Google Search Timeline
Structured Querying of Text Databases

Information Extraction, or Deriving Structured Information from Text Documents

- Natural-language text embeds “structured” data
- Properly trained extraction systems extract this data
- Extraction output can be regarded as “relations”
  - one attribute (entity extraction)
  - several attributes (relation extraction)

<table>
<thead>
<tr>
<th>Disease Outbreaks</th>
<th>Proteus [Grishman et al., HLT 2000]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disease</td>
<td>Country</td>
</tr>
<tr>
<td>Cholera</td>
<td>Angola</td>
</tr>
</tbody>
</table>

Significant progress over the last decade+ [MUC]
Information Extraction has Many Applications

- Over a corporation’s customer report or email complaint database: enabling sophisticated querying and analysis
- Over biomedical literature: identifying drug/condition interactions
- Over newspaper archives: tracking disease outbreaks, terrorist attacks; intelligence

Thomson Reuters’s OpenCalais
www.opencalais.com

A Dutch court ruled against navigation systems company TomTom on Thursday in a patent infringement lawsuit against rival navigation device maker IBM from the United States.
**Snowball: An Easily Trained Information Extraction System**

- Requires only minimal user input (just a handful of example tuples)
- Works for simple binary relations (i.e., for simple relations with two attributes)

---

**Example Relation: Headquarters(Organization, Location)**

<table>
<thead>
<tr>
<th>Organization</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>Redmond</td>
</tr>
<tr>
<td>Apple Computer</td>
<td>Cupertino</td>
</tr>
<tr>
<td>Nike</td>
<td>Portland</td>
</tr>
</tbody>
</table>

Microsoft’s central headquarters in **Redmond** is home to almost every product group and division.

Brent Barlow, 27, a software analyst and beta-tester at **Apple Computer** headquarters in **Cupertino**, was fired Monday for "thinking a little too different."

*Apple*’s programmers "think different" on a "campus" in **Cupertino, Calif**. *Nike* employees "just do it" at what the company refers to as its "World Campus," near **Portland, Ore.**

Note redundancy of information
**Snowball: Occurrences of Seed Tuples**

Find occurrences of seed tuples:

- Computer servers at **Microsoft**’s headquarters in **Redmond**.
- In mid-afternoon trading, share of **Redmond**-based **Microsoft** fell.
- The **Armonk**-based **IBM** introduced a new line.
- The combined company will operate from **Boeing**’s headquarters in **Seattle**.
- **Intel, Santa Clara**, announced a new P4...

**Initial Seed Tuples** ➔ **Occurrences of Seed Tuples** ➔ **Generate New Seed Tuples** ➔ **Tag Entities** ➔ **Augment Table** ➔ **Generate Extraction Patterns**

---

**Snowball: Tagging Entities**

Use MITRE’s Alembic Named Entity tagger

- Computer servers at **Microsoft**’s headquarters in **Redmond**.
- In mid-afternoon trading, share of **Redmond**-based **Microsoft** fell.
- The **Armonk**-based **IBM** introduced a new line.
- The combined company will operate from **Boeing**’s headquarters in **Seattle**.
- **Intel, Santa Clara**, announced a new P4...

**Initial Seed Tuples** ➔ **Occurrences of Seed Tuples** ➔ **Generate New Seed Tuples** ➔ **Tag Entities** ➔ **Augment Table** ➔ **Generate Extraction Patterns**
Snowball: Pattern Representation

A Snowball pattern is a 5-tuple \(<\text{left}, \text{tag}_1, \text{middle}, \text{tag}_2, \text{right}>\),

- \text{tag}_1, \text{tag}_2\) are named-entity tags
- \text{left}, \text{middle}, and \text{right}\) are vectors of weighed terms.

