Chapter 9:
Named Entity Tagging
(NE Tagging)
Reference

Nymble: a High-Performance Learning Name-finder

Daniel M. Bikel
BBN Corporation
70 Fawcett Street
Cambridge, MA 02138
dbikel@bbn.com

Scott Miller
BBN Corporation
70 Fawcett Street
Cambridge, MA 02138
szmiller@bbn.com

Richard Schwartz
BBN Corporation
70 Fawcett Street
Cambridge, MA 02138
schwartz@bbn.com

Ralph Weischedel
BBN Corporation
70 Fawcett Street
Cambridge, MA 02138
weisched@bbn.com
Identify in a document

- Names of
  - Organizations
  - Locations
  - Persons
- Times
- Dates
- Monetary amounts
- Percentages
What is

• Masouleh
• Martonyi

• Person? Location? Organisation?
Example

Some 30 people have been killed and 45 others been injured in floods and landslide caused by torrential rains in the historical city of <ENAMEX id="76" type="LOCATION">Masouleh</ENAMEX>, <ENAMEX id="77" type="LOCATION">Iran</ENAMEX>'s northern Gilan province.
Example

<ENAMEX id="238" type="LOCATION">Germany</ENAMEX> was Hungary's most important ally on this issue,
<ENAMEX id="242" type="PERSON">Martonyi</ENAMEX> added.
Task

- Classification task
- Given:
  - sequence of words $w_1 .. w_n$
- Wanted:
  - Sequence of name classes $N_1 .. N_n$
- Note:
  - The name class can also be “not-a-name”
Formal Solution

\[
(\hat{N}_1, \hat{N}_2...\hat{N}_n) = \arg \max_{N_1, N_2...N_n} P(N_1, N_2...N_n \mid w_1, w_2...w_n)
\]

\[
= \arg \max_{N_1, N_2...N_n} P(w_1, w_2...w_n \mid N_1, N_2...N_n) P(N_1, N_2...N_n)
\]

\[
= \arg \max_{N_1, N_2...N_n} P(w_1, w_2...w_n N_1, N_2...N_n)
\]

Do decomposition of probability in a more principled fashion
\[\mapsto\] black board
Formal Solution

\[ P(w_1, w_2 \ldots w_n \, N_1, N_2 \ldots N_n) = \]
\[ \prod_{i=1}^{n} P(w_i \mid w_1 \ldots w_{i-1}, N_1 \ldots N_i) \]
\[ \times \prod_{i=1}^{n} P(N_i \mid w_1 \ldots w_{i-1}, N_1 \ldots N_{i-1}) \]
Markov-Approximations

Emission probabilities

\[ P(w_i \mid w_{1\ldots i-1}, N_{1\ldots i}) \approx P(w_i \mid w_{i-1}, N_{i-1} N_i) \]

Transition probabilities

\[ P(N_i \mid w_{1\ldots i-1}, N_{1\ldots i-1}) \approx P(N_i \mid w_{i-1}, N_{i-1}) \]

→ if you want, you can keep more terms
Backing-Off Hierarchy for Transition Probabilities

\[ P(N_i \mid w_{i-1}, N_{i-1}) \]
\[ \rightarrow P(N_i \mid N_{i-1}) \]
\[ \rightarrow P(N_i) \]
\[ \rightarrow \frac{1}{\text{Num. Name Classes}} \]

Use absolute discounting or linear interpolation to implement this hierarchy
Backsc-Off Hierarchy for Emission Probabilities

\[ P(w_i \mid w_{i-1}, N_{i-1}N_i) \]
\[ \rightarrow P(w_i \mid w_{i-1}N_i) \]
\[ \rightarrow P(w_i \mid N_i) \]
\[ \rightarrow P(w_i \mid f_i)P(f_i \mid N_i) \]
\[ \rightarrow \frac{1}{|V|} \]

Use absolute discounting or linear interpolation to implement this hierarchy
Class features f

