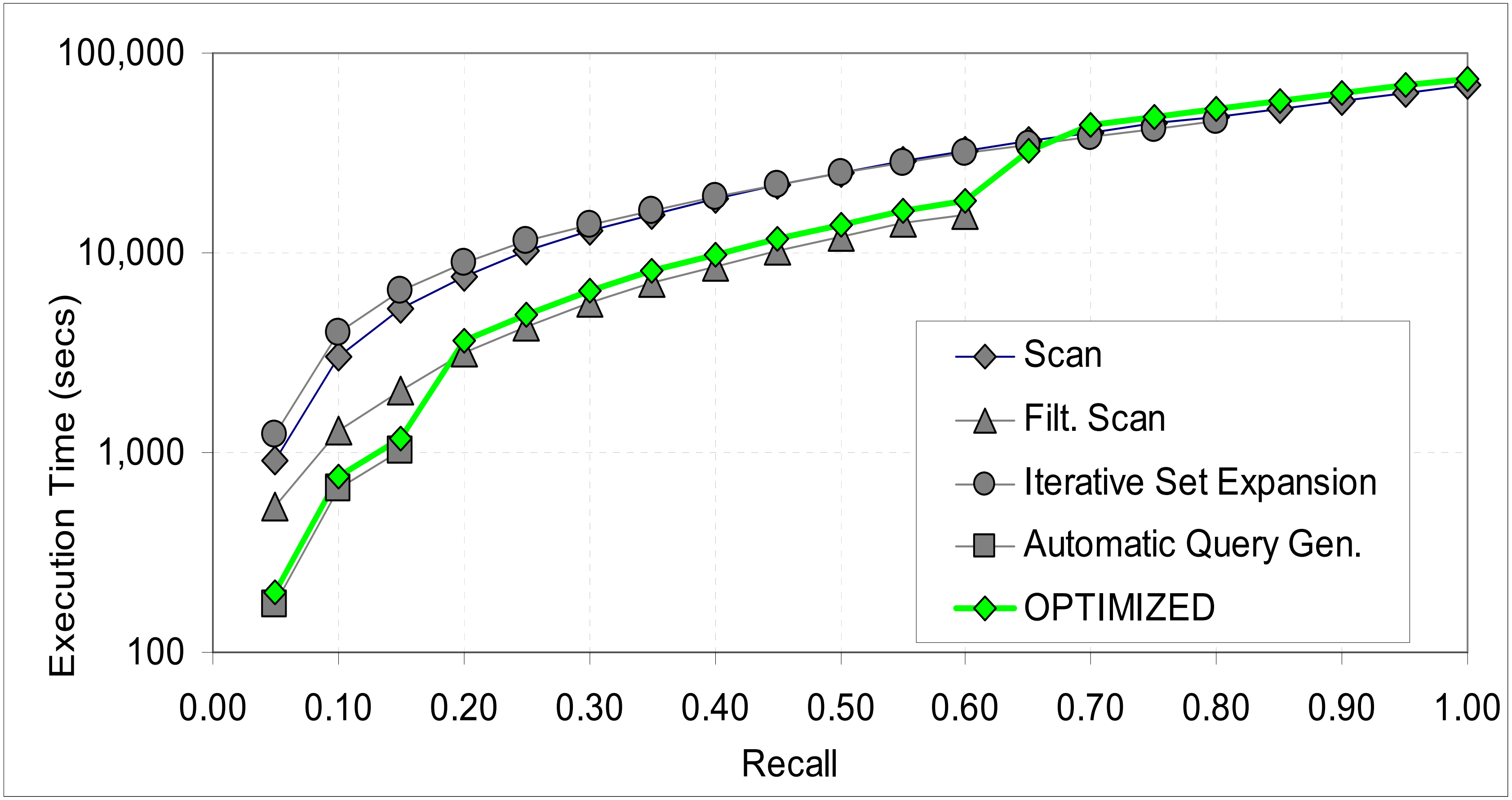
Map/Reduce Framework



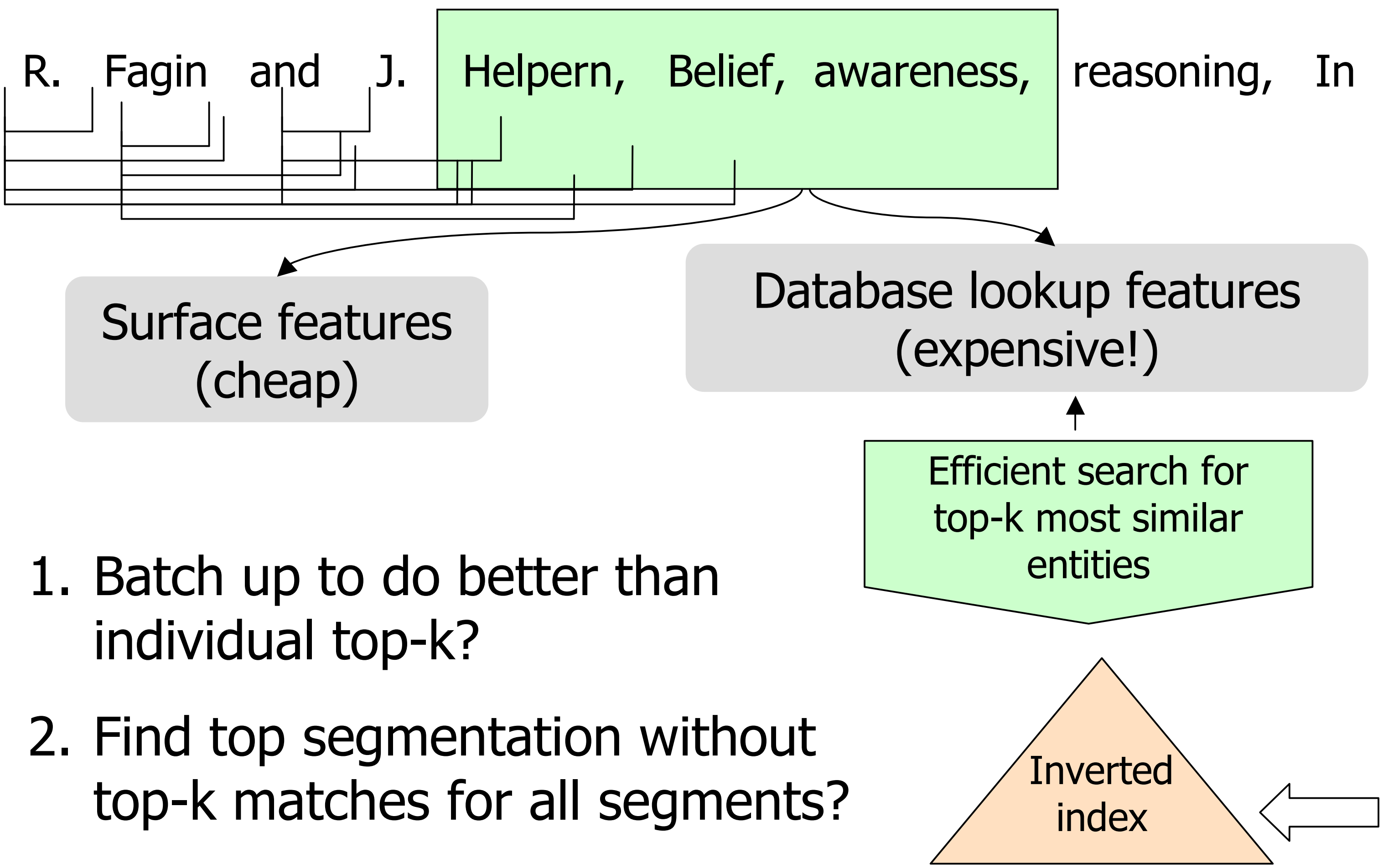
Map/Reduce Framework

- General framework
  - Scales to 1000s of machines
  - Implemented in Nutch
- "Maps" easily to information extraction
  - Map phase:
    - Parse individual documents
    - Tag entities
    - Propose candidate relation tuples
  - Reduce phase
    - Merge multuple mentiones of same relation tuple
    - Resolve co-references, duplicates

