Information extraction: Named Entities

Vivi Nastase

with material from Marius Pasca’s CIKM-2011 tutorial on IE
Summer semester 2012, ICL, University of Heidelberg
Machine learning – roughly

(a) Training

- Label
- Input → Feature extractor → Features → Machine learning algorithm

(b) Prediction

- Input → Feature extractor → Features → Classifier model → Label
Evaluation (most frequently)

<table>
<thead>
<tr>
<th>Actual classification</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>positive</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true positive (tp)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false positive (fp)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>negative</td>
<td></td>
<td></td>
</tr>
<tr>
<td>false negative (fn)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>true negative (tn)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Precision**  
\[ P = \frac{TP}{TP + FP} \]

**Recall**  
\[ R = \frac{TP}{TP + FN} \]

**Accuracy**  
\[ A = \frac{TP + TN}{TP + FP + FN + TN} \]

**F-measure**
\[ F = \frac{(1 + \beta^2)PR}{\beta^2 P + R} \]
\[ F = \frac{PR}{(1-\alpha)P + \alpha(R)} ; \alpha = \frac{1}{1+\beta^2} \]

Most commonly used:
\[ \beta = 1 \rightarrow F1 = \frac{2PR}{P + R} \]
Evaluation – correlation

Pearson’s correlation coefficient
\[ r = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \]

Spearman’s rank correlation
\[ \rho = \frac{\sum_{i}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i}(x_i - \bar{x})^2} \sum_{i}(y_i - \bar{y})^2} \]

\( x_i, y_i \) – ranks of \( X_i, Y_i \)
Dr. Michael Jordan of the University of California Berkeley presents "Machine Learning from an Nonparametric Bayesian Point of View" March 27, 2008.
## Named entity recognition

### Phrase classification
- Predefine a set of entity types (Person, Organization, Location, ...)
- Classify each phrase into one of the entity types + non-entity
- Models = sets of extraction patterns

### Token classification
- Classify = tagging each token as Inside or Outside a named entity
- Extract contiguous tokens tagged I as named entities
NER with HMMs

Markov assumption: \( P(x_i|x_{i-1}...x_1) = P(x_i|x_{i-1}) \)

Maximize \( P(X|O, \lambda) \)

- \( X = x_1...x_T \) – sequence of hidden variable values
- \( O = o_1...o_T \) – observations
- \( Q = \{q_1, ..., q_N\} \) – possible states

\( \lambda = (A, B) \)

- \( A (N \times N) \)
  - \( a_{ij} = p(x_i|x_j) \) transition probabilities
  - \( a_{0j} = p(x_j) \) initial state probabilities
  - \( \sum_{j=1}^{N} a_{ij} = 1 \ \forall i \)

- \( B (T) : b_i(o_k) = p(o_k|x_i) \) emission probabilities
Sequence labeling with HMMs

The Viterbi Algorithm

\[ v_1(j) = a_{0j}b_j(o_1) \quad j = 1, N \]
\[ v_t(j) = \max_i v_{t-1}(i)a_{ij}b_j(o_t) \quad j = 1, N \]
\[ back(j) = \arg\max_i v_{t-1}(i)a_{ij}b_j(o_t) \]
Forward variable

Forward variable $= \text{probability of being in state } j \text{ after the first } t \text{ observations}$

\[
\alpha_t(j) = \mathbb{P}(o_1, \ldots, o_t, q_t = j | \lambda)
\]

\[
\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)
\]
Backward variable

Backward variable $= \text{probability of seeing the observations } o_{t+1}, \ldots, o_T \text{ given that the state at time } t \text{ is } i$

$$\beta_t(i) = P(o_{t+1}, \ldots, o_T | q_t = i, \lambda)$$
$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$
Learning an HMM – $\lambda = (A, B)$

Forward-backward algorithm

**initialize** $A$, $B$

**iterate** until convergence

**E-step** $\forall i, j, t$

$$\gamma_t(j) = P(q_t = j | O, \lambda) = \frac{\alpha_t(j) \beta_t(j)}{P((O) | \lambda)}$$

