CS544: Information Extraction, Named Entity Recognition and Classification

March 23, 2010

Zornitsa Kozareva
USC/ISI
Marina del Rey, CA
kozareva@isi.edu
www.isi.edu/~kozareva

Named Entity Recognition and Classification

• Identify mentions in text and classify them into a predefined set of categories of interest:
  – Person Names: Prof. Jerry Hobbs, Jerry Hobbs
  – Organizations: Hobbs corporation, FbK
  – Locations: Ohio
  – Date and time expressions: February 2010
  – E-mail: mkg@gmail.com
  – Web address: www.usc.edu
  – Names of drugs: paracetamol
  – Names of ships: Queen Marry
  – Bibliographic references:
    – ...

<PER>Prof. Jerry Hobbs</PER> taught CS544 during <DATE>February 2010</DATE>.
<PER>Jerry Hobbs</PER> killed his daughter in <LOC>Ohio</LOC>.
<ORG>Hobbs corporation</ORG> bought <ORG>FbK</ORG>.
Why simple things would not work?

- Capitalization is a strong indicator for capturing proper names, but it can be tricky because:
  - nouns in German are capitalized
  - first word of a sentence is capitalized
  - in nested named entity 
    - University of Southern California is Organization
  - sometimes titles in web pages are all capitalized

- Currently, no gazetteer contains all existing proper names.

- New proper names constantly emerge
  - movie titles, books, singers etc.

Why simple things would not work?

- The same entity can have multiple variants of the same proper name
  - Beyonce
  - Beyonce Knowles
  - B

- Proper names are ambiguous
  - Jordan the person vs. Jordan the location
  - JFK the person vs. JFK the airport
  - May the person vs. May the month

- Proper names have abbreviations and acronyms
  - Information Sciences Institute and ISI
Knowledge NER vs. Learning NER

**Knowledge Engineering**
- very precise (hand-coded rules)
- small amount of training data
- expensive development & test cycle
- domain dependent
- changes over time are hard

**Learning Systems**
- higher recall
- no need to develop grammars
- developers do not need to be experts
- annotations are cheap
- require lots of training data

---

**Rule Based NER (1)**

- **Create regular expressions:** a set of pattern matching rules encoded in a string according to certain syntax rules.

Suppose you are looking for a word that:

1. starts with a capital letter “P”
2. is the first word on a line
3. the second letter is a lower case letter
4. is exactly three letters long
5. the third letter is a vowel

The regular expression would be “^P[a-z][aeiou]” where

^ - indicates the beginning of the string
[a-z] – any letter in range a to z
[aeiou] – any vowel
Perl RegEx

- \w (word char) any alpha-numeric
- \d (digit char) any digit
- \s (space char) any whitespace
- . (wildcard) anything
- \b word boundary
- ^ beginning of string
- $ end of string
- ? For 0 or 1 occurrences
- + for 1 or more occurrences
- specific range of number of occurrences: {min,max}.
  - A{1,5} One to five A’s.
  - A{5,} Five or more A’s
  - A{5} Exactly five A’s

Rule Based NER (1)

- Create regular expressions
  - E-mail
  - Capitalized names
  - Telephone number

blocks of digits separated by hyphens

RegEx = (\d+\-\d+)+

- matches valid phone numbers like 900-865-1125 and 725-1234
- incorrectly extracts social security numbers 123-45-6789
- fails to identify numbers like 800.865.1125 and (800)865-CARE

Improved RegEx = (\d{3}\-\d{3}\-\d{4})\[(\d[A-Z]{4})\]
Rule Based NER (2)

- Create rules like
  - Capitalized word + {city, center, river} indicates location
    Ex. New York city
    Hudson river
  - Capitalized word + {street, boulevard, avenue} indicates location
    Ex. Fifth avenue

Rule Based NER (3)

- Use context patterns
  - [PERSON] earned [MONEY]
    Ex. Frank earned $20
  - [PERSON] joined [ORGANIZATION]
    Ex. Sam joined IBM
  - [PERSON],[JOBTITLE]
    Ex. Mary, the teacher
  
still not so simple:
  - [PERSON][ORGANIZATION] fly to [LOCATION][PERSON][EVENT]
    Ex. Jerry flew to Japan
    Sarah flies to the party
    Delta flies to Europe
Machine Learning NER