Cost Optimizer for Text-Centric Tasks



Other Dimensions of Scalability: Managing Complex Features [CNS2006]

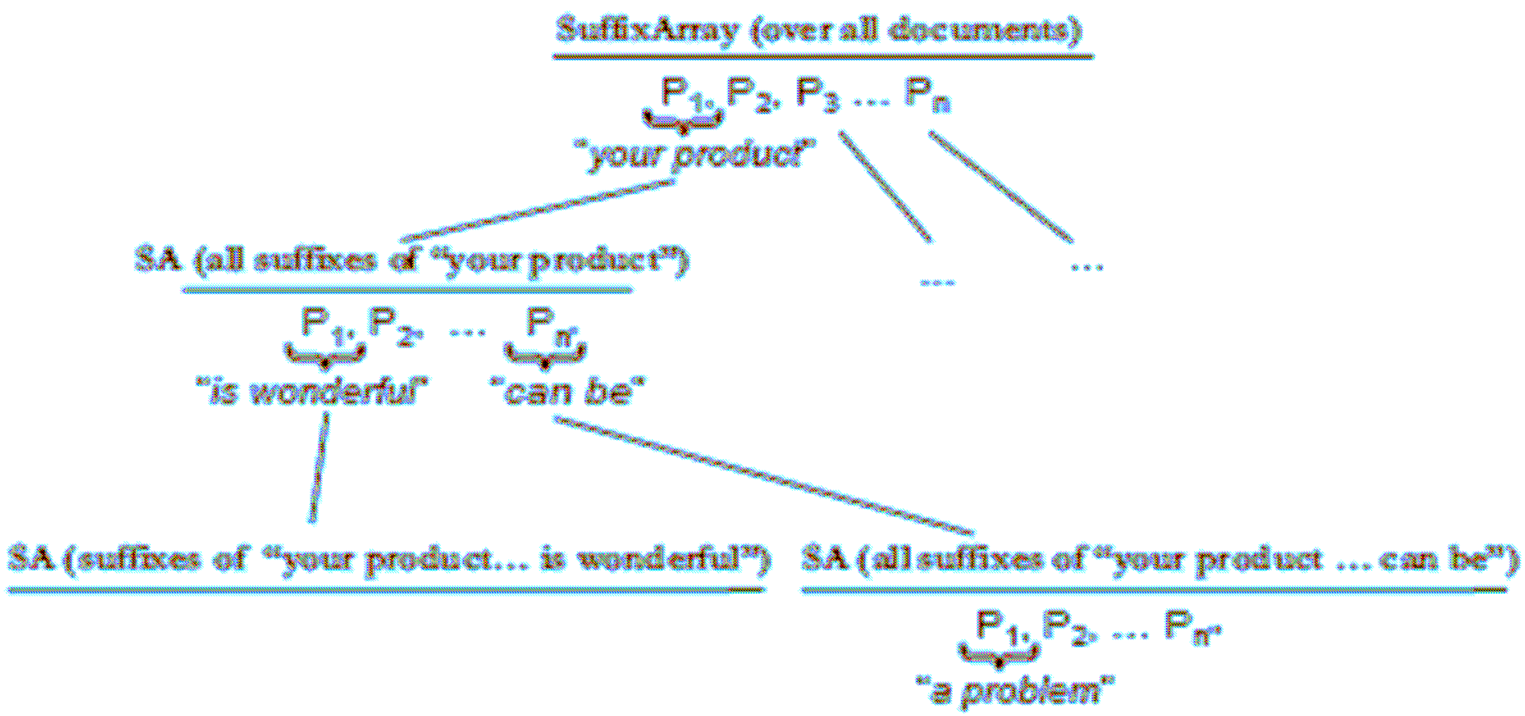


Authors
Ronald Fagin
Steve Cook
S. Sudarshan
S. Chakrabarti
Nick Koudas
R. K. Narayan
E. F. Codd
J. Widom

1. Batch up to do better than individual top-k?
2. Find top segmentation without top-k matches for all segments?

Other Dimensions of Scalability: Extraction Pattern Discovery [Konig and Brill, KDD 2006]

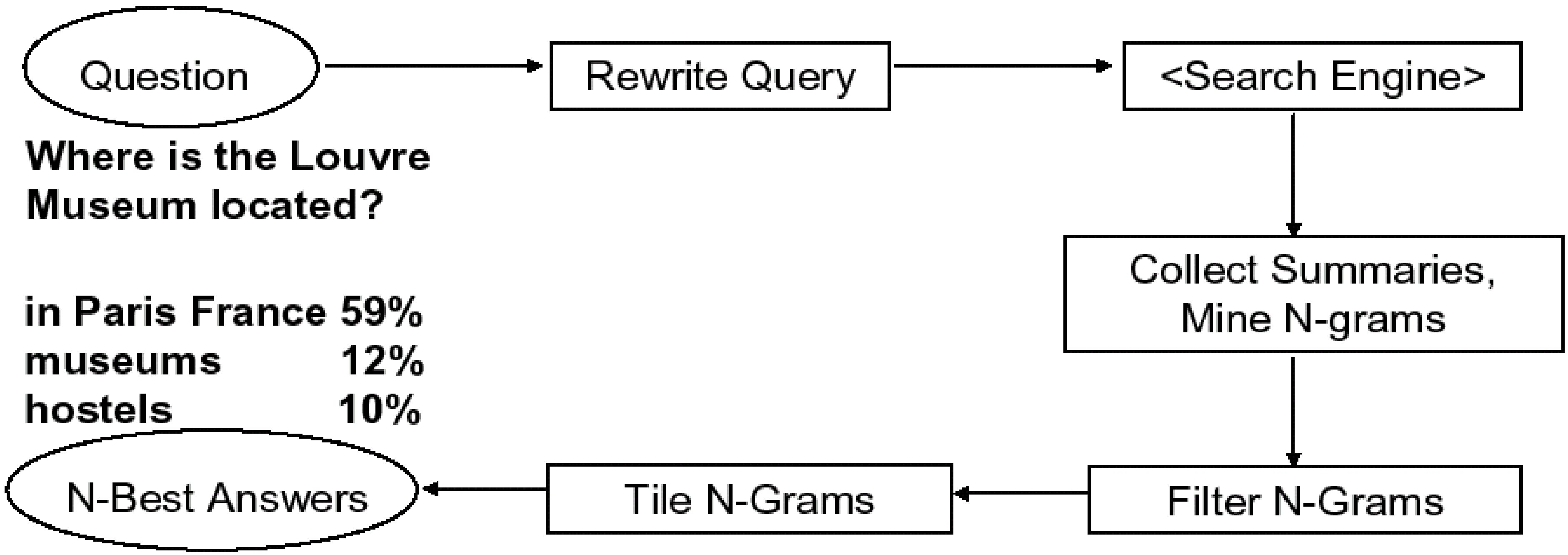
- Use suffix array to efficiently explore candidate patterns



Application: Web Question Answering

AskMSR: does not use patterns

- Simplicity → scalability (cheap to compute n-grams)
- Challenge: do better than n-grams on web QA



Summary

- Brief overview of information extraction from text

- Techniques to scale up information extraction
  - Scan-based techniques (limited impact)
  - Exploiting general indexes (limited accuracy)
  - Building specialized index structures (most promising)

- Scalability is a data mining problem
  - Querying graphs → link discovery
  - Workload mining for index optimization
  - Must be optimized for specific text mining application?

