Named-entity recognition

What is a name?
Message Understanding Conference (MUC)
Named Entity Task Definition
MUC results around 90-98%

Named-entity recognition: The default case:

- A name is the longest string matching:
  \[\text{UCLetter Letter+} \text{ SPACE UCLetter Letter+}*\]

- Thus, to mark up these, we say:
  \[\text{UCLetter Letter+} \text{ SPACE UCLetter Letter+}* \rightarrow %[ ... %]\]

Named-entity recognition: Some difficult cases:

- Sentence initial words
- "Även", "Det",...
- "Björn", "Ström", "Ek", "The",...
- Capitalized words that are not names
- "Master of Fine Arts (MFA)"
- Abbreviations that are not names
- "Mr.", "Dr.",...
- Some use of pronouns (e.g. in letters)
- "It has almost 30 out of an estimated 150 feeling,..."
- Compounds that are not names
- "English teacher", "Swedish teacher",
- Two names in a row
- "Vera Erik John"
- Names that include non-names
- "Transport for London", "Ernst von Blanken", "Gleis transportvernert",
- Foreign names
- "The International Institute for Strategic Studies"
- Uppercase words (e.g. in headings)
- "Några av de 12 st" (..."

Things that might help:
- Wordlist lookup
- Gazetteer (phone numbers, people...)
- Ordinary words (not including names)
- Clue words in names (e.g. "the", "of", "in", "from", (effect) included in names or words (effect) in the context of names)
- Sentence splitting

Named-entity recognition in the LaSIE-II system (Sheffield)

Modules:
- Tokenizer
- Gazetteer lookup
- Sentence splitter
- Brill tagger
- Morphological analyzer
- Parser
  - with named-entity grammar
  - with general phrase grammar
- Name matcher
- Discourse Interpreter
- Template Writer

Part-of-speech tagging

Knowledge

Text

Processor

POS tagged text

Needed:
- some strategy for representing the knowledge
- some method for acquiring the knowledge
- some method of applying the knowledge
Some POS-tagging issues
- Accuracy
- Speed
- Space requirements
- Robustness
- Learning
- Underspecification
- Declarativity
- Order-independence of rules

Finite-state POS tagging
- Approaches
  - Transformation-based tagging (Brill 1992, Roche & Schabes 1995)
  - Negative Grammar (Roche 1995)
  - Simulating a HMM with a FSM (not in this course)
  - Constraint Grammar (not in this course, but note the similarity with Negative Grammar)

Transformation-based tagging
- Representational strategy:
  - Simple lexica
  - Ordered lists of transformations, conditioned on (small amounts) of local context
- Learning strategy: Transformation-based learning (not in this course)

Transformation-based tagging
- Three steps:
  - Lexical look-up
  - Lexical rule application for unknown words
  - Contextual rule application

Transformation-based tagging
- Principle:
  - Pick the most frequent POS for each word
  - For unknown words, assign PN if it begins with capital letter, else assign NN.
- Example:
  - occurrences of the word "table" gets assigned the tag NN, even though it might also occur (less often) as a VB.
Handling unknown words

- Example 1:
  - NN's fhassuf 1 NNS
  - means "if a word is currently tagged NN, and has a suffix of length 1 which consists of the letter 's', change its tag to NNS"

- Example 2:
  - ly hassuf 2 RB
  - means "if a word has a suffix of length 2 consisting of the letter sequence 'ly', change its tag to RB (regardless of the initial tag)"

Contextual rule application

- Sample rules
  - vbd vbn PREVTAG np
  - vbd vbn NEXTTAG by

- Application: Apply each rule in the cascade once to the whole corpus. Start from the top.

Contextual rule application

- Examples after lexical lookup and lexical rule application, but before contextual rule application:
  - Chapman/np killed/vbd John/np Lennon/np
  - He/pps witnessed/vbd John/np Lennon/np

- Examples after application of contextual rule 'vbd vbn PREVTAG np':
  - Chapman/np killed/vbd John/np Lennon/np
  - He/pps witnessed/vbd John/np Lennon/np

- Examples after application of contextual rule 'vbd vbn NEXTTAG by':
  - Chapman/np killed/vbd John/np Lennon/np
  - He/pps witnessed/vbd John/np Lennon/np

Assessing the FST Brill tagger

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy</td>
<td>96.5%</td>
</tr>
<tr>
<td>Speed</td>
<td>Very fast</td>
</tr>
<tr>
<td>Space req.</td>
<td>Moderate</td>
</tr>
<tr>
<td>Robustness</td>
<td>Robust</td>
</tr>
<tr>
<td>Learning</td>
<td>Yes</td>
</tr>
<tr>
<td>Declarativity</td>
<td>Yes</td>
</tr>
<tr>
<td>Order-independence</td>
<td>No</td>
</tr>
</tbody>
</table>

Negative Grammar for POS tagging

- Regular expressions
- Ambiguity FST tagged text as FSA
- FSA
- Less ambiguous FOS tagged text as FSA

Negative grammar (1)

- define S [he Pro hopes [NIV] that [Pro Conj Det] this [Det] [Det] works [NIV]]:
- define C1 [that Det this Det]:
- define S' [S - [P* C3 P*]]:
- print net S'

- define C2 [this Det ? V]:
- define S'' [S' - [P* C3 P*]]:
- print net S''
Negative grammar (2)

define S [he Pro hopes [N|V] that [Pro|Conj|Det] this [Det|Pro] works [N|V]];
define C1 (that Det this Det); define C2 (this Det ? V); define C3 (? Pro ? N);
define S' [S - [?* [C1|C2|C3] ?*]];
print net S'

Negative grammar (3)

define S [he Pro hopes [N|V] that [Pro|Conj|Det] this [Det|Pro] works [N|V]];
define C1 (that Det this Det); define C2 (this Det ? V); define C3 (? Pro ? N);
define S' [S - [?* [C1|C2|C3] ?*]];
print net S'

Negative Grammar

- Since things like inclusion and equivalence between regular languages are decidable we may test our proposed rules with respect to these properties before adding them to our grammar.

Assessing negative grammar

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>Brill</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>96.5%</td>
<td>?</td>
</tr>
<tr>
<td>Speed</td>
<td>Fast</td>
<td>Slow?</td>
</tr>
<tr>
<td>Space req.</td>
<td>Moderate</td>
<td>?</td>
</tr>
<tr>
<td>Robustness</td>
<td>Robust</td>
<td>Not robust</td>
</tr>
<tr>
<td>Underspecification</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Learning</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>Declarativity</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Order-independence of rules</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Tentative comparison of FST POS taggers

<table>
<thead>
<tr>
<th>PARAMETER</th>
<th>Brill</th>
<th>Negative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Correctness</td>
<td>96.5%</td>
<td>?</td>
</tr>
<tr>
<td>Speed</td>
<td>Fast</td>
<td>Slow?</td>
</tr>
<tr>
<td>Space req.</td>
<td>Moderate</td>
<td>?</td>
</tr>
<tr>
<td>Robustness</td>
<td>Robust</td>
<td>Not robust</td>
</tr>
<tr>
<td>Underspecification</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Learning</td>
<td>Yes</td>
<td>?</td>
</tr>
<tr>
<td>Declarativity</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Order-independence of rules</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>