- NED: Identify named entities using BIO scheme
  - B beginning of an entity
  - I continues the entity
  - O word outside the entity

- NEC: Classify into a predefined set of categories
  - Person names
  - Organizations (companies, governmental organizations, etc.)
  - Locations (cities, countries, etc.)
  - Miscellaneous (movie titles, sport events, etc.)
Learning for Categorization

• A training example is an instance $x \in X$, paired with its correct category $c(x)$: $\langle x, c(x) \rangle$ for an unknown categorization function, $c$.

• Given:
  – A set of training examples, $T$.
  – A hypothesis space, $H$, of possible categorization functions, $h(x)$.

• Find a consistent hypothesis, $h(x) \in H$, such that:

$$\forall \langle x, c(x) \rangle \in T : h(x) = c(x)$$

$k$ Nearest Neighbor

• Learning is just storing the representations of the training examples.

• Testing instance $x_p$:
  – compute similarity between $x_p$ and all training examples
  – take vote among $x_p$ $k$ nearest neighbours
  – assign $x_p$ with the category of the most similar example in $T$
Distance measures

• Nearest neighbor method uses similarity (or distance) metric.

• Given two objects $x$ and $y$ both with $n$ values

\[ x = (x_1, x_2, \ldots, x_n) \]
\[ y = (y_1, y_2, \ldots, y_n) \]

calculate the Euclidean distance as

\[ d(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \]

An Example

<table>
<thead>
<tr>
<th></th>
<th>isPersonName</th>
<th>isCapitalized</th>
<th>isLiving</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jerry Hobbs</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>USC</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Euclidean distance:

\[ d(JerryHobbs, USC) = \sqrt{1^2 + 0^2 + 1^2} = 1.41 \]
1-Nearest Neighbor

3-Nearest Neighbor

choose the category of the closer neighbor (can be erroneous due to noise)

choose the category of the majority of the neighbors
5-Nearest Neighbor

the value of $k$ is typically odd to avoid ties

$k$ Nearest Neighbours

**Pros**
- robust
- simple
- training is very fast (storing examples)

**Cons**
- depends on similarity measure & k-NNs
- easily fooled by irrelevant attributes
- computationally expensive
Decision Trees

- The classifier has a tree structure, where each node is either:
  - a leaf node which indicates the value of the target attribute (class) of examples
  - a decision node which specifies some test to be carried out on a single attribute-value, with one branch and sub-tree for each possible outcome of the test

- An instance \( x_p \) is classified by starting at the root of the tree and moving through it until a leaf node is reached, which provides the classification of the instance

<table>
<thead>
<tr>
<th></th>
<th>isPersonName</th>
<th>isCapitalized</th>
<th>isLiving</th>
<th>X is PersonName?</th>
</tr>
</thead>
<tbody>
<tr>
<td>profession</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>NO</td>
</tr>
<tr>
<td>Jerry Hobbs</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>YES</td>
</tr>
<tr>
<td>USC</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>NO</td>
</tr>
<tr>
<td>Jordan</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>NO</td>
</tr>
</tbody>
</table>

An Example

Each internal node tests an attribute

Each branch corresponds to an attribute value node

Each leaf node assigns a classification
Building Decision Trees

• Select which attribute to test at each node in the tree.

• The goal is to select the attribute that is most useful for classifying examples.

• Top-down, greedy search through the space of possible decision trees. It picks the best attribute and never looks back to reconsider earlier choices.

Decision Trees

<table>
<thead>
<tr>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ generate understandable rules</td>
<td>- error prone in multi-class classification and small number of training examples</td>
</tr>
<tr>
<td>+ provide a clear indication of which features are most important for classification</td>
<td>- expensive to train due to pruning</td>
</tr>
</tbody>
</table>
Carreras et al. 2002

- Learning algorithm: AdaBoost
- Binary classification
- Binary features

\[ f(x) = \sum_{t=1}^{T} \alpha_t h_t(x) \]  
(Schapire & Singer, 99)
- Weak rules (h_t): Decision Trees of fixed depth.

