Range of problems that make named entity recognition (NE) hard
Mikheev et al’s (1998) cascading NE system
NE is the simplest kind of IE task: no relations between entities must be determined
NIST MUC conferences pose three kinds of harder IE tasks
Today: more of the full task (scenario templates), and on learning
• “Flattened-out” semantic representations with lexemes directly hard-wired into them
• String-based matching with type of semantic category to be found directly expressed in lexical pattern
• Problem with all string-based mechanisms: generalisation to other strings with similar semantics, and to only those
• Do generalisation by hand...
  – <Perpetrator> (APPOSITION) {blows/blew/has blown} {himself/herself} up
  – <Perpetrator> detonates
  – {blown up/detonated} by <Perpetrator>
• Manual production of patterns is time-consuming, brittle, and not portable across domains

Learning of lexico-semantic patterns (Riloff 1993)

• UMASS participant system in MUC-4: AutoSlog
• Lexico-semantic patterns for MUC-3 took 1500 person hours to build → knowledge engineering bottleneck
• AutoSlog achieved 98% performance of manual system; AutoSlog dictionary took 5 person hours to build
• “Template mining:”
  – Use MUC training corpus (1500 texts + human answer keys; 50% non-relevant texts) to learn contexts
  – Have human check the resulting templates (30% - 70% retained)
• 389 Patterns (“concept nodes”) with enabling syntactic conditions, e.g. active or passive:
  – kidnap-passive: \(<\text{VICTIM}>\) expected to be subject
  – kidnap-active: \(<\text{PERPETRATOR}>\) expected to be subject
• Hard and soft constraints for fillers of slots
  – Hard constraints: selectional restrictions; soft constraints: semantic preferences
• Semantic lexicon with 5436 entries (including semantic features)

Heuristics for supervised template mining (Riloff 1993)

• Stylistic conventions: relationship between entity and event made explicit in first reference to the entity
• Find keyword there which triggers the pattern: kidnap, shot,
• Heuristics to find these trigger words
• Given: filled template plus raw text. Algorithm:
  – Find first sentence that contains slot filler
  – Suggest good conceptual anchor point (trigger word)
  – Suggest a set of enabling conditions

"the diplomat was kidnapped" + VICTIM: the diplomat

Suggest: \(<\text{SUBJECT}>\) passive-verb + trigger=kidnap
System uses 13 “heuristics” (= syntactic patterns):

<table>
<thead>
<tr>
<th>EXAMPLE</th>
<th>PATTERN</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;victim&gt; was murdered</td>
<td>&lt;subject&gt; passive-verb</td>
</tr>
<tr>
<td>&lt;perpetrator&gt; bombed</td>
<td>&lt;subject&gt; active-verb</td>
</tr>
<tr>
<td>&lt;perpetrator&gt; attempted to kill</td>
<td>&lt;subject&gt; verb infinitive</td>
</tr>
<tr>
<td>&lt;victim&gt; was victim</td>
<td>subject auxiliary &lt;noun&gt;</td>
</tr>
<tr>
<td>killed &lt;victim&gt;</td>
<td>passive-verb &lt;dobj&gt;</td>
</tr>
<tr>
<td>bombed &lt;target&gt;</td>
<td>active-verb &lt;dobj&gt;</td>
</tr>
<tr>
<td>to kill &lt;victim&gt;</td>
<td>infinitive &lt;dobj&gt;</td>
</tr>
<tr>
<td>threatened to attack &lt;target&gt;</td>
<td>verb infinitive &lt;dobj&gt;</td>
</tr>
<tr>
<td>killing &lt;victim&gt;</td>
<td>gerund &lt;dobj&gt;</td>
</tr>
<tr>
<td>fatality was &lt;victim&gt;</td>
<td>noun auxiliary &lt;dobj&gt;</td>
</tr>
<tr>
<td>bomb against &lt;target&gt;</td>
<td>noun prep &lt;np&gt;</td>
</tr>
<tr>
<td>killed with &lt;instrument&gt;</td>
<td>active-verb prep &lt;np&gt;</td>
</tr>
<tr>
<td>was aimed at &lt;target&gt;</td>
<td>passive-verb prep &lt;np&gt;</td>
</tr>
</tbody>
</table>

Riloff 1993: a good concept node

ID: DEV-MUC4-0657
Slot Filler: “public buildings”
Sentence: IN LA OROYA, JUNIN DEPARTMENT, IN THE CENTRAL PERUVIAN MOUNTAIN RANGE, PUBLIC BUILDINGS WERE BOMBED AND A CAR-BOMB WAS DETONATED.