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | O, \lambda) = \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(N)}$$

**M-step**

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{j=1}^{N} \xi_t(i,j)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1}^{T} s.t. o_t = v_k \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

**return** $A$, $B$
IOB sequence labeling with HMMs

- tokenize text
- split text into sentences (our sequences)
- hidden variable possible values: I, O, B
- estimate $\lambda$ from an annotated corpus

$$a_{ij} = p(x_i| x_j) = \frac{c(x_i, x_j)}{c(x_j)}$$

$$b_j(o_t) = \frac{c(x_j, o_t)}{c(x_j)}$$

or learn $\lambda$ using the forward-backward algorithm
Shortcomings of HMMs

- the output independence assumption – observations are independent from each other
- no general information (e.g. capitalization, POS)
- the inferred sequence of labels maximizes the likelihood
  \[ X = \text{argmax}_X P(O|X) \]
Markov Random Fields

undirected graphical models $G = (V, E)$

- $V$ = a set of vertices (corresp. to random variables)
- $E$ = a set of undirected edges (corresp. to dependencies)
- $N(V_i)$ = the set of neighbours of vertex $V_i \in V$

Markov Random Field

$\forall V_i \in V, P(V_i|V - V_i) = P(V_i|N(V_i))$

Conditional Markov Random Field

$V = X \cup Y$

- $X$ = set of observed variables
- $Y$ = set of hidden variables

$\forall Y_i \in Y, P(Y_i|X, Y - Y_i) = P(Y_i|X, N(Y_i))$
Learning with Conditional Random Fields

\[ P_{\Lambda}(x|o) = \frac{1}{Z_o} e^{\sum_{t=1}^{T} \sum_k \lambda_k f_k(x_{t-1}, x_t, o, t)} \]

\[ Z_o = \sum_{x \in X^T} e^{\sum_{t=1}^{T} \sum_k \lambda_k f_k(x_{t-1}, x_t, o, t)} \] – normalization factor

\[ f_k(x_{t-1}, x_t, o, t) \] – feature function

\[ \Lambda = \{ \lambda_1, ..., \lambda_K \} \]

\[ \lambda_k \] – learned weight for each feature function
Computing $Z_o$ 

Forward-backward algorithm with:

$$\alpha_{t+1}(x) = \sum_{x'} \alpha_t e^{\sum_k \lambda_k f_k(x', x, o, t)}$$

$$Z_o = \sum_x \alpha_T(x)$$
Estimate feature weights

Training data

\[ \mathcal{D} = \{(o^{(l)}, y^{(l)})\}_{l=1}^{M} \]
\[ o^{(l)} = (o_1^{(l)}, \ldots, o_T^{(l)}) \]
\[ y^{(l)} = (y_1^{(l)}, \ldots, y_T^{(l)}) \]

Conditional log-likelihood

\[ \hat{\Lambda} = \arg\max_{\Lambda} P_\Lambda(y|x) \]
\[ = \arg\max_{\Lambda} \log P_\Lambda(y|x) \]
\[ = \arg\max_{\Lambda} \sum_{l=1}^{M} \log P_\Lambda(y^{(l)}|x^{(l)}) \]
\[ = \ldots \]
\[ = \arg\max_{\Lambda} \sum_{l=1}^{M} \sum_{l=1}^{M} \log P_\Lambda(y^{(l)}|x^{(l)}) - \sum_{j=1}^{N} \frac{w_j^2}{2\sigma_j^2} \]
Features for NER with CRFs

begins-with-number
begins-with-punctuation
begins-with-question-word
begins-with-subject
blank
contains-alphanum

contains-question-mark
contains-question-word
ends-with-question-mark
first-alpha-is-capitalized
indented
only-punctuation

...
Named entity disambiguation

Bunescu & Pasca, 2006 *Using encyclopedic knowledge for named entity disambiguation*
Michael Jordan and his family moved from Brooklyn to Wilmington, N.C., when he was a small child and his famed basketball career has taken flight from ...