<table>
<thead>
<tr>
<th>Word Feature</th>
<th>Example Text</th>
<th>Intuition</th>
</tr>
</thead>
<tbody>
<tr>
<td>twoDigitNum</td>
<td>90</td>
<td>Two-digit year</td>
</tr>
<tr>
<td>fourDigitNum</td>
<td>1990</td>
<td>Four digit year</td>
</tr>
<tr>
<td>containsDigitAndAlpha</td>
<td>A8956-67</td>
<td>Product code</td>
</tr>
<tr>
<td>containsDigitAndDash</td>
<td>09-96</td>
<td>Date</td>
</tr>
<tr>
<td>containsDigitAndSlash</td>
<td>11/9/89</td>
<td>Date</td>
</tr>
<tr>
<td>containsDigitAndComma</td>
<td>23,000.00</td>
<td>Monetary amount</td>
</tr>
<tr>
<td>containsDigitAndPeriod</td>
<td>1.00</td>
<td>Monetary amount, percentage</td>
</tr>
<tr>
<td>otherNum</td>
<td>456789</td>
<td>Other number</td>
</tr>
<tr>
<td>allCaps</td>
<td>BBN</td>
<td>Organization</td>
</tr>
<tr>
<td>capPeriod</td>
<td>M.</td>
<td>Person name initial</td>
</tr>
<tr>
<td>firstWord</td>
<td>first word of sentence</td>
<td>No useful capitalization information</td>
</tr>
<tr>
<td>initCap</td>
<td>Sally</td>
<td>Capitalized word</td>
</tr>
<tr>
<td>lowercaseCase</td>
<td>can</td>
<td>Uncapitalized word</td>
</tr>
<tr>
<td>other</td>
<td>,</td>
<td>Punctuation marks, all other words</td>
</tr>
</tbody>
</table>

Feature class are processed in order
First matching feature class is taken
For the given approximations, the Viterbi algorithm can still be used.
### Performance

<table>
<thead>
<tr>
<th>Case</th>
<th>Language</th>
<th>Best Reported Score</th>
<th>Nymble</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed</td>
<td>English</td>
<td>96</td>
<td>93</td>
</tr>
<tr>
<td>Upper</td>
<td>English</td>
<td>89</td>
<td>91</td>
</tr>
<tr>
<td>Mixed</td>
<td>Spanish</td>
<td>93</td>
<td>90</td>
</tr>
</tbody>
</table>

All upper case was interesting for extracting named entities from text from a speech recognizer.
Performance vs. Amount of Training Data

System already well trained
Could build more sophisticated models
What is LingPipe?
LingPipe is a suite of Java libraries for the linguistic analysis of human language.

Feature Overview
LingPipe's information extraction and data mining:
- tracks mentions of entities (e.g. people or proteins);
- links entity mentions to database entries;
- uncovers relations between entities and actions;
- classifies text passages by language, character encoding, genre, topic, or sentiment;
- corrects spelling with respect to a text collection;
- clusters documents by implicit topic and discovers significant trends over time; and
- provides part-of-speech tagging and phrase chunking.

Architecture
LingPipe's architecture is defined to be efficient, scalable, small and robust. Highlights include:
- Java API with source code and unit tests;
- multi-lingual, multi-domain, multi-genre models;
- training with new data for new tasks;
- n-best output with statistical confidence estimates;
- online training (learn-a-little, tag-a-little);
- thread-safe models and decoders for concurrent-read single-write (CRSW) synchronization;
- character encoding-sensitive shared I/O framework.
Named Entity Detection

• The default detection distinguishes between three types of entities.
  • People (distinguishes male and female)
  • Place
  • Organization

• It can be trained to recognize any type of entity.
  • You can get corpora from online
  • You can annotate your own corpora using WordFreak, which also comes with LingPipe.
Dictionary

• To increase the accuracy of LingPipe, you can import a Dictionary.
• A dictionary will force the recognition of certain strings to be certain types.
• Common dictionaries include:
  • Gazeteer
  • List of people’s names
  • Company names
Language-Independent Named Entity Recognition (II)

Named entities are phrases that contain the names of persons, organizations, locations, times and quantities. Example:

\[ \text{[ORG U.N.] official [PER Ekeus] heads for [LOC Baghdad].} \]

The shared task of CoNLL-2003 concerns language-independent named entity recognition. We will concentrate on four types of named entities: persons, locations, organizations and names of miscellaneous entities that do not belong to the previous three groups. The participants of the shared task will be offered training and test data for two languages. They will use the data for developing a named-entity recognition system that includes a machine learning component. For each language, additional information (lists of names and non-annotated data) will be supplied as well. The challenge for the participants is to find ways of incorporating this information in their system.