CONCEPT NODE
Name: target-subject-passive-verb-bombed
Trigger: bombed
Variable slots: (target (*S* 1))
Constraints: (class phys-target *S*)
Constant slots: (type bombing)
Enabling Conditions: ((passive))
CONCEPT NODE
Name: perpetrator-subject-verb-innitive-threatened-to-murder
Trigger: murder
Variable slots: (perpetrator (*S* 1))
Constraints: (class perpetrator *S*)
Constant slots: (type perpetrator)
Enabling Conditions: ((active) (trigger-preceded-by? 'to 'threatened))
<table>
<thead>
<tr>
<th>System/Test Set</th>
<th>Recall</th>
<th>Prec</th>
<th>F-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUC-4/TST3</td>
<td>46</td>
<td>56</td>
<td>50.5</td>
</tr>
<tr>
<td>AutoSlog/TST3</td>
<td>43</td>
<td>56</td>
<td>48.7</td>
</tr>
<tr>
<td>MUC-4/TST4</td>
<td>44</td>
<td>40</td>
<td>41.9</td>
</tr>
<tr>
<td>AutoSlog/TST4</td>
<td>39</td>
<td>45</td>
<td>41.8</td>
</tr>
</tbody>
</table>

- 5 hours of sifting through AutoSlog’s patterns
- Porting to new domain in less than 10 hours of human interaction
- But: creation of training corpus ignored in this calculation

**Agichtein, Gravano (2000): Snowball**

- Find locations of headquarters of a company and the corresponding company name (<o,l> tuples)

<table>
<thead>
<tr>
<th>Organisation</th>
<th>Location of Headquarters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft</td>
<td>Redmond</td>
</tr>
<tr>
<td>Exxon</td>
<td>Irving</td>
</tr>
<tr>
<td>IBM</td>
<td>Armonk</td>
</tr>
<tr>
<td>Boeing</td>
<td>Seattle</td>
</tr>
<tr>
<td>Intel</td>
<td>Santa Clara</td>
</tr>
</tbody>
</table>

“Computer servers at Microsoft’s headquarters in Redmond”

- Use minimal human interaction (handful of positive examples)
  - no manually crafted patterns
  - no large annotated corpus (IMass system at MUC-6)
- Automatically learn extraction patterns
- Less important to find every occurrence of patterns; only need to fill table with confidence
Agichtein, Gravano (2000): Overall process

- Start from table containing some $<o, l>$ tuples (which must exist in document collection)
- Perform NE (advantage over prior system DIPRE (Brin 98))
- System searches for occurrences of the example $<o, l>$ tuples in documents
- System learns extraction patterns from these example contexts, e.g.:

  $<$ORGANIZATION$>$’s headquarters in $<$LOCATION$>$
  $<$LOCATION$>$-based $<$ORGANIZATION$>$

- Evaluate patterns; use best ones to find new $<o, l>$ tuples
- Evaluate new tuples, choose most reliable ones as new seed tuples
- Iteratively repeat the process
A SNOWBALL pattern is a 5-tuple \(<\text{left}, \text{tag}_1, \text{middle}, \text{tag}_2, \text{right}>\)

<table>
<thead>
<tr>
<th>left</th>
<th>Tag1</th>
<th>middle</th>
<th>Tag2</th>
<th>right</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Irving-based Exxon Corporation</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(&lt;{\text{the}, 0.2&gt;}, \text{LOCATION}, {\text{&lt;&gt;,0.5&gt; &lt;based}, 0.5&gt;}, \text{ORGANIZATION}, {\text{&gt;} &gt;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Associate term weights as a function of frequency of term in context
- Normalize each vector so that norm is 1; then multiply with weights \(W_{\text{left}}, W_{\text{right}}, W_{\text{mid}}\).
- Degree of match between two patterns \(t_p = <l_p, t_1, m_p, t_2, r_p>\) and \(t_s = <l_s, t'_1, m_s, t'_2, r_s>\):

  \[
  \text{match}(t_p, t_s) = l_pl_s + m_pm_s + r_pr_s \quad \text{(if tags match, 0 otherwise)}
  \]

- Similar contexts form a pattern
  - Cluster vectors using a clustering algorithm (minimum similarity threshold \(\tau_{\text{sim}}\))
  - Vectors represented as cluster centroids \(\bar{l}_s, \bar{m}_s, \bar{r}_s\)
- Generalised Snowball pattern defined via centroids:

  \(<\bar{l}_s, \text{tag}_1, \bar{m}_s, \text{tag}_2, \bar{r}_s>\)

- Remember for each Generalised Snowball pattern
  - All contexts it came from
  - The distances of contexts from centroid
• We want **productive** and **reliable** patterns
  
  – productive but not reliable:
  
  \[ < \{\}, ORGANIZATION, \{"","", 1 \} >, LOCATION, \{\} > \]

  "Intel, Santa Clara, announced that. . ."  
  "Invest in Microsoft, New York-based analyst Jane Smith said. . ."

  – reliable but not productive:

  \[ < \{\}, ORGANIZATION, \{ < whose, 0.1 >, < headquarter, 0.4 >, < is, 0.1 > < located, 0.3 >, < in, 0.09 >, < nearby, 0.01 > \} >, LOCATION, \{\} > \]

  "Exxon, whose headquarter is located in nearby Irving. . ."

