InstaRead: An IDE for IE
Short Presentation in CSE454

Raphael Hoffmann
January 17th, 2013

From Text To Knowledge
Citigroup has taken over EMI, the British music label of the Beatles and Radiohead, under a restructuring of its debt, EMI announced on Tuesday.

Human Effort

<table>
<thead>
<tr>
<th>Evaluation</th>
<th>#rels</th>
<th>#ann words</th>
<th>dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACE 2004</td>
<td>51</td>
<td>train 300K</td>
<td>12</td>
</tr>
<tr>
<td>MR IC 2010</td>
<td>15</td>
<td>train 115K</td>
<td>12</td>
</tr>
<tr>
<td>MR KBP 2011</td>
<td>16</td>
<td>train 8K</td>
<td>1</td>
</tr>
</tbody>
</table>

Motivates research on learning with less supervision, but best-performing systems use more direct human input (eg. rules)

A Typical IE System

Development Cycle

Need Many Iterations
Iteration is Slow

- ML expertise required
- Different prog. lang., different data structures
- No interactive speeds
- No expressive rule language connecting components
- Analyze
- Adjust
- ML algorithm
- Integrate resource (e.g. WordNet)
- No appropriate visualization
- No standardized formats
- No direct manipulation
- No advanced queries
- Remove barriers!

InstaRead: An IDE for IE

- Load text datasets
- Create relations
- Provide DBs of instances
- Visualizations
- See distributionally similar words
- Write extraction rules in logic
- Set rule recommendations
- Collect & organize rules
- Statistics

Key Ideas

1. **User writes rules in simple, expressive language**
   - Use First-Order Logic
2. **User instantly sees extractions on lots of text**
   - Use Database Indexing
3. **User gets automatic rule suggestions**
   - Use Bootstrapping, Learning

Demo

- Browser-based tool
- API
  - cd /projects/pardosa/s1/raphael/github/readr/exp
  - source ../../init.sh
  - mvn exec:java -Dexec.mainClass=newexp.Materialize

Project Ideas

In general: Create a new component & use InstaRead to explore, tie components together, and analyze

- Integrate exist. component (eg. Entity Linking, OpenIE)
- Mine rules from WordNet, FrameNet, VerbNet, Wiktionary
- Generate rule-suggestions through clustering
- Generate rule-suggestions through (better) bootstrapping
- Set rule (and thus extraction) confidence based on overlap with knowledge-base
- Develop rules/code to handle specific linguistic phenomenon (eg. time, modality, negation)
- Experiment with joint-inference of parsing + rules

More detailed slides (optional)
Rules are Crucial

For both hand-engineered and learned systems

- Example
  \[ e(x, y) = \text{Friends}(x, y) \]

- Rules as patterns
- Rules as features
- Rules (or space of rules) typically supplied

Goal: Enable experts to write quality rules extremely quickly

![Graph showing quality vs. time with 1h label]

Writing Rules in Logic

- FOL\(^1\) is simple, expressive, extensible, widely used
- Introduce predicates for NER, dependencies, ...

- Rules are deterministic, execute in defined order

![Example rule]

Rule Composition

- Define new predicates for similar substructure

- More compact set of rules, better generalization

Enabling Instant Execution

- Translation to SQL allows dynamic optimization

- Each predicate maps to a fragment of SQL
- Intensional and extensional predicates
- Indices, caching, SSDs important

Experimental Setup

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>22M news sentences</td>
<td>Develop 4 relational extractors in 55min each</td>
</tr>
<tr>
<td>NYTimes1(^3)</td>
<td></td>
</tr>
<tr>
<td>1M news sentences</td>
<td>Compare to a weakly supervised system</td>
</tr>
<tr>
<td>NYTimes07(^2)</td>
<td></td>
</tr>
<tr>
<td>22M news sentences</td>
<td>- Bootstrap, Keyword, Morphology, and Decomposition features</td>
</tr>
<tr>
<td>NYTimes07(^2)</td>
<td>- To gold annotations for error analysis</td>
</tr>
<tr>
<td>5K selected sentences</td>
<td></td>
</tr>
<tr>
<td>(some gold annotations)</td>
<td></td>
</tr>
<tr>
<td>CoNLL04(^4)</td>
<td></td>
</tr>
</tbody>
</table>