Related Challenges

- Duplicate entities, relation tuples extracted
- Missing values
  - Extraction errors
  - Information spans multiple documents
- Combining relation tuples into complex events

Break

- Eugene Agichtein, Microsoft & Emory University
  - http://www.mathcs.emory.edu/~eugene/
  - eugene@mathcs.emory.edu
- Next: Scalable Information Integration
  - Core set of techniques to enable large-scale IE, text mining
  - Sunita Sarawagi

References

- [AGI2005] E. Agichtein, Scaling Information Extraction to Large Document Collections, IEEE Data Engineering Bulletin, 2005
- [AG2003] E. Agichtein and L. Gravano. Querying text databases for efficient information extraction. ICDE 2003
- [AIG 2003] E. Agichtein, P. Ipeirotis, and L. Gravano, Modeling Query-Based Access to Text Databases, WebDB 2003
- [CDS+2005] I J. Cafarella, D. Downey, S. Soderland, and Oren Etzioni. KnowItNow: Fast, scalable information extraction from the web. (HLT/EMNLP), 2005.
- [CE2005] M. J. Cafarella and O. Etzioni. A search engine for natural language applications. (WWW), 2005
- [CNS2006] A. Chandel, P.C. Nagesh, and S. Sarawagi. Efficient batch top-k search for dictionary-based entity recognition. ICDE 2006
- [CRW2005] S. Chaudhuri, R. Ramakrishnan, and G. Weikum. Integrating db and ir technologies: What is the sound of one hand clapping?, CIDR 2005.
- [CGHX2006] K. Chakrabarti, V. Ganti, Jiawei Han, D. Xin, Ranking Objects Based on Relationships, SIGMOD 2006
- [CPD 2006] S. Chakrabarti, Kriti Puniyani and Sujatha Das, Optimizing Scoring Functions and Indexes for Proximity Search in Type-annotated Corpora. WWW 2006

References II

- [DBB+2002] S. Dumais, M. Banko, E. Brill, J. Lin and A. Ng (2002). P. Bennett, S. Dumais and E. Horvitz (2002). Web question answering: Is more always better? SIGIR 2002
- [GHY2002] R. Grishman, S. Huttunen, and R. Yangarber. Information extraction for enhanced access to disease outbreak reports. *Journal of Biomedical Informatics*, 2002.
- [GCG+2004] D. Gruhl, L. Chavet, D. Gibson, J. Meyer, P. Pattanayak, A. Tomkins, and J. Zien. How to build a WebFountain: An architecture for very large-scale text analytics. IBM Systems Journal, 2004.
- [IAJG2006] Ipeirotis, E. Agichtein, P. Jain, and L. Gravano, To Search or to Crawl: Towards a Query Optimizer for Text-Centric Tasks, SIGMOD 2006
- [KRV+2004] R. Krishnamurthy, S. Raghavan, S. Vaithyanathan, H. Zhu, Avatar: A Database Approach to Semantic Search, SIGMOD 2006
- [PRH2004] P. Pantel, D. Ravichandran, and E. Hovy. Towards terascale knowledge acquisition. In Conference on Computational Linguistics (COLING), 2004.
- [PE2005] P. Resnik and A. Elkiss. The linguist’s search engine: An overview (demonstration). In ACL, 2005.
- [PDT2001] P.D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In European Conference on Machine Learning (ECML), 2001.
- C. König and E. Brill, Reducing the Human Overhead in Text Categorization, KDD 2006

The data integration problem

Extracted entities

- People/organization names,
- Addresses,
- Citations,
- Disease outbreaks

ADDR
11810 WILLS RD ALPHARETTA GA 30076
11810 WILLS ROAD ALPHARETTA GA30004
FLR 110 RM 155 11810 WILLS RD ALPHARETTA GA 300042055
FLR 110 RM MAIN 11810 WILLIS RD ALPHARETTA GA 300042055
FLR 110 RM MAIN 11810 WILLS RD ALPHARETTA GA 30076
FLR BLDG 110 RM 155 11810 WILLS RD ALPHARETTA GA 300042055
FLR BLDG 110 RM MAIN 111810 WILLS RD ALPHARETTA GA 30076
FLR BLDG 110 RM MAIN 11810 WILLS RD ALPHARETTA GA 300042055
FLR MAIN RM 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR MAIN RM BLGD 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR NA RM NA 11810 WILLS RD ALPHARETTA GA 300042055
BLDG 110 FLR 1 RM 110 11810 WILLIS RD ALPHARETTA GA 30004205
BLDG 110 FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042055
BLDG 110 FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042081
BLDG 110 FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 30076
BLDG 110 FLR 1 RM RING 11810 WILLS RD ALPHARETTA GA 30004208
FLR 1 RM 1 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM 110 11801 WILLIS RD ALPHARETTA GA 30076
FLR 1 RM 110 11810 WILLIS RD ALPHARETTA GA 300042055
FLR 1 RM 110 11810 WILLIS RD ALPHARETTA GA 30076
FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 300042081
FLR 1 RM 110 11810 WILLS RD ALPHARETTA GA 30076
FLR 1 RM 110 11810 WILLS RD ATLANTA GA 30076
FLR 1 RM 110 11810 WILLS ROAD ALPHARETTA GA 30076
FLR 1 RM BLDG 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM BLDG 11810 WILLS RD ALPHARETTA GA 300042055
FLR 1 RM COMPUTER 11810 WILLS RD ALPHARETTA GA 300042055

ADDR
11810 WILLS RD ALPHARETTA GA 30076
11810 WILLS ROAD ALPHARETTA GA30004
FLR 110 RM 155 11810 WILLS RD ALPHARETTA GA 300042055
FLR 110 RM MAIN 11810 WILLIS RD ALPHARETTA GA 300042055
FLR 110 RM MAIN 11810 WILLS RD ALPHARETTA GA 30076
FLR BLDG 110 RM 155 11810 WILLS RD ALPHARETTA GA 300042055
FLR BLDG 110 RM MAIN 111810 WILLS RD ALPHARETTA GA 30076
FLR BLDG 110 RM MAIN 11810 WILLS RD ALPHARETTA GA 300042055
FLR MAIN RM 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR MAIN RM BLGD 110 11810 WILLS RD ALPHARETTA GA 300042055
FLR NA RM NA 11810 WILLS RD ALPHARETTA GA 300042055

De-duplication

Multiple related entities with mostly text attributes

The challenge of data integration

- Large lists with multiple noisy mentions of the same entity
- No single key to order or cluster likely duplicates while separating them from similar but different entities.
- Need to depend on fuzzy and compute-intensive string similarity functions
- Cannot afford to compare with mention with every other.