Michael Jeffrey Jordan (born February 17, 1963) is a retired American professional basketball player, active entrepreneur, and majority owner of the Charlotte Bobcats.

Dr. Michael Jordan of the University of California Berkeley presents "Machine Learning from an Nonparametric Bayesian Point of View" March 27, 2008.
Disambiguation approach

- Build a dictionary $D$ of named entities
  - large scale – Wikipedia
  - map each name $d \in D$ to the set of entities $d.E$ in Wikipedia it can refer to
- Supervised disambiguation method:
  - detection – detect when a proper name refers to a named entity
  - disambiguation – find the correct referent given the context
Names in Wikipedia

- Assumption: Wikipedia articles describe concepts
- Names for Wikipedia articles:
  - article titles
  - redirect links
  - disambiguation articles
  - anchor texts of hyperlinks
Notations

\( e \) – entity (Wikipedia article)

\( e.title \) – title name (e.g. Michael Jordan (mycologist))

\( e.name \) – clean name (e.g. Michael Jordan)

\( e.T \) – text of the article (e.g. United States)

\( e.R \) – set of all names that redirect to \( e \) (e.g. USA, US, ...)

\( e.D \) – set of names whose disambiguation page contains a link to \( e \) (e.g. US, America, ...)
Named entity dictionary from Wikipedia

Collect all entity names that satisfy one of:

1. \( e.title \) is a multi-word term, and all content words are capitalized (e.g. *The Witches of Eastwick*)
2. \( e.title \) is a one-word term which contains at least 2 capital letters (e.g. NATO)
3. at least 75% of the title occurrences inside the article are capitalized
Notation

\(d \in D\) – a proper name entry in the dictionary \(D\)
\(d.E\) – the set of entities whose name may be \(d\)
\(e \in d.E \leftrightarrow e.name \lor d \in e.R \lor d \in e.D\)
Disambiguation training data

The [[Vatican City|Vatican]] is now an enclave surrounded by [[Rome]].

**Notation**

- $q.E$ – the set of entities associated with the query $q$ in $D$
- $q.e \in q.E$ – the true entity associated with $q$
- $q.T$ – the text within a window of size $n$ centered on $q$’s hyperlink.
Disambiguation through ranking

- a scoring function that computes the compatibility between $q$ and any of the potential referents $e_k$:

$$score(q, e_k) = w\phi(q, e_k) \quad \phi = [\phi_{\text{cos}}|\phi_{w,c}|\phi_{\text{out}}]$$

  - context-article similarity:
    $$\phi_{\text{cos}}(q, e_k) = \cos(q.T, e_k.T) = \frac{q.T \cdot e_k.T}{\|q.T\| \cdot \|e_k.T\|}$$
    $\forall w \in q.T$ or $w \in e_k.T : d_w = f(w)\ln\frac{N}{df(w)}$

  - category score, for each Wikipedia word $w$ and category $C$ pair:
    $$\phi_{w,c}(q, e_k) = \begin{cases} 
    1 & \text{if } w \in q.T, c \in e_k.C \\
    0 & \text{otherwise}
    \end{cases}$$

  - special feature for out-of-Wikipedia entities:
    $$\phi_{\text{out}}(q, e_k) = \delta(e_k, e_{\text{out}})$$

- for an instance $q$, select $e = \arg\max_{e_k \in q.E} score(q, e_k)$
Disambiguation – system overview

**Training**

- **Disambiguation Dataset** → **Ranking Examples features** $\phi(q, e_k)$ → **SVM training** → **Ranking Model weights $w$**

**Testing**

- **NE query $q$** → **Ranking Instances features** $\phi(q, e_k)$ → **Answer:**
  $$\arg\max_{e_k \in q_E} w \cdot \phi(q, e_k)$$

- **NE Dictionary** → **Ranking Model weights $w$**
Extracting information from open texts