Background information

Named Entity Recognition (NER) is a subtask of Information Extraction. Different NER systems were evaluated as a part of the Sixth Message Understanding Conference in 1995 (MUC6). The target language was English. The participating systems performed well. However, many of them used language-specific resources for performing the task and it is unknown how they would have performed on another language than English (PD97).

After 1995, NER systems have been developed for some European languages and a few Asian languages. There have been at least two studies that have applied one NER system to different languages. Palmer and Day (PD97) have used statistical methods for finding named entities in newswire articles in Chinese, English, French, Japanese, Portuguese and Spanish. They found that the difficulty of the NER task was different for the six languages but that a large part of the task could be performed with simple methods. Cucerzan and Yarowsky (Cy99) used both morphological and contextual clues for identifying named entities in English, Greek, Hindi, Rumanian and Turkish. With minimal supervision, they obtained overall F measures between 40 and 70, depending on the languages used. In the shared task at CoNLL-2002, twelve different learning systems were applied to data in Spanish and Dutch.

Software and Data
Results

Sixteen systems have participated in the CoNLL-2003 shared task. They used a wide variety of machine learning techniques and different feature sets. Here is the result table for the English test set:

<table>
<thead>
<tr>
<th>English</th>
<th>precision</th>
<th>recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIJ2003</td>
<td>88.99%</td>
<td>88.54%</td>
<td>88.76±0.7</td>
</tr>
<tr>
<td>CN03</td>
<td>88.12%</td>
<td>88.51%</td>
<td>88.31±0.7</td>
</tr>
<tr>
<td>KSNM03</td>
<td>85.93%</td>
<td>86.21%</td>
<td>86.07±0.8</td>
</tr>
<tr>
<td>ZJ03</td>
<td>86.13%</td>
<td>84.88%</td>
<td>85.50±0.9</td>
</tr>
<tr>
<td>CMP03b</td>
<td>84.05%</td>
<td>85.96%</td>
<td>85.00±0.8</td>
</tr>
<tr>
<td>CC03</td>
<td>84.29%</td>
<td>85.50%</td>
<td>84.89±0.9</td>
</tr>
<tr>
<td>MM03</td>
<td>84.45%</td>
<td>84.90%</td>
<td>84.67±1.0</td>
</tr>
<tr>
<td>CMP03a</td>
<td>85.81%</td>
<td>82.84%</td>
<td>84.30±0.9</td>
</tr>
<tr>
<td>ML03</td>
<td>84.52%</td>
<td>83.55%</td>
<td>84.04±0.9</td>
</tr>
<tr>
<td>BNO3</td>
<td>84.68%</td>
<td>83.18%</td>
<td>83.92±1.0</td>
</tr>
<tr>
<td>MLP03</td>
<td>80.87%</td>
<td>84.21%</td>
<td>82.50±1.0</td>
</tr>
<tr>
<td>MNC03</td>
<td>82.02%</td>
<td>81.39%</td>
<td>81.70±0.9</td>
</tr>
<tr>
<td>MPO03</td>
<td>81.60%</td>
<td>78.05%</td>
<td>79.78±1.0</td>
</tr>
<tr>
<td>HV03</td>
<td>76.33%</td>
<td>80.17%</td>
<td>78.20±1.0</td>
</tr>
<tr>
<td>DD03</td>
<td>75.84%</td>
<td>78.13%</td>
<td>76.97±1.2</td>
</tr>
<tr>
<td>Ham03</td>
<td>69.09%</td>
<td>53.26%</td>
<td>60.15±1.3</td>
</tr>
<tr>
<td>baseline</td>
<td>71.91%</td>
<td>50.90%</td>
<td>59.61±1.2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>German</th>
<th>precision</th>
<th>recall</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIJ2003</td>
<td>83.87%</td>
<td>63.71%</td>
<td>72.41±1.3</td>
</tr>
<tr>
<td>KSNM03</td>
<td>80.38%</td>
<td>65.04%</td>
<td>71.90±1.2</td>
</tr>
<tr>
<td>ZJ03</td>
<td>82.00%</td>
<td>63.03%</td>
<td>71.27±1.5</td>
</tr>
</tbody>
</table>
[CC03]

[DD03]

[FIJZ03]

[Ham03]

[HVo03]

[KSNM03]
Summary

- Named entity tagging:
  - Find names of people, organizations, locations …
- Nymble:
  - High quality system
  - Except for the class features language independent
- If you need ready to use software: LingPipe
- If you want to do research: look at CoNNL