Snowball Pattern Generation:
Cluster Similar Occurrences

- \{<\text{servers} 0.75>, <\text{at} 0.75>\}
- \{<\text{shares} 0.75>, <\text{of} 0.75>\}
- \{<\text{the} 1>\}
- \{<\text{operate} 0.75>, <\text{from} 0.75>\}
**Snowball: Tuple Extraction**

Using the patterns, scan the collection to generate new seed tuples:

- **Initial Seed Tuples**
- **Occurrences of Seed Tuples**
- **Generate New Seed Tuples**
- **Tag Entities**
- **Augment Table**
- **Generate Extraction Patterns**

**Snowball: Automatic Pattern Evaluation**

Pattern “ORGANIZATION, LOCATION” in action:

<table>
<thead>
<tr>
<th>Boeing, Seattle, said.</th>
<th>Positive</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel, Santa Clara, cut prices...</td>
<td>Positive</td>
</tr>
<tr>
<td>invest in Microsoft, New York-based analyst Jane Smith said</td>
<td>Negative</td>
</tr>
</tbody>
</table>

Automatically estimate probability of a pattern generating valid tuples:

\[
\text{Conf(Pattern)} = \frac{\text{Positive}}{\text{Positive} + \text{Negative}}
\]

E.g., \(\text{Conf(Pattern)} = \frac{2}{3} = 66\%\)
**Snowball: Automatic Tuple Evaluation**

Conf(Tuple) = 1 - Π(1 - Conf(Pi))

- Estimation of Probability(Correct(Tuple))
- A tuple will have high confidence if generated by multiple high-confidence patterns (Pi).

Brent Barlow, 27, a software analyst and beta-tester at Apple Computer headquarters in Cupertino, was fired Monday for "thinking a little too different."

Apple's programmers "think different" on a "campus" in Cupertino, Cal.

**Snowball: Filtering Seed Tuples**

<table>
<thead>
<tr>
<th>ORGANIZATION</th>
<th>LOCATION</th>
<th>CONF</th>
</tr>
</thead>
<tbody>
<tr>
<td>AG EDWARDS</td>
<td>STL</td>
<td>0.93</td>
</tr>
<tr>
<td>AIR CANADA</td>
<td>MONTREAL</td>
<td>0.89</td>
</tr>
<tr>
<td>7TH LEVEL</td>
<td>RICHARDSON</td>
<td>0.88</td>
</tr>
<tr>
<td>3COM CORP</td>
<td>SANTA CLARA</td>
<td>0.8</td>
</tr>
<tr>
<td>3DO</td>
<td>REDWOOD CITY</td>
<td>0.8</td>
</tr>
<tr>
<td>3M</td>
<td>MINNEAPOLIS</td>
<td>0.8</td>
</tr>
<tr>
<td>MACWORLD</td>
<td>SAN FRANCISCO</td>
<td>0.7</td>
</tr>
<tr>
<td>157TH STREET</td>
<td>MANHATTAN</td>
<td>0.52</td>
</tr>
<tr>
<td>15TH CENTURY EUROPE</td>
<td>NAPOLEON</td>
<td>0.3</td>
</tr>
<tr>
<td>15TH PARTY CONGRESS</td>
<td>CHINA</td>
<td>0.3</td>
</tr>
<tr>
<td>MAD</td>
<td>SMITH</td>
<td>0.3</td>
</tr>
</tbody>
</table>

Generate new seed tuples:
**Snowball: An Easily Trained Information Extraction System**

- Requires only minimal user input (just a handful of example tuples)
- Works for simple binary relations (i.e., for simple relations with two attributes)

---

**“SQL” Queries Over Text Databases**

- `SELECT Company, CEO FROM Headquarters H, Executives E WHERE H.Company = E.Company and H.Location = 'Redmond'`
- `SELECT * FROM DiseaseOutbreaks WHERE Country = 'Sudan'`
- `SELECT * FROM DiseaseOutbreaks WHERE Country = 'Angola'`

---

BBC: May, 2006

A **Cholera** epidemic in **Angola** has now killed more than **200** people in the past three months …

Carbide Group Inc., of Pittsburgh won a **$627,068** contract from the Army Research Lab in Adelphi for research and development.

Mr. Song-nam conveyed greetings and best wishes of the CEO of Tokyo-based Mitsubishi, Mr. Ichiro Taniguchi.

---

Mr. Bond's contract and the wishes of the CEO of Tokyo-based Mitsubishi, Mr. Ichiro Taniguchi.
Processing a SQL Query over a Text Database: A Naïve Execution

- For every database document:
  - Retrieve document
  - Feed document to all relevant information extraction systems
  - Add extracted tuples to corresponding “relations”
- Evaluate SQL query over extracted relations

Any problems?

Information Extraction is Expensive!

Complex text analysis often required:
- parsing
- named-entity tagging
- cleaning
- ...

