# Details of Two Unsupervised NE Learning Methods

- Unsupervised NE Classification
  - Michael Collins and Yoran Singer, 1999

- Unsupervised Learning of Generalized Names
  - Yangarber, Lin, Grishman, 2002
  - Lin, Yangarber, Grishman, 2003

# Unsupervised NE classification

based on Michael Collins and Yoran Singer, EMNLP 1999

- The task: to learn a decision list to classify strings as person, location or organization

The learned decision list is an ***ordered*** sequence of if-then rules

*… says Mr. Gates, founder of Microsoft …*

$R_1$ : if features then person
$R_2$ : if features then location
$R_3$ : if features then organization
…
$R_n$ : if features then person

*… says Mr. Gates, founder of Microsoft …*

# Outline of Unsupervised Co-Training

- Parse an unlabeled document set
- Extract each NP, whose head is tagged as proper noun
- Define a set of relevant features, which can be applied on extracted NPs
- Define two separate types of rules on basis of feature space
- Determine small initial set of seed rules
- Iteratively extend the rules through co-training

# Two Categories of Rules

- The key to the method is redundancy in the two kind of rules.

…says Mr. Cooper, a vice president of…

Paradigmatic or spelling

Syntagmatic or contextual

Huge amount of unlabeled data gives us these hints!

# The Data


- 971,746 New York Times sentences were parsed using full sentence parser.
- Extract consecutive sequences of proper nouns (tagged as NNP and NNPS) as named entity examples if they met one of following two criterion.
- Note: thus seen, NNP(S) functions as a generic NE-type, and the main task is now to sub-type it.

# Kinds of Noun Phrases

1. There was an appositive modifier to the NP, whose head is a singular noun (tagged NN).
   - *…says [Maury Cooper], [a vice president]…*

2. The NP is a complement to a preposition which is the head of a PP. This PP modifies another NP whose head is a singular noun.
   - *… fraud related to work on [a federally funded sewage plant] [in [Georgia]].*

# *(spelling, context) pairs created*

- *…says Maury Cooper, a vice president…*
  - (Maury Cooper, president)
- *… fraud related to work on a federally funded sewage plant in Georgia.*
  - *(Georgia, plant_in)*

# Features

## for representing examples for the learning algorithm

- Set of spelling features
  - Full-string=x (full-string=Maury Cooper)
  - Contains(x) (contains(Maury))
  - Allcap1 IBM
  - Allcap2 N.Y.
  - Nonalpha=x A.T.&T. (nonalpha=..&.)

---

- Set of context features
  - Context = x (context = president)
  - Context-type = x appos or prep

**It is strongly assumed that the features can be partitioned into two types such that each type alone is sufficient for classification**

# Examples of named entities and their features

| Sentence | Entities(Spelling/Context) | (Active) Features |
|---|---|---|
| But Robert Jordan, a partner at Steptoe & Johnson who took … | Robert Jordon/partner | Full-string=Robert_Jordan, contains(Robert), contains(Jordan), context=partner, context-type=appos |
| | Steptoe & Johnson/partner_at | Full-string=Steptoe_&_Johnson, contains(Steptoe), contains(&), contains(Johnson), nonalpha=& , context=partner_at, context-type=prep |
| By hiring a company like A.T.&T. … | A.T.&T./company_like | Full-string= A.T.&T., allcap2, nonalpha=..&. , context=company_like, context-type=prep |
| Hanson acquired Kidde Incorporated, parent of Kidde Credit, for … | Kidde Incorporated/parent | Full-string=Kidde_Incorporated, contains(Kidde), contains(Incorporated), context=parent, context-type=appos |
| | Kidde Credit/parent_of | Full-string=Kidde_Credit, contains(Kidde), contains(Credit), context=parent_of, context-type=prep |

# Rules

Two separate types of rules:
Spelling rules
Context rules

Feature → NE-type, h(Feature,NE-type)

h(x,y): the strength of a rule, defined as

$$\arg\max_{x,y} \frac{Count(x,y)+\alpha}{Count(x)+k\alpha}$$

where

$$Count(x) = \sum_{y \in Y} Count(x,y)$$

$\alpha$ is a smoothing parameter

$k$ = #NE-types

Is an estimate of the conditional probability of the NE-type given the feature, P(y|x)

The rules ordered according to their strengths h form a decision list: the sequence of rules are tested in order, and the answer to the ***first*** satisfied rule is output.

# 7 SEED RULES

Note: only one type of rules used as seed rules, and all NE-types should be covered

- Full-string = New York → Locat
- Full-string = California → Location
- Full-string = U.S. → Location
- Contains(Mr.) → Person
- Contains(Incorporated) → Organization
- Full-string=Microsoft → Organization
- Full-string=I.B.M. → Organization

# The Co-training algorithm

1. Set N=5 (max. # of rules of each type induced in each iteration)
2. **Initialize**: Set the spelling decision list equal to the set of seed rules. Label the training set using these rules.
3. Use these to get contextual rules. (x = feature, y = label)
    1. Compute h(x,y), and induce at most N * K rules
    2. all must be above some threshold $p_{min}$=0.95
4. Label the training set using the contextual rules.
5. Use these to get N*K spelling rules (same as step 3.)
6. Set spelling rules to seed plus the new rules.
7. If N < 2500, set N=N+5, and goto step 3.

