Unsupervised Models for Named Entity Classification, Michael Collins and Yoram Singer, 1999

Presented by Michal Novemsky

Using unlabeled examples for the problem of named entity classification

- This is a complex problem, and would seem to require many labeled examples to train a classifier
- But only 7 “seed” rules are needed for supervision with unlabeled data
- Spelling of a name and its context are often sufficient to determine its type

Task: to learn a function from an input string to its type

- Categories of type:
  - Person, e.g. “Mrs. Green”
  - Organization, e.g. “Johnson & Johnson”
  - Location, e.g. “Manhattan”

This approach uses spelling and contextual rules

- A spelling rule can be:
  - A look up for the string (e.g. that “Manhattan” is a location)
  - A rule that looks at words within a string (e.g. if a string contains “Mrs.”, the string is a person)
- A contextual rule considers the context, i.e. the words around the string in the sentence (e.g. any proper name modified by an appositive whose head is “president” is a person, such as: “… Mr. Cooper, a vice president of…”

Using unlabeled data greatly reduces the need for supervision

- Using about 90,000 unlabeled examples, the methods used here classify names with over 91% accuracy
- Hints such as in the previous examples (e.g. that when “Mr.” is in the string and “president” is in apposition to it, this suggests that “Mr. Cooper” is a person) frequently appear and are very useful in constructing classifiers

Two algorithms are used

- DL-CoTrain, a decision-list learning algorithm that uses only the set of 7 “seed” rules to classify names
- CoBoost, a boosting algorithm that uses both labeled and unlabeled data, and builds two classifiers in parallel

DL-CoTrain is based on both:

- A decision list method from Yarowsky, 1995, an algorithm for word-sense disambiguation that makes use of redundancy in contextual features
- A method proposed by Blum and Mitchell, 1998, which provides the impetus for separating spelling and contextual features

The Data Used

- 971,746 sentences from The New York Times were parsed, and word sequences that met the following criteria were extracted as named entity examples (see next slide):

Word Sequences Considered

- A sequence was one of consecutive proper nouns within a noun phrase, whose last word was head of the noun phrase (e.g. “Maury Cooper”)
- The noun phrase containing the word sequences appeared in one of the following contexts, shown on the next slide:

Two Contexts for the NP Containing the Word Sequence

- There was an appositive modifier to the NP, whose head was a singular noun,

  e.g. “…says Maury Cooper, a vice president at S.&P.”

- The NP was a complement to a preposition, which was the head of a prepositional phrase, which in turn modified another noun phrase, whose head was a singular noun,

  e.g. “…fraud related to work on a federally funded sewage plant in Georgia”

A Contextual Predictor Was Also Extracted

- In the appositive case, this was the head of the modifying appositive (in the example given, this would be “president”)
- In the preposition case, this was the preposition along with the noun the PP modified (in the example given, this would be “plant_in”)

(Spelling, Context) pairs

- Named-entity string itself is referred to as “spelling”
- Contextual predicate referred to as “context”
- These pairs allow features to be extracted

Feature Extraction

- Extracted features are used to represent each example for the learning algorithm, but each feature only takes into account either the spelling or context

List of Features Used

full-string=x The full string (e.g., for *Maury Cooper*, `full-string=Maury_Cooper`).

contains(x) If the spelling contains more than one word, this feature applies for any words that the string contains (e.g., *Maury Cooper* contributes two such features, `contains(Maury)` and `contains(Cooper)`.

allcap1 This feature appears if the spelling is a single word which is all capitals (e.g., `IBM` would contribute this feature).

allcap2 This feature appears if the spelling is a single word which is all capitals or full periods, and contains at least one period. (e.g., `N.Y.` would contribute this feature, `IBM` would not).

nonalpha=x Appears if the spelling contains any characters other than upper or lower case letters. In this case `nonalpha` is the string formed by removing all upper/lower case letters from the spelling (e.g., for *Thomas E. Petry* `nonalpha=.`, for *A.T.&T.* `nonalpha=..&.`).

context=x The *context* for the entity. The *Maury Cooper* and *Georgia* examples would contribute `context=president` and `context=plant_in` respectively.

context-type=x `context-type=appos` in the appositive case, `context-type=prep` in the PP case.