Integration: outline

- Duplicate Entity elimination
  - Defining duplicates: string similarity functions
- Scalable algorithms for finding similar pairs
  - Batch mode
    - Join algorithms to find duplicate pairs
  - Online mode
    - Efficient index design (same as indices in batch mode)
- Create groups from duplicate entity pairs
  - Single entity: data partitions
  - Multiple entities: Collective multi-attribute deduplication

String similarity measures

- Token-based
  - Examples
    - Jaccard
    - TF-IDF Cosine similarities
  - Suitable for large documents
- Character-based
  - Examples:
    - Edit-distance and variants like Levenshtein, Jaro-Winkler
    - Soundex
  - Suitable for short strings with spelling mistakes
- Hybrids

Token based

- Tokens/words
  - 'AT&T Corporation' -> 'AT&T' , 'Corporation'
- Similarity: various measures of overlap of two sets S,T
  - Jaccard(S,T) = $|S \cap T| / |S \cup T|$
  - Example
    - S = 'AT&T Corporation' -> 'AT&T' , 'Corporation'
    - T = 'AT&T Corp' -> 'AT&T' , 'Corp.'
    - Jaccard(S,T) = 1/3
  - Variants: weights attached with each token
- Useful for large strings: example web documents

Cosine similarity with TF/IDF weights

- Cosine similarity:
  - Sets transformed to vectors with each term as dimension
  - Similarity: dot-product of two vectors each normalized to unit length
    - → cosine of angle between them
- Term weight == TF/IDF
  - log (tf+1) * log idf where
    - tf : frequency of 'term' in a document d
    - idf : number of documents / number of documents containing 'term'
  - Intuitively: rare 'terms' are more important
  - Widely used in traditional IR
- Example:
    - 'AT&T Corporation', 'AT&T Corp' or 'AT&T Inc'
      - Low weights for 'Corporation','Corp','Inc', Higher weight for 'AT&T'

Edit Distance [G98]

- Given two strings, S,T, edit(S,T):
  - Minimum cost sequence of operations to transform S to T.
    - Character Operations: I (insert), D (delete), R (Replace).
  - Example: edit(Error,Eror) = 1, edit(great,grate) = 2
- Folklore dynamic programming algorithm to compute edit();
  - $O(m^2)$ versus O(2m log m) for token-based measures
- Several variants (gaps,weights) --- becomes NP-complete easily.
  - Varying costs of operations: can be learnt [RY97].
- Observations
  - Suitable for common typing mistakes on small strings
    - Comprehensive vs Comprenhensive
  - Problematic for specific domains

Edit Distance with affine gaps

- Differences between ‘duplicates’ often due to abbreviations or whole word insertions.
  - IBM Corp. closer to ATT Corp. than IBM Corporation
  - John Smith vs John Edward Smith vs John E. Smith
- Allow sequences of mis-matched characters (gaps) in the alignment of two strings.
- Penalty: using the affine cost model
  - $Cost(g) = s + e \cdot l$
  - s: cost of opening a gap
  - e: cost of extending the gap
  - l: length of a gap
- Similar dynamic programming algorithm
- Parameters domain-dependent, learnable, e.g., [BM03, MBP05]

Hybrids [CRF03]

- Example: Edward, John Vs Jon Edwerd
- Let $S = \{a_1,\ldots,a_K\}$, $T = \{b_1,\ldots b_L\}$ sets of terms:
- Sim(S,T) = $\frac{1}{K}\sum_{i=1}^{K} \max_{j=1}^{L} sim'(a_i, b_j)$
- Sim'() some other similarity function
- $C(t,S,T) = \{w \in S \text{ s.t } \exists\, v \in T, sim'(w,v) > t\}$
- $D(w,T) = \max_{v \in T} sim'(w,v)$, $w \in C(t,S,T)$
  - sTFIDF = $\sum_{w \in C(t,S,T)} W(w,S) * W(w,T) * D(w,T)$

Integration: outline

- Duplicate Entity elimination
  - Defining duplicates: string similarity functions
- Scalable algorithms for finding similar pairs
  - Join algorithms to find duplicate pairs
  - Efficient index design
- Create groups from duplicate entity pairs
  - Single entity: data partitions
  - Multiple entities: Collective multi-attribute deduplication

Finding all duplicate pairs in large lists

- Input: a large list of records R with string attributes
- Output: all pairs (S,T) of records in R which satisfy a Similarity Criteria:
  - Jaccard(S,T) > 0.7
  - Overlapping tokens (S,T) > 5
  - TF-IDF-Cosine(S,T) > 0.8
  - Edit-distance(S,T) < k
- More complicated similarity functions use these as filters (high recall, low precision)
- Naïve method: for each record pair, compute similarity score
  - I/O and CPU intensive, not scalable to millions of records
- Goal: reduce $O(n^2)$ cost to $O(n*w)$, where $w << n$
  - Reduce number of pairs on which similarity is computed

General template for similarity functions

- Sets: r,s

Common tokens

threshold

$$\sum_{w \in (r \cap s)} score(w,r) \times score(w,s) \geq T(r,s)$$

$$\text{Overlap}(r,s) = |r \cap s| \geq t$$

$$\rightarrow T(r,s) = t \qquad \text{score}(w,r)=1$$

$$\text{Jaccard}(r,s) = \frac{|r \cap s|}{|r \cup s|} \geq f$$

$$\rightarrow T(r,s) = \frac{|r|+|s|}{1+1/f}, \quad \text{score}(w,r)=1$$

$$\text{Cosine}(r,s) = \frac{\sum_w \text{TfIdf}(w,r).\text{TfIdf}(w,s)}{||r|| \; ||s||} \geq \theta$$

$$\rightarrow T(r,s) = \theta \qquad \text{score}(w,r)= \frac{\text{TfIdf}(w,r)}{||r||}$$

Approximating edit distance [GIJ+01]

- EditDistance(s,t) ≤ d → |q-grams(s) ∩ q-grams(t)| ≥ max(|s|,|t|) - (d-1)*q – 1
- Q-grams (sequence of q-characters in a field)
  - ‘AT&T Corporation’
  - 3-grams: {‘AT&’,’T&T’,’&T ‘, ‘T C’,’ Co’,’orp’,’rpo’,’por’,’ora’,’rat’,’ati’,’tio’,’ion’}
- Typically, q=3 ➔ Large q-gram sets
- Approximate large q-gram sets to smaller sets

Reducing size of large sets

- MinHash method
- Random projection method

Minhash method

- MinHash signature of sets
  - Choose a random permutation P of all tokens
  - MinHash(S={s_1,s_2,..,s_n}) = s_j if s_j is the first token in permutation P
- Jaccard(s,t) = Probability(Minhash(s) = Minhash(t))
  - Choose B such permutations to improve accuracy
- Extended to Jaccard on weighted terms (Charikar STOC 2002)
- Highly effective in practice
  - Mirror detection on webpages (Broder et al 1997)
  - Cleaning database records (Chaudhuri et al 2004)

Random Projection method

- Project each token to a random value in [-1 1]
- Random projection of set $S=\{s_1,s_2,..,s_n\}$
  - 1 if sum of projection of each token in S > 0
  - 0 otherwise
- Cosine(s,t) $\propto$ Probability(projection of s and t same)
  - Repeat for B projections to improve accuracy

