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)

BBC: May, 2006
A Cholera epidemic in Angola has now killed more than 1,200 people in the past three months …

<table>
<thead>
<tr>
<th>Disease Outbreaks</th>
<th>Disease</th>
<th>Country</th>
<th>Deaths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cholera</td>
<td>Angola</td>
<td></td>
<td>1,200</td>
</tr>
</tbody>
</table>
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

Information Extraction Example: Organizations’ Headquarters

Input: Documents

Named-Entity Tagging

Pattern Matching

Output: Tuples

<table>
<thead>
<tr>
<th>Id</th>
<th>Organization</th>
<th>Location</th>
<th>W</th>
<th>Useful</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Eastman Kodak</td>
<td>Rochester</td>
<td>0.9</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Apple Computer</td>
<td>Cupertino</td>
<td>0.8</td>
<td></td>
</tr>
</tbody>
</table>
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 for Simple IE Tasks**

- 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>

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.

**Snowball: Tagging Entities**

Use 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.
**Snowball: Pattern Representation**

A *Snowball* pattern is a 5-tuple `<left, tag1, middle, tag2, right>`,
- *tag1, tag2* are named-entity tags
- *left, middle, and right* are vectors of weighed terms.

```
ORGANIZATION's central headquarters in LOCATION is home to...
```

```
ORGANIZATION <s 0.5>, <central 0.5> <headquarters 0.5>, <in 0.5>

LOCATION <is 0.75>, <home 0.75>
```

---

**Snowball Pattern Generation:**

*Cluster Similar Occurrences*

```
<servers 0.75> <at 0.75>

LOCATION <of 0.75>

ORGANIZATION <based 0.75>

LOCATION <fell 1>

LOCATION <the 1>

LOCATION <introduced 1.75> <a 0.75>

ORGANIZATION <operate 0.75> <from 0.75>

ORGANIZATION <headquarters 0.7> <in 0.7>

LOCATION
```

Eugene Agichtein’s slide
**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:

- Boeing, Seattle, said... **Positive**
- Intel, Santa Clara, cut prices... **Positive**
- invest in Microsoft, New York-based analyst Jane Smith said **Negative**

Pattern Confidence:

Automatically estimate probability of a pattern generating valid tuples:

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

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

Conf(Tuple) = 1 - Π(1 - Conf(P_i))

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

---

**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>ST LUIS</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:

Initial Seed Tuples

Occurrences of Seed Tuples

Generate New Seed Tuples

Tag Entities

Augment Table

Generate Extraction Patterns
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)

IE for Complex Relations is Challenging and Time-Consuming

- Operates over large sets of features
  - Bag of words, N-grams, grammar productions, dependency paths

- Requires complex text analysis
  - Dependency parsing, entity recognition, syntactic parsing, shallow parsing, part-of-speech tagging, semantic role labeling

May grow as large as number of unique words and sequences of N words

May take several seconds per document
Problematic over large document collections
“SQL” Queries Over Text Databases

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

BBC: May, 2006
A Cholera epidemic in Angola has now killed more than 1,200 people in the past three months...

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

Text Database

Mr. Song-nam conveyed greetings and best 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?
Reducing Processing Time: Opportunities

- Documents are “useful” if they produce output for a given IE task
- Small, topic-specific fraction of collection is useful
  - Only 2% of documents in a New York Times archive, mostly environment-related, are useful for Natural Disaster-Location with a state-of-the-art IE system
  - Useful documents share distinctive words and phrases
    - “Earthquake,” “storm,” “Richter,” “volcano eruption” for Natural Disaster-Location
  - Information extraction system “labels” documents as useful or not for free as documents are processed

Efficient Processing of a SQL Query over a Text Database: Top-\(k\) Query Processing

- Often “complete” query results are not necessary
- Top-\(k\) query results are preferable for efficiency
- Appropriate value of \(k\) is user-specific
  - Some users after relatively comprehensive results
  - Other users after “quick-and-dirty” results
Processing a SQL Query over a Text Database: Outline

- One variant 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

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

**Text Database**

**Information Extraction**

**Relation**

**Similar to relational world**

- **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

- 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** = $|\text{Retrieved Docs}| \cdot (R + P)$

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

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 + σP)

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| . (R + P) + |Queries| . Q

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

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?**

Tuples    Documents

$t_1$    $d_1$
$t_2$    $d_2$
$t_3$    $d_3$
$t_4$    $d_4$
$t_5$    $d_5$

Reachability Graph:

$t_1$ retrieves document $d_1$, which contains $t_2$

Upper limit on number of extracted tuples, if starting with one seed tuple: size of biggest connected component

---

**Automatic Query Generation 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

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

**Text Database**

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

**Information Extraction**
3. Process retrieved documents

**Relation**
4. Extract tuples from documents

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

**Key questions:** How many queries and how many documents does **Automatic Query Generation** need to extract k tuples?

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

**Text Database**

1. Retrieve documents from database

**Information Extraction**
2. Process documents with information extraction system

**Relation**
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
Full Execution of an “Any-k” Query over a Text Database

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

---

Full Execution of an “Any-k” Query

**SELECT** H.Company, E.CEO  
**FROM** Headquarters H, Executives E  
**WHERE** H.Company = E.Company  
**AND** H.Location = ‘Redmond’
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
Alternatives to “Any-k” Query Model to Account for Output Correctness

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

- “Any-k” query results
- “Any-k correct” query results (not covered in course)
- “Top-k” query results (not covered in course)

Top-k scoring function based on expected tuple correctness
Optional Readings from our Research Group

- ... and many others, see [http://extraction.cs.columbia.edu/](http://extraction.cs.columbia.edu/)

Exciting research area, with many challenging open problems!