OPEN INFORMATION EXTRACTIVE FROM THE WEB

Michele Banko, Michael J Cafarella, Stephen Soderland, Matt Broadhead and Oren Etzioni
Call for a Shake Up in Search!

- Question Answering rather than indexed keyword search
- Gravity of keyword search
- Massive, heterogeneous data
- Knowledge assertion
- Call for a general-purpose question-answering systems
Watson, Siri
Motivation

- Traditional Information Extraction (IE)
  - Require hand-crafted extraction rule, training example
  - Re-specify relation of interest
  - Usually domain specific

- Dose not scale well with large and heterogeneous corpora
About this paper

- High level description on system components
- Framework design
- Technical details largely based on description rather than rigorous details
  - Work on Maximum Entropy Methods (part-of-speech labeling, identifying noun phrases...)
  - Work on KnowItAll paper
Several terminologies

- **Tuple**: \((e_i, r_{ij}, e_j)\), \(r_{ij}\) is relation

- **Relation**: general rules for connecting entities, e.g. City *such as* New York, Tokyo, London, Beijing...

- **Relation arguments**: for tuple \((e_i, r_{ij}, e_j)\), \(e_i\) and \(e_j\) are arguments for relation \(r_{ij}\)
Design Goals

- Automation
- Corpus heterogeneity
- Efficiency
Key components:
- Self-supervised learner
- Single-pass extractor
- Redundancy-based extractor
- Query Processing
Key components:

- Self-supervised learner
- Single-pass extractor
- Redundancy-based extractor
- Query Processing
Self-supervised Learner

- Step 1: Label training data as positive or negative (using parser to train extractor)
- Step 2: Use labeled data (extract features) to train a Naïve Bayes classifier
Self-supervised Learner

- Step 1: Label training data as positive or negative (using parser to train extractor)
- Step 2: Use labeled data (extract features) to train a Naïve Bayes classifier
Self-supervised Learner

1.1 Trainer parses through text. For each sentence, find all base noun phrase $e_i$, for each pair $(e_i, e_j)$, identify potential relation $r_{ij}$ (sequence of words) in tuple $t = (e_i, r_{ij}, e_j)$

1.2 Using constrains to label $t$ as positive or negative
   - Length of dependency chain connecting $(e_i, e_j)$
   - Path from $(e_i, e_j)$ does not cross sentence boundary
Self-supervised Learner

- Step 1: Label training data as positive or negative (using parser to train extractor)
- Step 2: Use labeled data (extract features) to train a Naïve Bayes classifier
Self-supervised Learner

- 2.1 Map each tuple to a feature vector
  - E.g. number of tokens in $r_{ij}$, presence of POS tag sequence in $r_{ij}$, POS tag to the left of $e_i$

- 2.2 Labeled feature vectors are as input to a Naïve Bayes classifier

- Classifier is language specific
Key components:
- Self-supervised learner
- Single-pass extractor
- Redundancy-based extractor
- Query Processing
Single-pass Extractor

- Make a single pass over corpus
- Tag POS label for each word in sentence
- Using tags and nous phrase chunker to identify entities
- Relations are extracted by analyzing text between noun phrases
- Classifier classifies Candidate tuples. TextRunner Stores the trustworthy tuples
Single-pass Extractor

- Relation Normalization: Non essential phrases are eliminated to have succinct relation text (e.g. “definitely developed” is reduced to “developed”)

- Entity Normalization: Chunker assigns probability to entities. Tuples containing entities with low confidence are dropped.
Key components:
- Self-supervised learner
- Single-pass extractor
- Redundancy-based extractor
- Query Processing
Redundancy-based Assessor

- Merge identical tuples
- Count distinct sentences
- The count is used to assign probability to each tuple (KnowItAll)
- Intuition: tuple \( t = (e_i, r_{ij}, e_j) \) is a correct instance of relation \( r_{ij} \) if it is extracted from many different sentences
Key components:
- Self-supervised learner
- Single-pass extractor
- Redundancy-based extractor
- Query processing
Query Processing