Comparison to Weakly Supervised Extraction

<table>
<thead>
<tr>
<th>Rules</th>
<th>attendedSchool</th>
<th>founded</th>
<th>killed</th>
<th>married</th>
</tr>
</thead>
<tbody>
<tr>
<td>Precision</td>
<td>1.00</td>
<td>0.91</td>
<td>0.90</td>
<td>0.90</td>
</tr>
<tr>
<td>#extractions</td>
<td>1,411</td>
<td>997</td>
<td>189</td>
<td>4,694</td>
</tr>
<tr>
<td>Weakly supervised</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Precision</td>
<td>0.71</td>
<td>N/A</td>
<td>0.50</td>
<td>0.50</td>
</tr>
<tr>
<td>#extractions</td>
<td>5</td>
<td>14</td>
<td>N/A</td>
<td>2</td>
</tr>
</tbody>
</table>

- Precision consistently at least 90%
- Works well, even when weak supervision fails

\(^1\) We use the subset referred to as 'safe domain - relational calculus'

\(^2\) LDC2008T19, \(^3\) Roth & Yih, 2004

\(^4\) Roth & Yih, 2004
Development Phases

1. Bootstrap Tool (0:15)
2. Keywords Tool (0:15)
3. Morphology Feature (0:05) mined morphology from Wiktionary (tense etc.)
4. Decomposition Feature (0:20) enable chaining of rules

Comparison of Development Phases

- Bootstrap initially effective, but recall limited
- Decomposition effective for some relations

Error Analysis

- On CoNLL04 ‘killed’ wrt gold annotations: Re .34, Pr .98
  
<table>
<thead>
<tr>
<th>False Negatives due to Preprocessing (missing predictions)</th>
<th>False Positives due to Preprocessing (wrong predictions)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NER</td>
<td>NER</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Dependencies</td>
</tr>
<tr>
<td>Co-references</td>
<td>Co-references</td>
</tr>
<tr>
<td>Rules (missing predictions)</td>
<td>Rules (wrong predictions)</td>
</tr>
<tr>
<td>Lexical items</td>
<td>Lexical items</td>
</tr>
<tr>
<td>Syntactic variation</td>
<td>Syntactic variation</td>
</tr>
<tr>
<td>Reasoning chain</td>
<td>Reasoning chain</td>
</tr>
</tbody>
</table>

- 36% of all errors are due to incorrect pre-processing

Joint Parsing & Relation Extraction

- If top parse is wrong ...
  - In 35% of cases: correct at top-2
  - In 50% of cases: correct among top-5
  - In 90% of cases: correct among top-50
- Heuristic: Select first parse among top-k with most extractions

<table>
<thead>
<tr>
<th>Recall</th>
<th>Precision</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pipeline</td>
<td>.342</td>
<td>.978</td>
</tr>
<tr>
<td>Joint</td>
<td>.412</td>
<td>.973</td>
</tr>
</tbody>
</table>

Runtime Performance

<table>
<thead>
<tr>
<th>avg</th>
<th>median</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>#SQL queries per relation (55min)</td>
<td>55</td>
<td>54</td>
</tr>
<tr>
<td>#join tables per SQL query</td>
<td>4.3</td>
<td>4</td>
</tr>
<tr>
<td>Execution time (s) per SQL query</td>
<td>1.5</td>
<td>0.74</td>
</tr>
</tbody>
</table>

- On 22M sentences, 3.7B rows in 75 tables, 140GB
- Most queries execute in 74ms or less
- Outliers due to Bootstrap tool
  - Aggregation over millions of rows motivates streaming or sampling

Logic

- Often Horn clauses are sufficient, but sometimes we need ¬, ∨, ∃
- Example 1: founded relation

  Michael Dell built his first company in a dorm-room.

  Mr. Harris built Dell into a formidable competitor to IBM.

  Desired conjunct

  - Example 2: Integration of co-reference

  “nearest noun-phrase which satisfies …”