Co-training & Bootstrapping

PD Dr. Günter Neumann
DFKI and Saarland University

Bootstrapping 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

... 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 ...

Bootstrapping 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 Bootstrapping 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 them.

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

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

Features

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

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

Feature → NE-type, h(Feature,NE-type)

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

where

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

α is a smoothing parameter

k = #NE-types

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.

Rules

Two separate types of rules:
Spelling rules
Context rules

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

Feature → NE-type, h(Feature,NE-type)

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

where

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

α is a smoothing parameter

$k = \#NE\text{-}types$

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.

Rules

Two separate types of rules:
Spelling rules
Context rules

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

Feature → NE-type, h(Feature,NE-type)

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

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

where

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

α is a smoothing parameter

$k = \#NE\text{-}types$

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

- Full-string = New York → Location
- 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

7 SEED RULES

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

- Full-string = New York → Location
- 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)

   Compute h(x,y), and induce at most N * K rules

   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:

Classification problem $f: X \rightarrow Y$

$f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$ for $i = 1 \ldots m$

$f_1(x_{1,i}) = f_2(x_{2,i})$ for $i = m+1 \ldots n$

(softer criteria requires f_1 and f_2 to minimize their disagreements $\rightarrow$ similarity)

Can partition features X into 2 types of features $x = (x_1, x_2)$

Each type is sufficient for classification

x_1, x_2 not correlated to tightly (e.g., no deterministic function from x_1 to x_2)

Why Separate Spelling, Context Features?

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

f_i must correctly classify labeled examples, and

Requirements:

Classification problem $f: X \rightarrow Y$

$f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$ for $i = 1 \ldots m$

$f_1(x_{1,i}) = f_2(x_{2,i})$ for $i = m+1 \ldots n$

(softer criteria requires f_1 and f_2 to minimize their disagreements $\rightarrow$ similarity)

Can partition features X into 2 types of features $x = (x_1, x_2)$

Each type is sufficient for classification

x_1, x_2 not correlated to tightly (e.g., no deterministic function from x_1 to x_2)

Why Separate Spelling, Context Features?

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

Requirements:

- Classification problem $f: X \rightarrow Y$
  - $f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$ for $i = 1 \ldots m$
  - $f_1(x_{1,i}) = f_2(x_{2,i})$ for $i = m+1 \ldots n$
  - (softer criteria requires f_1 and f_2 to minimize their disagreements → similarity)
- Can partition features X into 2 types of features $x = (x_1,x_2)$
- Each type is sufficient for classification
- 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.

Why Separate Spelling, Context Features?

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

Requirements:

Classification problem $f: X \rightarrow Y$

$f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$ for i = 1…m

$f_1(x_{1,i}) = f_2(x_{2,i})$ for i = m+1…n

(softer criteria requires f_1 and f_2 to minimize their disagreement similarity)

Can partition features X into 2 types of features x = (x_1,x_2)

Each type is sufficient for classification

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?

Why Separate Spelling, Context Features?

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

f_i must correctly classify labeled examples, and

Requirements:

Classification problem $f: X \rightarrow Y$

$f_1(x_{1,i}) = f_2(x_{2,i}) = y_i$ for i = 1…m

must agree with each other on unlabeled ex.

$f_1(x_{1,i}) = f_2(x_{2,i})$ for i = m+1…n

(softer criteria requires f_1 and f_2 to minimize their disagreement similarity)

Open question: best similarity function?

Can partition features X into 2 types of features $x = (x_1,x_2)$

Each type is sufficient for classification

x_1,x_2 not correlated too tightly (e.g., no deterministic function from x_1 to x_2)

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:

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

➢ The algorithm maximizes the number of unlabeled examples on which the two decision lists 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