Approximate join problem

- Find all pairs s,t where Token-overlap(s,t) > T
- Main idea: exploit threshold T to reduce work
- Two algorithms
  - Pair-Count
  - Indexed Count

Pair-Count

- Step 1: Pass over data, for each token create list of sets that contain it
- Step 2: generate pairs of sets, count and output those with count > T

The pair-counting table could be large → too memory intensive
Not good when list lengths highly skewed

(Broder et al WWW 1997)

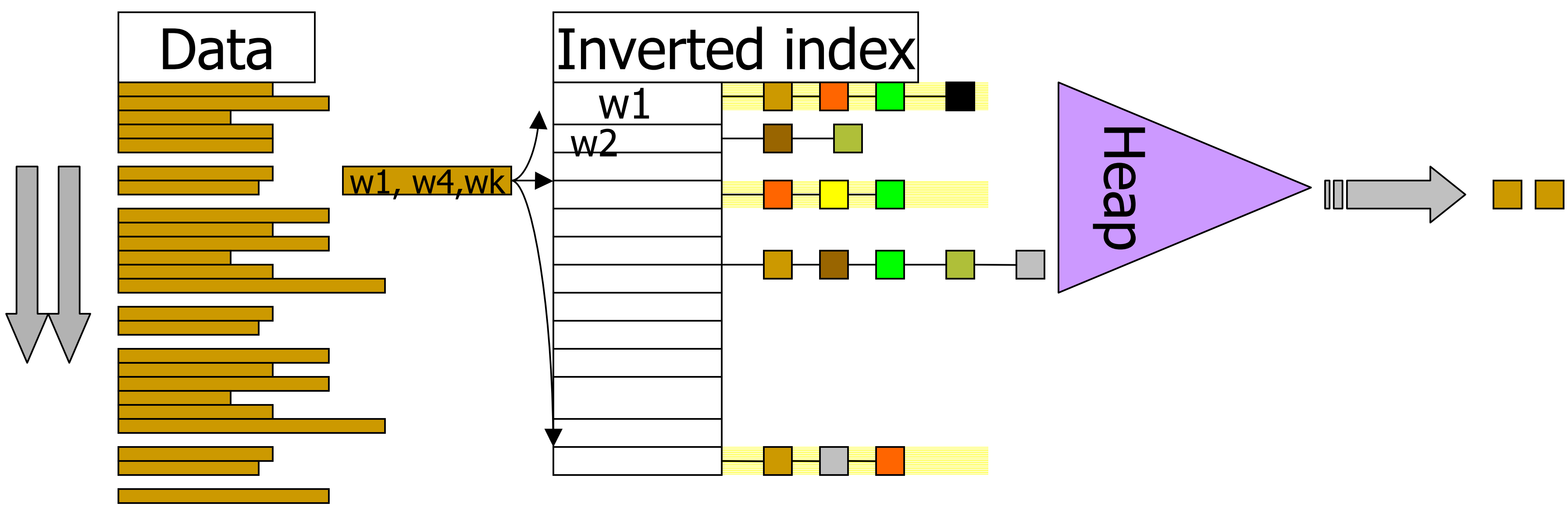
Probe-Count

Step 1: Create inverted index

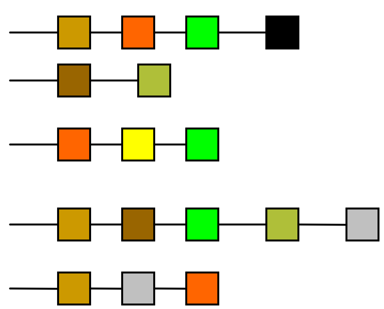
Step 2: Using each record,

probe

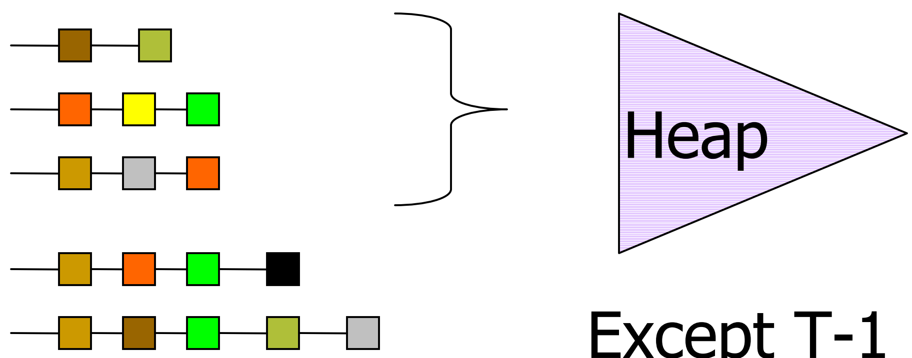
merge lists to find rids in T of them



Threshold sensitive list merge

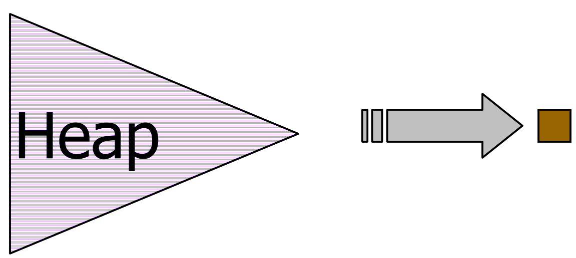


Lists to be merged

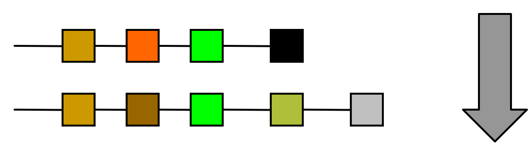


Sort by increasing size

Except T-1 largest, organize rest in heap (T=3)



Pop from heap successively



Search in large lists in increasing order
Use lower bounds to terminate early

(SK04, CGK06)

Summary of the pair creation step

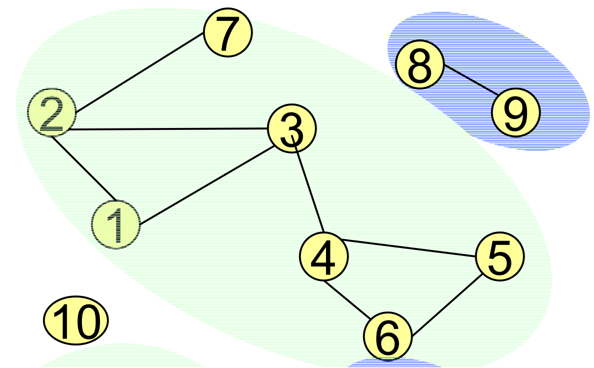
- Can be extended to the weighted case fitting the general framework.
- More complicated similarity functions use set similarity functions as filters
- Set sizes can be reduced through techniques like MinHash (weighted versions also exist)
  - Small sets (average set size < 20), most database entities with word tokens: use as-is
  - Large sets: web documents, sets of q-grams
    - Use Minhash or random projection