Top-\(k\) Query Variants to the Rescue

- Often “complete” query results are not necessary
- Top-\(k\) query results are preferable for efficiency (but wait!)
- Appropriate value of \(k\) is user-specific
  - Some users after relatively comprehensive results
  - Other users after “quick-and-dirty” results

Processing Top-\(k\) Query Variants over Text Databases: Outline

- Variants of top-\(k\) query model
  - “Any-\(k\)” query results

- Query optimization approach for each model
Executing an “Any-k” Query over a Text Database

1. Retrieve documents from database
2. Process documents with information extraction system
3. Extract relation tuples
4. Evaluate SQL query over extracted relation

Goal: Return \( k \) extracted results as fast as possible:

- Information extraction output is trusted
- All extracted tuples are considered correct
- “Any-\( k \)” query results

Properties of Alternative Execution Plans

- Execution time

- Recall: fraction of tuples that a plan manages to extract
  Or similarly, absolute number of tuples that a plan manages to extract

Execution plans vary in their document retrieval strategy

Fastest plan that produces \( k \) tuples for an any-\( k \) query?
Document Retrieval for Query Processing

Similar to relational world
- Two major execution paradigms:
  - Scan-based: Retrieve documents sequentially (as in Naïve execution)
  - Index-based: Retrieve documents via keyword queries (e.g., [case fatality rate])
- Underlying data distribution dictates what is best

Unlike relational world
- Indexes are only "approximate": index is on keywords, not on tuples of interest
- Choice of execution plan affects output completeness (not only speed)

Query Plans for Any-\(k\) Queries: Outline

- Scan- and index-based document retrieval strategies:
  - Scan
  - Filtered Scan
  - Iterative Set Expansion
  - Automatic Query Generation
- Index-based

- Analysis: execution time and number of extracted tuples
**Scan for Any-k Queries**

1. Retrieve documents from database
2. Process documents with information extraction system
3. Extract relation tuples

*Scan* retrieves and processes documents sequentially, until *k* tuples extracted.

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

*Key question*: How many documents does *Scan* retrieve to extract *k* tuples?

**Filtered Scan for Any-k Queries**

1. Retrieve documents from database
2. Filter documents
3. Process filtered documents
4. Extract relation tuples

*Filtered Scan* retrieves and processes documents sequentially, until *k* tuples extracted.

*Filtered Scan* uses a classifier to process only promising documents (e.g., a Sports article is unlikely to describe disease outbreaks).

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

*Key question*: How many documents does *Filtered Scan* retrieve to extract *k* tuples?
Query Plans for Any-k Queries: Outline

- Scan- and index-based document retrieval strategies:
  - Scan
  - Filtered Scan
  - Iterative Set Expansion
  - Automatic Query Generation

- Analysis: execution time and number of extracted tuples

Iterative Set Expansion for Any-k Queries

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

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

Key questions: How many queries and how many documents does Iterative Set Expansion need to extract k tuples?

Time for retrieving a document
Time for processing a document
Time for answering a query
Understanding *Iterative Set Expansion*: Querying Graph

- Querying graph is bipartite, with tuples and documents
- Each tuple (transformed into a keyword query) retrieves documents
- Documents contain tuples

Reachability Graph: Can $k$ Tuples be Extracted?

Upper limit on number of extracted tuples, if starting with one seed tuple: size of biggest connected component
**Automatic Query Generation (“PromD”) for Any-k Queries**

- **Iterative Set Expansion** might not fully answer an any-k query due to iterative nature of query generation

- **Automatic Query Generation** avoids this problem by learning queries offline, designed to return documents with tuples

---

**Automatic Query Generation for Any-k Queries**

1. Offline Query Generation
   - Learn keyword queries to retrieve documents with tuples

2. Text Database
   - Query database with learned queries
     (e.g., [case fatality rate])

3. Information Extraction
   - Process retrieved documents

4. Relation
   - Extract tuples from documents

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

**Key questions:** How many queries and how many documents does **Automatic Query Generation** need to extract k tuples?
Query Plans for Any-k Queries: Outline

- Scan- and index-based document retrieval strategies:
  - Scan
  - Filtered Scan
  - Iterative Set Expansion
  - Automatic Query Generation