Features for NE Detection

- **Contextual**
  - current word W_0
  - words around W_0 in [-3,...,+3] window

- **Part-of-speech tag** (when available)

- **Orthographic (binary and not mutually exclusive)**

  - initial-caps
  - roman-number
  - acronym
  - single-char
  - all-caps
  - contains-dots
  - lonely-initial
  - functional-word*
  - all-digits
  - contains-hyphen
  - punctuation-mark
  - URL

- **Word-Type Patterns**

  - functional
  - capitalized
  - lowercased
  - punctuation mark
  - quote
  - other

- **Left Predictions**
  - the tag predicted in the current classification for W-3, W-2, W-1

*functional-word is preposition, conjunction, article
Results for NE Detection

<table>
<thead>
<tr>
<th>CoNLL-2002 Spanish Evaluation Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data sets</td>
</tr>
<tr>
<td>Train</td>
</tr>
<tr>
<td>Development</td>
</tr>
<tr>
<td>Test</td>
</tr>
</tbody>
</table>

Evaluation Measures

\[
\text{Precision} = \frac{\text{# correct identified NEs}}{\text{# identified NEs}}
\]

\[
\text{Recall} = \frac{\text{# correct identified NEs}}{\text{# gold standard data}}
\]

<table>
<thead>
<tr>
<th>Carreras et al., 2002</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIO dev.</td>
<td>92.45</td>
<td>90.88</td>
<td>91.66</td>
</tr>
</tbody>
</table>

Features for NE Classification (1)

- **Contextual**
  - current word \( W_0 \)
  - words around \( W_0 \) in [-3,...,+3] window

- **Part-of-speech tag** (when available)

- **Bag-of-Words**
  - words in [-5,...,+5] window

- **Trigger words**
  - for person (\( Mr, Miss, Dr, PhD \))
  - for location (\( city, street \))
  - for organization (\( Ltd., Co. \))

- **Gazetteers**
  - geographical
  - first name
  - surname
Features for NE Classification (2)

- Length in words of the entity being classified
- Pattern of the entity with regard to the type of constituent words
- **For each class**
  - whole NE is in gazetteer
  - any component of the NE appears in gazetteer
- **Suffixes** (length 1 to 4)
  - each component of the NE
  - whole NE

Results for NE Classification*

<table>
<thead>
<tr>
<th>Spanish</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>79.04</td>
<td>80.00</td>
<td>79.52</td>
</tr>
<tr>
<td>MISC</td>
<td>55.48</td>
<td>54.61</td>
<td>55.04</td>
</tr>
<tr>
<td>ORG</td>
<td>79.57</td>
<td>76.06</td>
<td>77.77</td>
</tr>
<tr>
<td>PER</td>
<td>87.19</td>
<td>86.91</td>
<td>87.05</td>
</tr>
<tr>
<td><strong>overall</strong></td>
<td>79.15</td>
<td>77.80</td>
<td><strong>78.47</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Spanish</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOC</td>
<td>85.76</td>
<td>79.43</td>
<td>82.47</td>
</tr>
<tr>
<td>MISC</td>
<td>60.19</td>
<td>57.35</td>
<td>58.73</td>
</tr>
<tr>
<td>ORG</td>
<td>81.21</td>
<td>82.43</td>
<td>81.81</td>
</tr>
<tr>
<td>PER</td>
<td>84.71</td>
<td>93.47</td>
<td>88.87</td>
</tr>
<tr>
<td><strong>overall</strong></td>
<td>81.38</td>
<td>81.40</td>
<td><strong>81.39</strong></td>
</tr>
</tbody>
</table>

System of Carreras et al., 2002
Homework
Named Entity Challenge

• **Given:** a train and development set of English sentences tagged with four named entity classes:
  – PER (people)
  – ORG (organization)
  – LOC (location)
  – MISC (miscellaneous)

• **Your objective is:** to develop a machine learning NE system, which when given a new previously unseen text (i.e. test set) will identify and classify the named entities correctly
Data Description

- The data consists of two columns separated by a single space. Each word has been put on a separate line and there is an empty line after each sentence.