Integration: outline

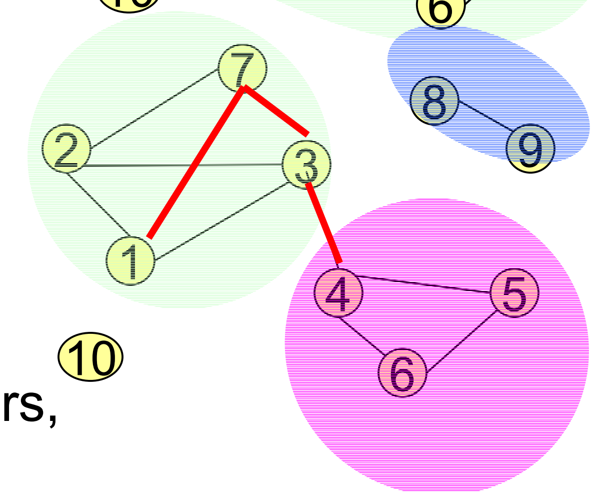
- Duplicate Entity elimination
  - Defining duplicates: string similarity functions
- Scalable algorithms for finding similar pairs
  - Join algorithms to find duplicate pairs
  - Efficient index design
- Create groups from duplicate entity pairs
  - Single entity: data partitions
  - Multiple entities: Collective multi-attribute deduplication

Creating partitions

- Transitive closure
  - Dangers: unrelated records collapsed into a single cluster



- Correlation clustering (Bansal et al 2002)
  - Partition to minimize total disagreements
    1. Edges across partitions
    2. Missing edges within partition
  - More appealing than clustering:
    - No magic constants: number of clusters, similarity thresholds, diameter, etc
  - Extends to real-valued scores
  - NP Hard: many approximate algorithms



3 disagreements

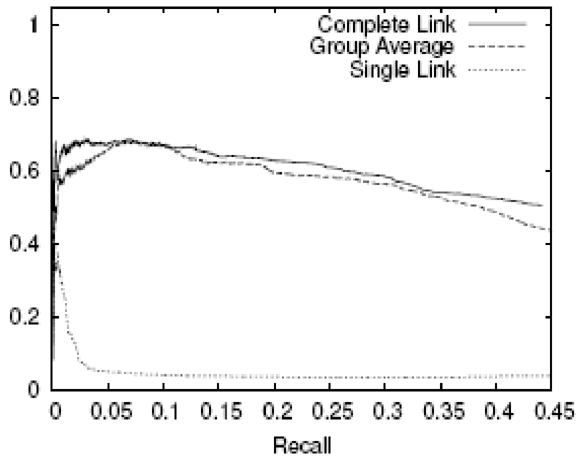
Algorithms for correlation clustering

- Integer programming formulation (Charikar 03)
  - X_{ij} = 1 if i and j in same partition, 0 otherwise

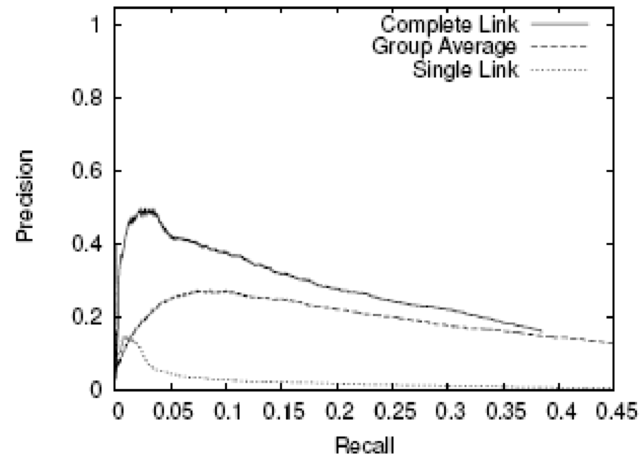
$$\min \sum_{ij \in edges} (1 - x_{ij}) + \sum_{ij \notin edges} x_{ij}$$
$$\text{such that} \quad x_{ij} \in \{0, 1\}$$
$$x_{ij} + x_{jk} \leq 1 + x_{ik} \qquad \text{(partitioning constraint)}$$

  - Impractical: $O(n^3)$ constraints
- Practical substitutes (Heuristics, no guarantees)
  - Agglomerative clustering: repeatedly merge closest clusters
    - Efficient implementation possible via heaps (BG 2005)
    - Definition of closeness subject to tuning
      - Greatest reduction in error
      - Average/Max/Min similarity

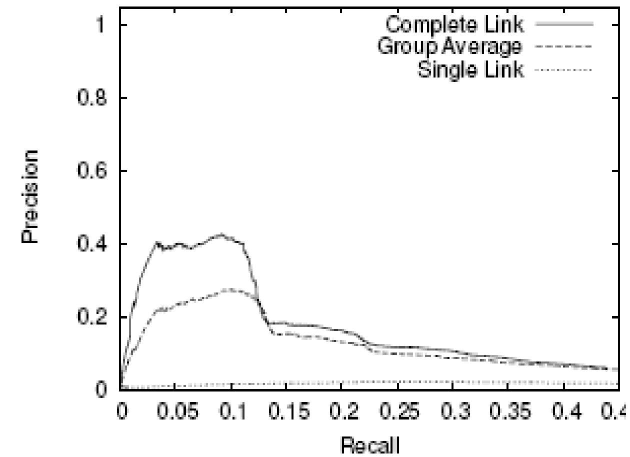
Empirical results on data partitioning



Digital cameras



Camcoder



Luggage

(From: Bilenko et al, 2005)

- Setup: Online comparison shopping,
  - Fields: name, model, description, price
  - Learner: Online perceptron learner
- Complete-link clustering >> single-link clustering(transitive closure)
- An issue: when to stop merging clusters

Other methods of partitioning

[Chaudhuri et al ICDE 2005]

- Partitions are compact and relatively far from other points
- A Partition has to satisfy a number of criteria
  - Points within partition closer than any points outside
  - #points within p-neighborhood of each partition < c
  - Either number of points in partition < K, or diameter < θ

Algorithm

- Consider case where partitions required to be of size < K ➔ if partition Pj of size m in output then
  - m-nearest neighbors of all r in Pi is Pi
  - Neighborhood of each point is sparse
- For each record, do efficient index probes to get
  - Get K nearest neighbors
  - Count of number of points in p-neighborhood for each m nearest neighbors
- Form pairs and perform grouping based on above insight to find groups

Summary: partitioning

- Transitive closure is a bad idea
- No verdict yet on best alternative
- Difficult to design an objective and algorithms
- Correlation clustering
  - Reasonable objective with a skewed scoring function
  - Poor algorithms
- Greedy agglomerative clustering algorithms ok
  - Greatest minimum similarity (complete-link), benefit
  - Reasonable performance with heap-based implementation
- Dense/Sparse partitioning
  - Positives: Declarative objective, efficient algorithm
  - Parameter retuning across domains
- Need comparison between complete-link, Dense/Sparse, and Correlation clustering.