8. Label the training data with the combined spelling/contextual decision list, then induce a final decision list from the labeled examples where all rules (regardless of strength) are added to the decision list.

# Example

- (IBM, company)
  - …IBM, the company that makes…
- (General Electric, company)
  - ..General Electric, a leading company in the area,…
- (General Electric, employer )
  - … joined General Electric, the biggest employer…
- (NYU, employer)
  - NYU, the employer of the famous Ralph Grishman,…

# Why Separate Spelling, Context Features?

Can use theory behind co-training to explain how algorithm works

Requirements:

1. Classification problem $f: X \rightarrow Y$
   1. $f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$ for $i = 1 \ldots m$
   2. $f_1(x_{1,i}) = f_2(x_{2,i})$ for $i = m+1 \ldots n$

   (softer criteria requires $f_1$ and $f_2$ to minimize the disagreements → similarity)
2. Can partition features X into 2 types of features $x = (x_1,x_2)$
3. Each type is sufficient for classification
4. $x_1,x_2$ not correlated to tightly (e.g., no deterministic function from $x_1$ to $x_2$)

$f_i$ must correctly classify labeled examples, and

must agree with each other on unlabeled ex.

Open question: best similarity function?

3. & 4. Say that features can be partitioned.

# The Power of the Algorithm

- Greedy method
  - At each iteration method increases number of rules
  - While maintaining a high level of agreement between spelling & context rules

For n= 2500:

1. The two classifiers give both labels on 49.2% of the unlabeled data
2. And give the *same* label on 99.25% of these cases

➢ The algorithm maximizes the number of unlabeled examples on which the two decision list agree.

# Evaluation

- 88,962 (spelling, context) pairs.
  - 971,746 sentences
- 1,000 randomly extracted to be test set.
- Location, person, organization, noise (items outside the other three)
- 186, 289, 402, 123 (- 38 temporal noise).
- Let $N_c$ be the number of correctly classified examples
  - Noise Accuracy: $N_c$ / 962
  - Clean Accuracy: $N_c$ /(962-85)

# Results

| Algorithm | Clean Accuracy | Noise Accuracy |
|---|---|---|
| Baseline | 45.8% | 41.8% |
| EM | 83.1% | 75.8% |
| Yarowsky 95 | 81.3% | 74.1% |
| Yarowsky Cautious | 91.2% | 83.2% |
| DL-CoTrain | 91.3% | 83.3% |
| CoBoost | 91.1% | 83.1% |

# Remarks

- Needs full parsing of unlabeled documents
  - Restricted language independency
  - Need linguistic sophistication for new types of NE
- Slow training
  - In each iteration, full size of training corpus has to be re-labeled
- DFKI extensions
  - Typed Gazetteers
  - Chunk parsing only
  - Integrated into a cross-language QA system

# Unsupervised Learning of Generalized Names

Yangarber, Lin, Grishman, Coling 2002 & Lin, Yangarber, Grishman, ICML 2003

- Much work on ML-NE focuses on classifying *proper names (PNs)*
  - Person/Location/Organization
- IE generally relies on domain-specific lexicon or *Generalized Names (GNs)*
  - Closer to terminology:
    single- or multi-word domain-specific expressions
- Automatic learning of GNs is an important first step towards truly adaptive IE
  - IE system that can automatically adapt itself to new domains

# How GNs differ from PNs

- Not necessary capitalized
  - tuberculosis
  - E. coli
  - Ebola haemorrhagic fever
  - Variant Creutzfeldt-Jacob disease
- Name boundaries are non-trivial to identify
  - “the four latest typhoid fever cases”
- Set of possible candidate names is broader and more difficult to determine
  - “<u>National Veterinary Services</u> Director <u>Dr. Gideon Bruckner</u> said no cases of <u>mad cow disease</u> have been in <u>South Africa</u>.”
- Ambiguity
  - E. coli : organism or disease
  - Encephalitis : disease or symptom

# NOMEN: the Learning Algorithm

1. Input: Seed names in several chosen categories
2. Tag occurrences of names
3. Generate local patterns around tags
4. Match patterns elsewhere in corpus
    1. Acquire top-scoring pattern(s)
5. Acquired patterns tags new names
    1. Acquire top-scoring name(s)
6. Repeat

# Pre-processing

- Text-Zoner
  - Extract textual content
  - Strips of headers, footers etc.
- Tokenizer
  - Produces lemmas
- POS tagger
  - Statistically trained on WSJ
  - Unknown or foreign words are not lemmatized and tagged as noun

# Seeds

- For each *target* category select N initial *trusted* seeds
  - Diseases:
    - Cholera, dengue, anthrax, BSE, rabies, JE, Japanese encephalitis, influenza, Nipah virus, FMD
  - Locations:
    - United States, Malaysia, Australia, Belgium, China, Europe, Taiwan, Hong Kong, Singapore, France
  - Others
    - Case, health, day, people, year, patient, death, number, report, farm
- Use frequency counts computed form corpus or some external data-base
- Many more additonal categories can be defined