- Analysis: execution time and number of extracted tuples
  - Time and number of extracted tuples depend on database properties:
    - Tuple “degree” distribution
    - Document “degree” distribution (e.g., in querying graph)
  - Database properties computed “on-the-fly” as documents are processed
    - Assume knowledge of data distributions based on known properties of text
    - Estimate parameters of distributions during query execution

Executing an “Any-k” Query over a Text Database

1. Retrieve documents from database
2. Process documents with information extraction system
3. Extract relation tuples
4. Evaluate SQL query over extracted relation

Goal: Return $k$ extracted results as fast as possible:

- Information extraction output is trusted
- All extracted tuples are considered correct
- “Any-k” query results [Ipeirotis et al., SIGMOD 2006, TODS 2007]
Information Extraction is Error-Prone

- **Recall**: fraction of correct tuples that are extracted
  Extraction systems may miss valid tuples

- **Precision**: fraction of extracted tuples that are correct
  Extraction systems may generate erroneous tuples
Processing Top-k Query Variants over Text Databases: Outline

- Variants of top-k query model
  - “Any-k” query results
  - “Any-k correct” query results

- Query optimization approach for each model

---

Executing an “Any-k Correct” Query over a Text Database

1. Select information extraction system for each relation in query
2. Select document retrieval strategy for each selected extraction system
3. Retrieve documents using selected retrieval strategies
4. Process documents using selected extraction systems
5. Clean and normalize extracted relations
6. Return query results

- **Goal:** Return *k correct* extracted results as fast as possible:

- Information extraction output is not trusted
- Extracted tuples are estimated to include incorrect tuples
- But no indication of tuple correctness is returned in results
- “Any-k correct” query results

[Jain et al., ICDE 2008 & 2009]
Document Retrieval Strategies for Any-\(k\) Correct Queries: Outline

- Scan- and index-based document retrieval strategies:
  - Scan
  - Filtered Scan
  - Iterative Set Expansion
  - Automatic Query Generation (“PromD”)
  - “Const”
  - “PromC” (PromD + Const)

- Analysis: execution time and number of correct extracted tuples

[Jain et al., ICDE 2008 & 2009]

---

**Const for Any-\(k\) Correct Tuples**

- Exploits selection “constants” in SQL query
  - Documents without them cannot contribute tuples for corresponding relations
- Avoids processing all documents, conservatively
- Still potentially retrieves useless documents

Keyword query: [Sudan]

```
SELECT *
FROM DiseaseOutbreaks
WHERE Country = 'Sudan'
```

- Documents that generate tuples
- Documents that do not generate tuples
- Documents that contain SQL query constants
**PromC for Any-\(k\) Correct Tuples**

- Exploits selection “constants” in SQL query
- Avoids processing all documents, aggressively
- May miss useful documents

\[
\text{[case fatality rate]} \text{ from PromD} + \text{[Sudan]} \text{ from Const:}
\]

Keyword query: [case fatality rate Sudan]

![Text Database Diagram](image)

- Documents that generate tuples
- Documents that do not generate tuples
- Documents that contain SQL query constants

**An Execution of an “Any-\(k\) Correct” Query**

**Headquarters** (Company, Location)  
**Executives** (Company, CEO)  

**SELECT** H.Company, E.CEO  
**FROM** Headquarters H, Executives E  
**WHERE** H.Company = E.Company  
**AND** H.Location = ‘Redmond’

![Query Execution Diagram](image)
Optimizing an “Any-\(k\) Correct” Query

1. Select information extraction system for each relation in query
2. Select document retrieval strategy for each selected extraction system
3. Retrieve documents using selected retrieval strategies
4. Process documents using selected extraction systems
5. Clean and normalize extracted relations
6. Return query results

Properties of Alternative Execution Plans

- Execution time
- Number of correct tuples that a plan manages to extract

Execution plans vary in their extraction systems and document retrieval strategies

Fastest plan that produces \(k\) correct tuples for an any-\(k\) correct query?
Executing an “Any-\(k\) Correct” Query over a Text Database

1. Select information extraction system for each relation in query
2. Select document retrieval strategy for each selected extraction system
3. Retrieve documents using selected retrieval strategies
4. Process documents using selected extraction systems
5. Clean and normalize extracted relations
6. Return query results

- **Goal:** Return \(k\) correct extracted results as fast as possible:
  - Information extraction output is not trusted
  - Extracted tuples are estimated to include incorrect tuples
  - But no indication of tuple correctness is returned in results
  - “Any-\(k\) correct” query results