Collective de-duplication: multi-attribute

Record	Title (a^1)	Author (a^2)	Venue (a^3)
b_1	"Record Linkage using CRFs"	"Linda Stewart"	"KDD-2003"
b_2	"Record Linkage using CRFs"	"Linda Stewart"	"9th SIGKDD"
b_3	"Learning Boolean Formulas"	"Bill Johnson"	"KDD-2003"
b_4	"Learning of Boolean Expressions"	"William Johnson"	"9th SIGKDD"

Associate variables for predictions

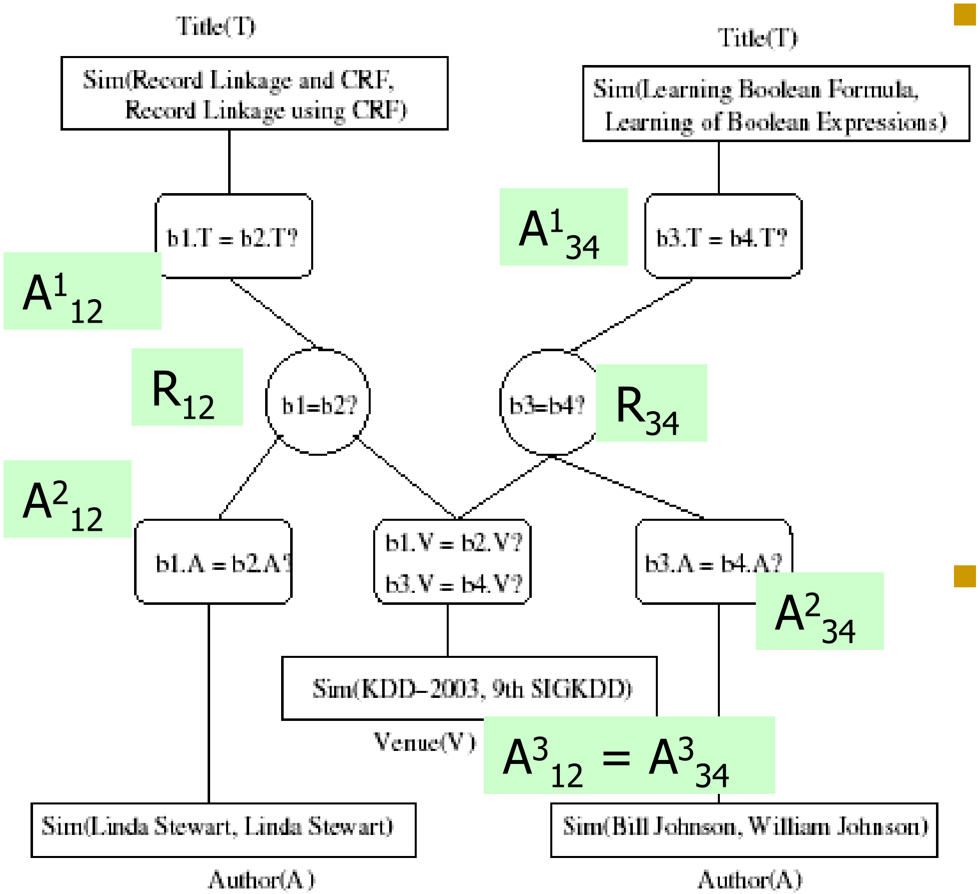
for each attribute k

each record pair (i,j) A^k_{ij}

for each record pair R_{ij}

from Parag & Domingos 2005

Dependency graph



Scoring functions

- Independent scores
  - $s_k(A^k,a_i,a_j)$ Attribute-level
    - Any classifier on various text similarities of attribute pairs
  - $s(R,b_i,b_j)$ Record-level
    - Any classifier on various similarities of all k attribute pairs
- Dependency scores
  - $d_k(A^k, R)$: record pair, attribute pair

	0	1
0	4	2
1	1	7

Joint de-duplication steps

- Jointly pick 0/1 labels for all record pairs R_{ij} and all K attribute pairs A^k_{ij} to maximize

$$\sum_{ij}[s(R_{ij})+\sum_{k}s_k(A^k_{ij})+d_k(R_{ij},A^k_{ij})]$$

- When dependency scores associative
  - $d_k(1,1) + d_k(0,0) >= d_k(1,0)+d_k(0,1)$
  - Can find optimal scores through graph MINCUT

Other issues and approaches

- Partitioning
  - Transitive-closure as a post processing
  - Results:

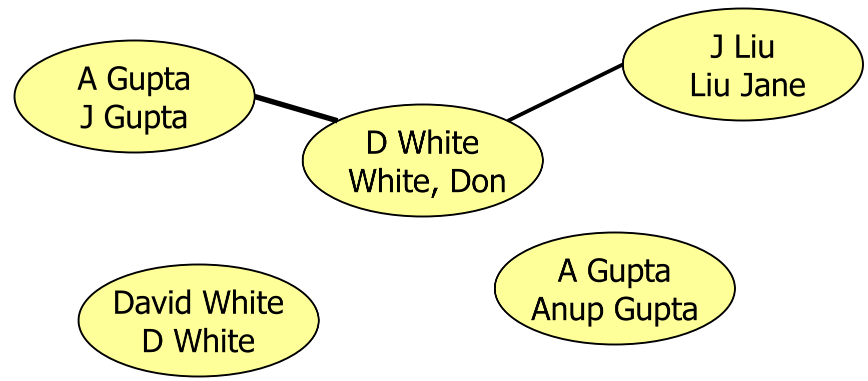
	Citation		Author		Venue	
	P	T	P	T	P	T
Independent	87	85	79	89	49	59
Collective	86	89	89	89	86	82

    - Collective deduplication
      - does not help whole citations,
      - helps attributes
    - Transitive closure can cause drop in accuracy
- Combined partitioning and linked dedup
  - Dong, HaLevy, Madhavan (SIGMOD 2005)
  - Bhattacharya and Getoor (2005)

Collective linkage: set-oriented data

(Bhattacharya and Getoor, 2005)

P1	D White, J Liu, A Gupta
P2	Liu, Jane & J Gupta & White, Don
P3	Anup Gupta
P4	David White



Scoring functions

- $S(A_{ij})$ Attribute-level
  - Text similarity
- $S(A_{ij}, N_{ij})$ Dependency with labels of co-author set
  - Fraction of co-author set assigned label 1.
- Score: $\alpha\, s(A_{ij}) + (1-\alpha)\, s(A_{ij}, N_{ij})$

Algorithm

Greedy agglomerative clustering

- Merge author clusters with highest score
- Redefine similarity between clusters of authors instead of single authors
  - Max of author-level similarity

Summary

- Scalable algorithms for finding pairs of duplicates solved for selective set similarity functions
  - Open problem: collection of weak similarity functions
- Data partitioning: agglomerative heap-based clustering algorithms ok.
  - Open problems:
    - scalable algorithms for correlation clustering
    - multi-attribute collective partitioning
    - Ambiguity resolution in networked entities
- Other issues: combined extraction and integration (Online integration)

R. Fagin and **J. Helpern**, **Belief, awareness, reasoning**. In **AI 1988** [10] also see

3 Top-level entities

Articles

Id	Title	Year	Journal	Canonical
2	Update Semantics	1983	10	

Journals

Id	Name	Canonical
10	ACM TODS	
17	AI	17
16	ACM Trans. Databases	

Writes

Article	Author
2	11
2	2
2	3

Authors

Id	Name	Canonical
11	M Y Vardi	
2	J. Ullman	4
3	Ron Fagin	3
4	Jeffrey Ullman	4

Variant links to canonical entries

Database: normalized, stores noisy variants

R. Fagin and **J. Helpern**, **Belief, awareness, reasoning**. In **AI 1988** [10] also see

Extraction

Author: R. Fagin A
Author: J. Helpern
Title Belief,..reasoning
Journal: AI
Year: 1998

Integration

Articles

Id	Title	Year	Journal	Canonical
2	Update Semantics	1983	10	
7	Belief, awareness, reasoning	1988	17	