Examples of Named Entities and Their Features

Sentence	Entities (Spelling/Context)	Features
But Robert Jordan, a partner at Steptoe & Johnson who took ...	Robert Jordan/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

The Supervised Version of the Decision List Algorithm

- Input: n labeled examples of form (x_i, y_i), where x_i is a set of m_i features associated with the ith example ($\mathcal{X}$ is the set of possible features), and y_i is the label of the ith example ($\mathcal{Y}$ is the set of possible labels)
- Output: function $h : \mathcal{X} \times \mathcal{Y} \rightarrow [0,1]$ where $h(x,y)$ is an estimate of $p(y \mid x)$. h can also be thought of as defining a decision list of rules $x \rightarrow y$ ranked by their "strength" $h(x, y)$.

The 7 Seed Rules for DL-CoTrain

- Each of them given a strength of 0.9999

```
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 DL-CoTrain Algorithm Used to Induce New Rules, part 1

- Set *n* = 5, where *n* is the max # of rules of each type induced at each iteration
- Set the spelling decision list equal to the set of seed rules
- Label the training set using the current set of spelling rules (do not label examples where no rule applies)
- Use labeled examples to induce a decision list of contextual rules, using the method in the supervised version

  (to be continued)

The DL-CoTrain Algorithm Used to Induce New Rules, part 2

Count'(*x*) = number of times feature *x* is seen with some known label in the training data. For each of the 3 labels, take the *n* contextual rules with the highest value of *Count'*(*x*) whose unsmoothed strength is above a p_{min} threshold (but if < *n* rules have precision above the threshold, then keep only the rules exceeding the threshold). In these experiments, p_{min} was fixed at 0.95.

So at each iteration, the method induces at most *n* x *k* rules, where *k* is the number of possible labels (in these experiments it is 3)

The DL-CoTrain Algorithm Used to Induce New Rules, part 3

- Label the training set using the current set of contextual rules (do not label examples where no rule applies)
- On this new labeled set, choose at most *n* x *k* spelling rules using the same method as in step 4 (which gets the contextual rules). Set the spelling rules to be the seed set plus all the rules selected

The DL-CoTrain Algorithm Used to Induce New Rules, part 4

- If $n < 2500$ set n to $n + 5$ and go back to step 3 (labeling the training set using the spelling rules). Else, label the training data with the combined spelling and contextual decision list and induce a final decision list from the labeled examples where all rules (no matter their strength) are added to the decision list

Differences Between DL-CoTrain and Yarowsky's Algorithm

- DL-CoTrain is more cautious, as it imposes a gradually increasing limit on the number of rules that can be added at each iteration
- DL-CoTrain separates the spelling and contextual features, alternating between labeling and learning with the two types of features. The algorithm assumes that either the spelling or context alone is sufficient to build a classifier

Why Separating Contextual and Spelling Features is Important

If two functions, f_1 and f_2 are generated, each corresponding to one type of feature (that is, spelling and contextual), they must each be able to classify the labeled examples on their own, and must agree with each other on the unlabeled examples. This leads to a high level of agreement between the two decision lists, which leads to a high level of accuracy for the DL-CoTrain algorithm

CoBoost

- Based on a previous algorithm called AdaBoost, which was developed for supervised learning, and finds a weighted combination of simple (i.e. weak) classifiers, where the weights are chosen to minimize a function that bounds the classification error on a set of training examples
- CoBoost is similar but minimizes a bound on the number of unlabeled examples on which two classifiers disagree, building two classifiers iteratively

Other Learning Methods

- Baseline: tags all entities as the most frequent class type (organization)
- Expected Maximization (EM): performs an expectation (E) step alternating with a maximization (M) step of likelihoods
- Yarowsky 95: the algorithm DL-CoTrain was based on (it didn't separate spelling and contextual features and wasn't cautious)
- Yarowsky-cautious: an intermediate algorithm between Yarowsky 95 and DL-CoTrain that limited the number of rules added at each stage

Evaluation, part 1

- 88,962 (spelling, context) pairs were extracted as training data, 1000 of which were picked at random and hand-labeled to get a test set
- Each example got 1 of 4 labels: the 3 types of names, or “noise” for those not considered a name. But of the noise examples, 38 were temporal expressions (day of the week or month of the year), and these were excluded, leaving 962 labeled examples

Evaluation, part 2

- The results were:
  - Location: 186
  - Person: 289
  - Organization: 402
  - Noise: 123 - 38 = 85
- N_c represents the number of examples an algorithm classified correctly, where all items labeled “noise” were considered incorrect, in two measures of accuracy
  - Accuracy (Clean) = $\frac{N_c}{962-85}$
  - Accuracy (Noise) = $\frac{N_c}{962}$

Accuracy of Different Learning Methods

Learning Algorithm	Accuracy (Clean)	Accuracy (Noise)
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%