---

Any-\(k\) (Correct) Query Results
Do Not Report Tuple “Correctness”

<table>
<thead>
<tr>
<th>Company</th>
<th>Location</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delta Airlines</td>
<td>Atlanta</td>
<td>0.97</td>
</tr>
<tr>
<td>Lighthearted Pocahontas-Lion King</td>
<td>Central Park</td>
<td>0.89</td>
</tr>
<tr>
<td>Vocaltec communications Ltd.</td>
<td>Redmond</td>
<td>0.87</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>Mountain View</td>
<td>0.75</td>
</tr>
<tr>
<td>Vocaltec communications Ltd.</td>
<td>Tel-Aviv</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Scoring function for top-\(k\) queries should reflect confidence in tuple correctness

[Answers from New York Times archive, extracted by Snowball information extraction system]
Processing Top-k Query Variants over Text Databases: Outline

- Variants of top-k query model
  - “Any-k” query results
  - “Any-k correct” query results
  - “Top-k” query results
    Scoring function based on expected tuple correctness
- Query optimization approach for each model

Modeling Expected Tuple Correctness

<table>
<thead>
<tr>
<th>Headquarters</th>
<th>Company</th>
<th>Location</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>Redmond</td>
<td></td>
<td>0.97</td>
</tr>
<tr>
<td>Delta Airlines</td>
<td>Atlanta</td>
<td></td>
<td>0.89</td>
</tr>
<tr>
<td>Vocalcom Ltd.</td>
<td>Tel Aviv</td>
<td></td>
<td>0.85</td>
</tr>
<tr>
<td>Sun Microsystems Inc.</td>
<td>Mountain View</td>
<td></td>
<td>0.73</td>
</tr>
<tr>
<td>Lighthearted Phoenix Ltd.</td>
<td>Central Park</td>
<td></td>
<td>0.2</td>
</tr>
<tr>
<td>NEC Corp.</td>
<td>Indonesia</td>
<td></td>
<td>0.01</td>
</tr>
</tbody>
</table>

- One approach: use “confidence” scores as returned by extraction systems
  ... handle within probabilistic databases
  (e.g., [Gupta and Sarawagi, VLDB 2006], [Cafarella et al., CIDR 2007], [Ilyas et al., 2008])
- Alternative approach: do some “external” computation
Modeling Expected Tuple Correctness

- Redundancy in text databases is key to counter noise and extraction errors
- Intuitively, the more times a tuple is extracted (independently), the higher its expected correctness
  - Snowball [Agichtein and Gravano, DL 2000]
  - URNS model [Downey et al., IJCAI 2005]
  - REALM model [Downey et al., ACL 2007], for “sparse” data
  - ...

Executing a “Top-k” Query over a Text Database

1. Retrieve documents from database
2. Process documents with information extraction system
3. Extract relation tuples
4. Evaluate SQL query over extracted relation, returning top-k results

Goal: Return top-k extracted results as fast as possible:

- Information extraction output is not trusted
- Extracted tuples are ranked according to expected correctness
- Execution should avoid processing all documents, for efficiency
- “Top-k” query results
Top-k Query Processing in this Scenario Has Not Been Solved...

- Recall “on-the-fly” extraction goal, need to avoid processing all documents, for efficiency
- Can we apply well-known top-k query processing “tricks” and algorithms?
  - Problem is lack of “sorted access”
    - No efficient access to tuples in sorted order of expected correctness
  - But interesting approximations, perhaps with probabilistic guarantees, may be possible...
    - [Jain and Srivastava 2009]

Processing Top-k Query Variants over Text Databases: Summary

- We discussed how to process top-k query variants over text databases: “any-k,” “any-k correct,” and “top-k” queries
- We focused on “on-the-fly” information extraction
  - Information extraction task short-lived, new
  - Text database previously unseen or too vast
  - So exhaustive extraction approaches may be impractical
- But intriguing connection with “open information extraction” [Banko et al., IJCAI 2007] still unexplored
- We focused on “blackbox” information extraction systems
- But information extraction system “knobs” (e.g., for tuning precision, recall) can play important role in query optimization
  - [Jain and Ipeirotis 2009]

Exciting research area, with many challenging open problems!