Marius Pasca – CIKM tutorial on Information Extraction, 2011
Sources of open-domain information

Human compiled knowledge resources
- created by experts
- created collaboratively by non-experts

Sources of textual data
- text document (various degrees of structure)
- (Web) search queries
Expert-built resources

**WordNet**  C. Fellbaum, 1998 *An Electronic Lexical Database*
- lexical database of English
- various extensions (languages, domains, sentiment)
- hypernym/hyponym and meronym/holonym hierarchies (and other relations)
- 155,000+ words / 117,000+ synsets

**Cyc**  D. Lenat, 1995 *CYC: A large-scale investment in knowledge infrastructure*
- knowledge base of common-sense and encyclopedic knowledge
- concepts and relations organized in hierarchies
- 300,000+ concepts / 3+ million assertions
Collaborative non-expert resources

**Open Mind**  P. Singh et al., 2002 *Open Mind Common Sense: Knowledge acquisition from the General Public*
- collect (common sense) knowledge in (simple) natural language
- 800,000+ facts in English

**MindPixel**  knowledge base of millions of true/false or probabilistic propositions

**Wikipedia**  M. Remy, 2002 *Wikipedia: The free encyclopedia*
- ≈ 4 million articles in English
- versions in 200+ languages
Collaborative non-expert resources

**DBpedia** C. Bizer et al., 2009 *DBpedia – A crystallization point for the web of data*
- converts information from Wikipedia’s databases
- mappings from subset of Wikipedia infoboxes to ontology
- mappings from Wikipedia articles to WordNet
- 2.5+ million instances / 250+ million relations

**YAGO** Suchanek et al., 2007 *YAGO – A core of semantic knowledge*
- semantic knowledge base derived from Wikipedia, WordNet, GeoNames
- 10+ million entities / 120+ million facts

**Freebase** K. Bollacker et al., 2008 *Freebase: A Collaboratively created graph database for structuring human knowledge*
- repository for storing structured data from Wikipedia, other sources, and additional user contributions
- 20+ million instances / 300+ million instances
Sources of textual data: documents
Sources of textual data: queries

Query logs:
- requests capture knowledge that the users already have
- the answers to requests capture knowledge the users don’t yet have
- short length (2-3 words)
- low quality
- self-contained
Challenges in open-domain extraction

scale – large text collections
- time-efficient algorithms
- shallow processing is preferred

diversity – fine-grained classes of instances/relations

uncertainty – use redundancy as a proxy for trustworthiness
Extraction methods

Methods for extraction of:

- concepts and instances
  - flat sets of unlabeled instances
  - flat sets of labeled instances
  - conceptual hierarchies

- relations and attributes
  - for flat concepts
  - for building ontologies
Extraction from web documents

- start with seed instances
- submit queries, fetch Web documents with seed instances
- construct patterns for identifying more candidate instances
- rank candidate instances
Extraction from web documents

(Courtesy R. Wang)
Pantel & Pennacchiotti, 2006 *Entity extraction via Ensemble semantics*

- target classes, each with its own set of instances
- combine multiple data sources: web documents, queries, HTML tables, articles from Wikipedia
# Features