- Using Inverted Index distributed over a pool of machines
- Each relation is assigned to one machine
- Each machine then store a reference to all tuples that are instances of any relation assigned to it
- Like a Distributed Hash Table
Query Processing

- Relation centric index
- Can be used for advanced natural language like searching and answering
- Distributed pool of machines support interactive search speed
Experimental Results

- Comparison with Traditional IE
- Global Statistics on Facts Learned
Comparison with Traditional IE

- TextRunner VS KnowItAll
  - Open IE vs Closed IE
  - 10 relations are pre-selected

```
(<proper noun>, acquired, <proper noun>)
(<proper noun>, graduated from, <proper noun>)
(<proper noun>, is author of, <proper noun>)
(<proper noun>, is based in, <proper noun>)
(<proper noun>, studied, <noun phrase>)
(<proper noun>, studied at, <proper noun>)
(<proper noun>, was developed by, <proper noun>)
(<proper noun>, was formed in, <year>)
(<proper noun>, was founded by, <proper noun>)
(<proper noun>, worked with, <proper noun>)
```
Comparison with Traditional IE

<table>
<thead>
<tr>
<th></th>
<th>Average Error rate</th>
<th>Correct Extractions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TextRunner</strong></td>
<td>12%</td>
<td>11,476</td>
</tr>
<tr>
<td><strong>KnowItAll</strong></td>
<td>18%</td>
<td>11,631</td>
</tr>
</tbody>
</table>

- Speed-wise: TextRunner, 85 CPU hours for all relations in corpus at once; KnowItAll, 6.3 hours per relation
Global Statistics on Facts Learned

□ Evaluation Goal:

- How many of the tuples found represent actual relationships with plausible arguments
- What subset of these tuples is correct?
- How many of these tuples are distinct?
Global Statistics on Facts Learned

- Data Set used:
  - 9 million Web pages
  - 133 million sentences
  - 60.5 million tuples extracted (2.2 tuples per sentence)
Filtering Criteria

- Tuples with probability > 0.8
- Tuple’s relation is supported by 10 distinct sentences
- Not a general relation (top 0.1% relations) e.g. (NP1, has, NP2)

- A result of 11.3 million tuples containing 278,085 distinct relation strings.
Estimating the Number of Distinct Facts

- Only address relation synonymy
- Merge relation by using linguistic/syntactic components (punctuation, auxiliary verbs, leading stopwords, use of active and passive voice)
- Reduce the number of distinct relations to 91% of the number before merging
Estimating the Number of Distinct Facts

- Difficulty: rare to find two distinct relations that are truly synonymous in all senses of each phrase
  - E.g. person *develop* diseases vs. scientist *develop* technology
- Use synonymy clusters, human involved assessment at tuple level
## Estimating the Number of Distinct Facts

<table>
<thead>
<tr>
<th>Relation</th>
<th>Entity 1</th>
<th>Entity 2</th>
<th>Property</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>Bletchley Park</td>
<td>was location of</td>
<td>Station X</td>
<td></td>
</tr>
<tr>
<td>$R_2$</td>
<td>Bletchley Park</td>
<td>being called</td>
<td>Station X</td>
<td></td>
</tr>
<tr>
<td>$R_2$</td>
<td>Bletchley Park</td>
<td>known as</td>
<td>Station X</td>
<td></td>
</tr>
<tr>
<td>$R_2$</td>
<td>Bletchley Park</td>
<td>codenamed</td>
<td>Station X</td>
<td></td>
</tr>
</tbody>
</table>

Cluster found by $(e_1, p, e_2)$, $(e_1, r, e_2)$, where $p \neq r$.

At least 92% of the tuples found by TEXTRUNNER express distinct assertions (over estimation).
Estimating the Number of Distinct Facts

- Challenge: find methods for detecting synonyms and resolving multiple mentions of entities
Related Work

- KnowItAll Project (umbrella project)
- IBM Watson
- TextRunner Demo
Box Questions?