Timeline

<table>
<thead>
<tr>
<th>Event</th>
<th>Date/Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Release</td>
<td></td>
</tr>
<tr>
<td>Train and Development data</td>
<td>March 24th 2010</td>
</tr>
<tr>
<td>Test data</td>
<td>April 9th 2010</td>
</tr>
<tr>
<td>Result submission deadline</td>
<td>April 10th 2010 (11:59 pm)</td>
</tr>
<tr>
<td></td>
<td>later submissions will not be accepted</td>
</tr>
<tr>
<td>Presentation submission deadline</td>
<td>April 13th 2010</td>
</tr>
</tbody>
</table>
Submit (1)

• The source code for the feature generation
  (make sure it will run under Linux)

• The official train and test feature files used in the
  final run, together with the final output of your
  system for the test data

• The additionally generated resources (if any)

• Write 1-2 page brief description of your approach
  explaining:
    – the used NLP tools
    – the designed features
    – the employed machine learning algorithm

Submit (2)

• Make a short power point presentation which you
  will present in 3 minutes to the class on April 15th.

• Please, be prompt so I can include your slides in
  the set to be presented

• Note you will have maximum 3 minutes to present
  your work in class, make sure your presentation is
  to the point
**Evaluation is based on**

- the ranking of your system against the rest
- the designed features
  - novel, previously unknown features will be favored
  - system’s pre or post processing
  - a study on the groups of features used
- the generated resources
  - size, methods and sources for gazetteer extraction
  - trigger lists

**Generate Your Own Resources**

- Extract gazetteers from Wikipedia
  - People (singers, teachers, mathematicians etc.)
  - Locations (cities, countries)
  - Organizations (universities, IT companies etc.)
- Extract trigger words from WordNet
  - look for hyponyms of person, location, organization
- Extract and rank the patterns in which the NEs occurred in the train and development data. Show what percentages of these were found in the final test data.
- Extract lists of verbs found next to the NEs. Do you find any similarity/regularity of the verbs associated with each one of the NE categories?
What must I do …

• Use the train and development data to design and tune your NE system

• Decide on the features you would like to incorporate in your NE system

• Choose a machine learning classifier from Weka
  • Intro by Marti Hearst
    [http://courses.ischool.berkeley.edu/i256/f06/lectures/lecture16.ppt](http://courses.ischool.berkeley.edu/i256/f06/lectures/lecture16.ppt)

• This is a big assignment so start early!

---

**WEKA GUI Chooser**

```
java -Xmx1000M -jar weka.jar
```
WEKA File Format: ARFF

@relation english_named_entity

@attribute position numeric
@attribute pos_tag { NN, NP, VB, DT}
@attribute word_length numeric
@attribute in_gazetteer { no, yes}
@attribute class { PER, LOC, ORG, MISC}

@data
3,DT,3,no,ORG
4,NP,10,yes,ORG
15,NP,6,yes,PER
7,NN,12?,MISC
...

Other attribute types:
- String
- Date

The Preprocessing Tab

- Classification
- Manual attribute selection
- Statistical attribute selection
- Filter selection
- List of attributes (last: class variable)
- Frequency and categories for the selected attribute
- Statistics about the values of the selected attribute
Choice of classifier

The attribute whose value is to be predicted from the values of the remaining ones. Default is the last attribute.

Cross-validation: split the data into e.g. 10 folds and 10 times train on 9 folds and test on the remaining one.

Choosing a classifier
all other numbers can be obtained from it

accuracy
different/easy class

Running on Test Set
Available Resources

- **WordNet** [http://wordnet.princeton.edu/](http://wordnet.princeton.edu/)
- **Part-of-speech taggers**
  - TreeTagger [http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html](http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html)
- **NP chunker**
  - [http://www.dcs.shef.ac.uk/~mark/index.html](http://www.dcs.shef.ac.uk/~mark/index.html)
- **Parser**
- **Named Entity Recognizer**
- **Other**

Good Luck!