<table>
<thead>
<tr>
<th>Family</th>
<th>Type</th>
<th>Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web ($w$)</td>
<td>Frequency</td>
<td>term frequency; document frequency; term frequency as noun phrase</td>
</tr>
<tr>
<td></td>
<td>Confidence</td>
<td>confidence score returned by $KE_{pat}$; pmi with the 100 most reliable</td>
</tr>
<tr>
<td></td>
<td>Score</td>
<td>patterns used by $KE_{pat}$</td>
</tr>
<tr>
<td></td>
<td>Distribution</td>
<td>distributional similarity with the centroid in $KE_{dis}$; distributional</td>
</tr>
<tr>
<td></td>
<td>Termness</td>
<td>ratio between term frequency as noun phrase and term frequency; pmi</td>
</tr>
<tr>
<td></td>
<td></td>
<td>between internal tokens of the instance; capitalization ratio</td>
</tr>
<tr>
<td>Query log ($q$)</td>
<td>Frequency</td>
<td>number of queries matching the instance; number of queries containing</td>
</tr>
<tr>
<td></td>
<td>Co-occurrence</td>
<td>the instance query log pmi with any seed in $S$</td>
</tr>
<tr>
<td></td>
<td>Pattern</td>
<td>pmi with a set of trigger words $T$ (i.e., the 10 words in the query</td>
</tr>
<tr>
<td></td>
<td>Distribution</td>
<td>logs with highest pmi with $S$</td>
</tr>
<tr>
<td></td>
<td>Termness</td>
<td>distributional similarity with $S$ (vector coordinates consist of the</td>
</tr>
<tr>
<td></td>
<td></td>
<td>instance’s pmi with the words in $T$</td>
</tr>
<tr>
<td>Web table ($t$)</td>
<td>Frequency</td>
<td>ratio between the two frequency features $F'$</td>
</tr>
<tr>
<td></td>
<td>Co-occurrence</td>
<td>table frequency</td>
</tr>
<tr>
<td></td>
<td></td>
<td>table pmi with $S$; table pmi with any seed in $S$</td>
</tr>
<tr>
<td>Wikipedia ($k$)</td>
<td>Frequency</td>
<td>term frequency</td>
</tr>
<tr>
<td></td>
<td>Co-occurrence</td>
<td>pmi with any seed in $S$</td>
</tr>
<tr>
<td></td>
<td>Distribution</td>
<td>distributional similarity with $S$</td>
</tr>
</tbody>
</table>

(Courtesy P. Pantel, M. Pennacchiotti)
Extracting NEs from query logs

Pasca, 2007 *Weakly-supervised discovery of named entities using Web search queries*

- **Input**
  - target classes, as sets of seeds: e.g. for Company – Honda, Oracle, Reuters, ...

- **Data** – anonymized search queries and their frequencies

- **Output** – ranked list of class instances
Extracting NEs from query logs

Query logs
- side effects of generic birth control pills
- can low blood pressure make you tired
- prescription vicodin online
- long term xanax use
- propanolol and vicodin interaction
- causes of low blood pressure during pregnancy
- taking lipitor during pregnancy
- buy xanax in uk
- taking beta blockers during pregnancy
- how oxycontin works
- long term lamictal use
- propanolol and lamictal interaction
- is xanax habit forming
- can lamictal be crushed
- how does lipitor work
- effect of beta blockers on exercise
- does vioxx cause weight gain
- can phenetamine make you tired
- buy vioxx in uk
- side effects of viagra pills
- long term vioxx use
- prescription lamictal online
- long term xanax use
- what effects does low blood pressure have
- buy lipitor online

Target classes (seed instances)
- Drug: {phenetamine, viagra, vicodin, vioxx, xanax}

Query templates matching seed instances
- [can] [make you tired] [prefix] [suffix]
- [buy] [in uk] [prefix] [suffix]
- [does] [cause weight gain] [prefix] [suffix]
- [side effects of] [pills] [prefix] [suffix]
- [prescription] [online] [prefix] [suffix]

Pool of candidate instances
- Drug: {low blood pressure, xanax, lamictal, generic birth control, lipitor, vicodin, beta blockers, ...}

Search–signature vectors (one per candidate instance)
- Drug: xanax
- [buy] [prefix] [in uk] [suffix]
- [long term] [prefix] [use] [suffix]

- Drug: low blood pressure
- [causes of] [prefix] [during pregnancy] [suffix]
- [can] [make you tired] [prefix] [suffix]

- Drug: vioxx
- [how does] [prefix] [work] [suffix]
- [does] [cause weight gain] [prefix] [suffix]

- Drug: lamictal
- [can] [be crushed] [prefix] [suffix]
- [long term] [prefix] [use] [suffix]