• Eliminate patterns supported by less than \( \tau_{sup} < o, l > \) tuples

Agichtein, Gravano (2000): Pattern reliability

• If \( P \) predicts tuple \( t = < o, l > \) and there is already tuple \( t' = < o, l' > \) with high confidence, then: if \( l = l' \rightarrow P\.positive++ \), otherwise \( P\.negative++ \) (uniqueness constraints: organization is key).

• Pattern reliability: \( Conf(P) = \frac{P\.positive}{P\.positive + P\.negative} \) (range \([0..1]\))

• Example:
  
  \( P_{43} = < \{\}, ORGANIZATION, \{"","", 1 \} >, LOCATION, \{\} > \) matches
  
  1. **Exxon**, **Irving**, said... (CORRECT: in table)
  2. **Intel**, **Santa Clara**, cut prices (CORRECT: in table)
  3. invest in **Microsoft**, **New York**-based analyst (INCORRECT, contradicted by entry \( <\text{Microsoft}, \text{Redmont}> \))
  4. found at **ASDA**, **Irving**. (????, unknown, no contradiction → disregard evidence)

  • disregard unclear evidence such as 4.

  • Thus, \( Conf(P_{43}) = \frac{2}{2+1} \)
• Consider productivity, not just reliability:

\[ Conf_{RlogF}(P) = Conf(P) \log_2(P_{positive}) \]

• Normalized \( Conf_{RlogFNorm}(P) \):

\[ Conf_{RlogFNorm}(P) = \frac{Conf_{RlogF}(P)}{\max_{i \in P} Conf(i)} \]

(this brings \( Conf_{RlogFNorm}(P) \) into range [0...1])

• \( \max_{i \in P} Conf(i) \) is the largest confidence value seen with any pattern

• \( Conf_{RlogFNorm}(P) \) is a rough estimate of the probability of pattern \( P \) producing a valid tuple (called \( Conf(P) \) hereafter)

---

Agichtein, Gravano (2000): Tuple evaluation I

• Confidence of a tuple \( T \) is probability that at least one valid tuple is produced:

\[ Conf(T) = 1 - \prod_{i=0}^{\left| P \right|} (1 - Conf(P_i) \cdot Match(C_i, P_i)) \]

\( P = \{ P_i \} \) is the set of patterns that generated \( T \)

\( C_i \) is the context associated with an occurrence of \( T \)

\( Match(C_i, P_i) \) is goodness of match between \( P_i \) and \( C_i \)

• Explanation: probability of every pattern matched incorrectly:

\[ Prob(T \text{ IS NOT valid}) = \prod_{i=0}^{\left| P \right|} (1 - P(i)) \]

• Formula due to the assumption that for an extracted tuple \( T \) to be valid, it is sufficient that at least one pattern matched the “correct” text context of \( T \).
• Then reset confidence of patterns:

\[ Conf(P) = Conf_{new}(P)W_{\text{updt}} + Conf_{old}(P)(1 - W_{\text{updt}}) \]

\(W_{\text{updt}}\) controls learning rate: does system trust old or new occurrences more? Here: \(W_{\text{updt}} = 0.5\)

• Throw away tuples with confidence < \(\tau_t\)

\[
\begin{array}{|c|c|c|}
\hline
\text{Conf} & \text{middle} & \text{right} \\
\hline
1 & <\text{based, .53}>, <\text{in, .53}> & <"\text{, .01}> \\
.69 & <"\text{, .42}>, <\text{, .42}>, <\text{headquarters, .42}>, <\text{, .42}> & <\text{, .93}>, <\text{,), .12}> \\
.61 & <\text{, .93}> & <\text{,), .12}> \\
\hline
\end{array}
\]

• Use training corpus to set parameters: \(\tau_{\text{sim}}, \tau_t, \tau_{\text{sup}}, I_{\text{max}}, W_{\text{left}}, W_{\text{right}}, W_{\text{middle}}\)

• Only input: 5 < \(o, l\) > tuples

• Punctuation matters: performance decreases when punctuation is removed

• Recall b/w .78 and .87 \((\tau_{\text{sup}} > 5)\); precision .90 \((\tau_{\text{sup}} > 4)\)

• High precision possible (.96 with \(\tau_t = .8\)); remaining problems come from NE recognition

• Pattern evaluation step responsible for most improvement over DIPRE
Possible to learn simple relations from positive examples (Snowball)
Possible to learn more diverse relations from annotated training corpus (Riloff)
Even modest performance can be useful
  – Later manual verification
  – In circumstances where there would be no time to review source documents, so incomplete extracted information is better than none

Current methods perform well if
  • Information to be extracted is expressed directly (no complex inference is required)
  • Information is predominantly expressed in a relatively small number of forms
  • Information is expressed locally within the text

Difference between IE and QA (next time):
  • IE is domain dependent, open-domain QA is not