Journals

Id	Name	Canonical
10	ACM TODS	
17	AI	17
16	ACM Trans. Databases	10

Writes

Article	Author
2	11
2	2
2	3
7	8
7	9

Authors

Id	Name	Canonical
11	M Y Vardi	
2	J. Ullman	4
3	Ron Fagin	3
4	Jeffrey Ullman	4
8	R Fagin	3
9	J Helpern	8

References

- [ACG02] Rohit Ananthakrishna, Surajit Chaudhuri, Venkatesh Ganti: Eliminating Fuzzy Duplicates in Data Warehouses. VLDB 2002: 586-597
- [BD83] Dina Bitton, David J. DeWitt: Duplicate Record Elimination in Large Data Files. ACM Trans. Database Syst. 8(2): 255-265 (1983)
- [BE77] Mike W. Blasgen, Kapali P. Eswaran: Storage and Access in Relational Data Bases. IBM Systems Journal 16(4): 362-377 (1977)
- [BG04] Indrajit Bhattacharya, Lise Getoor: Iterative record linkage for cleaning and integration. DMKD 2004: 11-18
- [C98] William W. Cohen: Integration of Heterogeneous Databases Without Common Domains Using Queries Based on Textual Similarity. SIGMOD Conference 1998: 201-212
- [C00] William W. Cohen: Data integration using similarity joins and a word-based information representation language. ACM Trans. Inf. Syst. 18(3): 288-321 (2000)
- [CCZ02] Peter Christen, Tim Churches, Xi Zhu: Probabilistic name and address cleaning and standardization. Australasian Data Mining Workshop 2002.
- [CGGM04] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, Rajeev Motwani: Robust and Efficient Fuzzy Match for Online Data Cleaning. SIGMOD Conference 2003: 313-324
- [CGG+05] Surajit Chaudhuri, Kris Ganjam, Venkatesh Ganti, Rahul Kapoor, Vivek R. Narasayya, Theo Vassilakis: Data cleaning in microsoft SQL server 2005. SIGMOD Conference 2005: 918-920
- [CGK06] Surajit Chaudhuri, Venkatesh Ganti, Raghav Kaushik: A primitive operator for similarity joins in data cleaning. ICDE 2006.
- [CGM05] Surajit Chaudhuri, Venkatesh Ganti, Rajeev Motwani: Robust Identification of Fuzzy Duplicates. ICDE 2005: 865-876
- [CRF03] William W. Cohen, Pradeep Ravikumar, Stephen E. Fienberg: A Comparison of String Distance Metrics for Name-Matching Tasks. IIWeb 2003: 73-78

References

- [DJ03] Tamraparni Dasu, Theodore Johnson: Exploratory Data Mining and Data Cleaning John Wiley 2003
- [DNS91] David J. DeWitt, Jeffrey F. Naughton, Donovan A. Schneider: An Evaluation of Non-Equijoin Algorithms. VLDB 1991: 443-452
- [DWI02] Data Warehousing Institute report 2002
- [E00] Larry English: Plain English on Data Quality: Information Quality Management: The Next Frontier. DM Review Magazine: April 2000. http://www.dmreview.com/article_sub.cfm?articleId=2073
- [FL95] Christos Faloutsos, King-Ip Lin: FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. SIGMOD Conference 1995: 163-174
- [FS69] I. Fellegi, A. Sunter: A theory of record linkage. Journal of the American Statistical Association, Vol 64. No 328, 1969
- [G98] D. Gusfield: Algorithms on strings, trees and sequences. Cambridge university press 1998
- [GFS+01] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, Cristian-Augustin Saita: Declarative Data Cleaning: Language, Model, and Algorithms. VLDB 2001: 371-380
- [GIJ+01] Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Divesh Srivastava: Approximate String Joins in a Database (Almost) for Free. VLDB 2001: 491-500
- [GIKS03] Luis Gravano, Panagiotis G. Ipeirotis, Nick Koudas, Divesh Srivastava: Text joins in an RDBMS for web data integration. WWW 2003: 90-101
- [GKMS04] S. Guha, N. Koudas, A. Marathe, D. Srivastava : Merging the results of approximate match operations. VLDB 2004.
- [GKR98] David Gibson, Jon M. Kleinberg, Prabhakar Raghavan: Clustering Categorical Data: An Approach Based on Dynamical Systems. VLDB 1998: 311-322

References

- [HS95] Mauricio A. Hernández, Salvatore J. Stolfo: The Merge/Purge Problem for Large Databases. SIGMOD Conference 1995: 127-138
- [HS98] Gísli R. Hjaltason, Hanan Samet: Incremental Distance Join Algorithms for Spatial Databases. SIGMOD Conference 1998: 237-248
- [J89] M. A. Jaro: Advances in record linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association 84: 414-420.
- [JLM03] Liang Jin, Chen Li, Sharad Mehrotra: Efficient Record Linkage in Large Data Sets. DASFAA 2003
- [JU91] Petteri Jokinen, Esko Ukkonen: Two Algorithms for Approximate String Matching in Static Texts. MFCS 1991: 240-248
- [KL51] S. Kullback, R. Liebler : On information and sufficiency. The annals of mathematical statistics 22(1): 79-86. 1959.
- [KMC05] Dmitri V. Kalashnikov, Sharad Mehrotra, Zhaoqi Chen: Exploiting Relationships for Domain-Independent Data Cleaning. SDM 2005
- [KMS04] Nick Koudas, Amit Marathe, Divesh Srivastava: Flexible String Matching Against Large Databases in Practice. VLDB 2004: 1078-1086
- [KMS05] Nick Koudas, Amit Marathe, Divesh Srivastava: SPIDER: flexible matching in databases. SIGMOD Conference 2005: 876-878
- [LLL00] Mong-Li Lee, Tok Wang Ling, Wai Lup Low: IntelliClean: a knowledge-based intelligent data cleaner. KDD 2000: 290-294
- [ME96] Alvaro E. Monge, Charles Elkan: The Field Matching Problem: Algorithms and Applications. KDD 1996: 267-270

References

- [ME97] Alvaro E. Monge, Charles Elkan: An Efficient Domain-Independent Algorithm for Detecting Approximately Duplicate Database Records. DMKD 1997
- [RY97] E. Ristad, P. Yianilos : Learning string edit distance. IEEE Pattern analysis and machine intelligence 1998.
- [S83] Gerry Salton : Introduction to modern information retrieval. McGraw Hill 1987.
- [SK04] Sunita Sarawagi, Alok Kirpal: Efficient set joins on similarity predicates. SIGMOD Conference 2004: 743-754
- [TF95] Howard R. Turtle, James Flood: Query Evaluation: Strategies and Optimizations. Inf. Process. Manage. 31(6): 831-850 (1995)
- [TKF01] S. Tejada, C. Knoblock, S. Minton : Learning object identification rules for information integration. Information Systems, Vol 26, No 8, 607-633, 2001.
- [W94] William E. Winkler: Advanced methods for record linkage. Proceedings of the section on survey research methods, American Statistical Association 1994: 467-472
- [W99] William E. Winkler: The state of record linkage and current research problems. IRS publication R99/04 (http://www.census.gov/srd/www/byname.html)
- [Y02] William E. Yancey: BigMatch: A program for extracting probable matches from a large file for record linkage. RRC 2002-01. Statistical Research Division, U.S. Bureau of the Census.