Reference search–signature vectors (one per class)
- Drug
- [buy] [prefix] [in uk] [suffix]
- [how does] [prefix] [work] [suffix]
- [does] [cause weight gain] [prefix] [suffix]
- [long term] [prefix] [use] [suffix]
**Vector representation**

- signature-vector for candidate instance – aggregates all (weighted) prefixes and postfixes for the instance
- signature-vector for the class – adds the signature vectors for all instances
- rank each candidate instance based on its signature-vector similarity to the class signature vector:
  - Jenses-Shannon divergence
    \[
    JSD(P \parallel Q) = \frac{1}{2} D(P \parallel M) + \frac{1}{2} D(Q \parallel M)
    \]
  - where \( M = \frac{1}{2}(P + Q) \)
  - Kullback-Leibler divergence
    \[
    D_{KL}(P\parallel Q) = \sum_i P(i) \ln \frac{P(i)}{Q(i)}
    \]
## Extracted patterns

<table>
<thead>
<tr>
<th><strong>Country:</strong></th>
<th><strong>Drug:</strong></th>
<th><strong>VideoGame:</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>what type of government does I have</td>
<td>how long does I stay in your system</td>
<td>how many copies of I have been sold</td>
</tr>
<tr>
<td>what do people in I eat</td>
<td>can I be crushed</td>
<td>how much does I cost</td>
</tr>
<tr>
<td>what is the weather like in I in March</td>
<td>what does the I pill look like</td>
<td>where can I play I online for free</td>
</tr>
<tr>
<td>how to apply for I visa</td>
<td>how much does I cost</td>
<td>how to install I mods</td>
</tr>
<tr>
<td>how did I gain independence</td>
<td>how does I affect the heart</td>
<td>when is I coming out on gamecube</td>
</tr>
<tr>
<td>where is I on the map</td>
<td>how is I manufactured</td>
<td>how many I levels</td>
</tr>
<tr>
<td>what continent is I on</td>
<td>does I cause weight gain</td>
<td>what copy protection does I use</td>
</tr>
<tr>
<td>what is I’s currency</td>
<td>when was I fda approved</td>
<td>why does I crash</td>
</tr>
<tr>
<td>why did I join the eu</td>
<td>can I make you tired</td>
<td>how to add bots to I server</td>
</tr>
<tr>
<td>why is I poor</td>
<td>what I is made out of</td>
<td>who made I 2</td>
</tr>
</tbody>
</table>
Extracting instances within labeled concepts
Extracting instances within labeled concepts

**Rule template**

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Class1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>NP1 <em>such as</em> NP2</td>
</tr>
<tr>
<td>Constraints</td>
<td>head(NP1) = plural(label(Class1)) &amp; properNoun(head(NP2))</td>
</tr>
<tr>
<td>Bindings</td>
<td>Class1(head(NP2))</td>
</tr>
</tbody>
</table>

**Extraction rule**

<table>
<thead>
<tr>
<th>Predicate</th>
<th>Car maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pattern</td>
<td>NP1 <em>such as</em> NP2</td>
</tr>
<tr>
<td>Constraints</td>
<td>head(NP1) = car makers &amp; properNoun(head(NP2))</td>
</tr>
<tr>
<td>Bindings</td>
<td>Car maker(head(NP2))</td>
</tr>
<tr>
<td>Keywords</td>
<td><em>car makers such as</em></td>
</tr>
</tbody>
</table>
Extraction loop

Extractor
- convert rules to keyword-based queries
- submit queries to search engine
- apply rules to retrieved Web documents

Assessor
- estimate probability of correctness for each extraction
- compute association strength between extracted instances (e.g. *Kuala Lumpur* for *City*) and “discriminator” phrases (e.g. *city*)

Bootstrapping
- convert rule templates into extraction rules and discriminators
- select most productive extraction rules from previous iteration
- apply Extractor, then Assessor
- add candidate extractions to output
- repeat until all extraction rules and/or queries have been used