# Positive vs. Negative Seeds

- A seed name serves as
  - a positive example for its own class, and
  - a negative example for all other classes.
- Negative examples help steer the learner away from unreliable patterns
  - Competing classes
  - Termination of unsupervised learning

# Pattern generation

- Tag every occurrence of each seed in corpus
  - "…new cases of <dis> cholera </dis> this year in …"
- For each tag, generate context rule: start/left-tag
  - [new case of <dis> cholera this year]
- Generalized left-side candidate patterns:
  - [new case of <dis> * * * ]
  - [* case of <dis> * * * ]
  - [* * of <dis> * * * ]
  - [* * * <dis> cholera this year ]
  - [* * * <dis> cholera this * ]
  - [* * * <dis> cholera * * ]

# Pattern generation

- For each tag, generate context rule: end/right-tag
  - [case of cholera </dis> this year in]
- Generalized right-side candidate patterns:
  - [case of cholera </dis> * * *]
  - [* of cholera </dis> * * *]
  - [* * cholera </dis> * * *]
  - [* * * </dis> this year in]
  - [* * * </dis> this year * ]
  - [* * * </dis> this * * ]
- Note: all are potential patterns

# Pattern application

- Apply each candidate pattern to corpus, observe where the pattern matches
  - E.g., the pattern [* * of <dis> * * *]
- Each pattern predicts one boundary: search for the partner boundary using a noun group NG regex:
  - [Adj* Noun+]
  - *"…distributed the yellow fever vaccine to the people"*
- The resulting NG can be (wrt. currently tagged corpus)
  - Positive: "…case <u>of</u> <dis> dengue </dis> …"
  - Negative: "…North <u>of</u> <loc> Malaysia </loc> …"
  - Unknown: "…symptoms <u>of</u> <?> swine fever </?> in …"

# Identify candidate NGs

- Sets of NG that the pattern p matched
  - Pos = distinct matched NG types of correct category
  - Neg = distinct matched NG types of wrong category
  - Unk = distinct matched NGs of unknown category

Collect statistics for each pattern

$$acc(p) = \frac{|Pos|}{(|Pos| + |Neg|)}$$

$$conf(p) = \frac{|Pos|}{(|Pos| + |Neg| + |Unk|)}$$

# Pattern selection

- Discard pattern p if acc(p) < $\theta$
- The remaining patterns are ranked by
  - Score(p) = conf(p)*log|Pos(p)|
- Prefer patterns that:
  - Predict the correct category with less risk
  - Stronger support: match more distinct known names
- Choose top n patterns for each category
  - [* die of <dis> * * *]
  - [* vaccinate against <dis> * * *]
  - [* * * </dis> outbreak that have ]
  - [* * * </dis> * * *]
  - [* case of <dis> * * *]

To get positive score, a pattern must have at least two distinct NGs as positive example, and more positive than negative exam.

# Name selection

- Apply each accepted pattern to corpus, to find candidate names (using the NG)
  - “More people die of <dis> profound heartbreak than grief.”
- Rank each name type t based on quality of patterns that match it:

$$Rank(t) = 1 - \prod_{p \in M_t} (1 - conf(p))$$

$M_t$ is the set of accepted patterns which match any of the instances of t

  - Require $|M_t| \geq 2 \Rightarrow$ t should appear $\geq$ 2 times
  - $M_t$ contains at least one pattern predicting the left boundary of t and one pattern predicting the right boundary
  - Conf(p) assigns more credit to reliable patterns

# Name selection

- Accept up to 5 top-ranked candidate names for each category
- Iterate learning algorithm until no more names can be learned
  - Bootstrap by using in each new iteration the extended set of new names to re-annotate the corpus

# Salient Features of Nomen

- Generalized names
- A few manually-selected seeds
- Un-annotated corpus
- Un-restricted context (no syntactic restrictions)
- Patterns for left and right contexts independently
- Multiple categories simultaneously

# Experiments

- Construction of reference lists for judging recall & precision of NOMEN

Compiled from multiple sources (medical DB, Web, manual review)

Appearing two or more time in development corpus

Manual list + acronyms + strip generic heads

| Reference List | Disease | Location |
|---|---|---|
| Manual | 2492 | 1785 |
| Recall (26K)<br>Recall (100K) | 322<br>616 | 641<br>1134 |
| Precision | 3588 | 2404 |

Score recal against recall list and precision against precision list; Distinguish type and token tests

# Results

- Final recall & precision for 8 categories
  - Around 70% (in case of type-based evaluation)
  - Classical PN: Recall: 86-92%, Precision: above 70%
- Multi-class learning has positive effects
  - A category is less likely to expand beyond its true territory
  - The accepted names in each category serve as negative example for all categories
  - The learners avoid acquiring patterns with too many negatives
  - In some sense, the categories *self-tune* each other
- Comparison with human-in-the-loop
  - "More groups" can be as good as "few groups + human reviewer"
- Using a negative category (noun groups that belong to neither category, but generic terms), then also substantial increase in performance