Self-training and Co-training Applied to Spanish Named Entity Recognition

Zornitsa Kozareva, Boyan Bonev, and Andres Montoyo

Departamento de Lenguajes y Sistemas Informáticos,
Universidad de Alicante, Spain
{zkozareva, montoyo}@dlsi.ua.es
{bib}@alu.ua.es

Abstract. The paper discusses the usage of unlabeled data for Spanish Named Entity Recognition. Two techniques have been used: self-training for detecting the entities in the text and co-training for classifying these already detected entities. We introduce a new co-training algorithm, which applies voting techniques in order to decide which unlabeled example should be added into the training set at each iteration. A proposal for improving the performance of the detected entities has been made. A brief comparative study with already existing co-training algorithms is demonstrated.

1 Introduction

Recently there has been a great interest in the area of *weakly supervised learning*, where unlabeled data has been utilized in addition to the labeled one. In machine learning, the classifiers crucially rely on labeled training data, which was previously created from unlabeled one with some associated cost. Self-training and co-training algorithms allow a classifier to start with few labeled examples, to produce an initial weak classifier and later to use only the unlabeled data for improving the performance.

In previous research by Collins and Singer [2], co-training was applied only to Named Entity classification, by making a split of features into contextual and spelling ones. They point as a future work the development of a complete Named Entity Recognition (NER)[1] system, which we build using self-training and co-training techniques.

We studied the already existing co-training methods and we introduce a new so called "voted co-training" algorithm. The method guarantees the labeling confidence of the unlabeled examples through a voting scheme.

The system has been developed and tested for Spanish language, but having the proper feature set for a NER system in another language, we can say that

[1] NER system consists in detecting the words that make up the entity and then classify these words into predefined categories such as people, organization and location names, temporal expressions etc.

the same experiments can be conducted with no restrains. For Spanish the obtained results are encouraging, 90.37% f-score for detecting 2000 entities using 8020 unlabeled examples and 67.22% f-score for entity classification with 792 unlabeled examples.

2 Named Entity Recognition

One Named Entity Recognition system plays important role in lots of natural language applications such as Information Extraction, Information Retrieval, Question Answering etc., by providing the most informative instances in a text, for instance names of people, locations, organizations. A NER task consists of two subtasks, one for entity detection and another for entity classification.

Recently there has been a great interest in entity recognition using diverse machine learning techniques such as AdaBoost, Maximum Entropy, Hidden Markov Models etc., as well as the development of the features needed by these classifiers. However the focus of our paper is NER system build on unlabeled data. For the experimental settings, we incorporated the feature set previously studied by *Kozareva et al.,* [5].

lexical

- **p**: position of w_0 (e.g. the word to be classified) in a sentence
- $c[-3,+3]$: word forms of w_0 and the words in its window ± 3
- **fW**: first word making up the entity
- **sW**: second word making up the entity if present

orthographic

- **aC**: all letters of w_0 in capitals
- $iC[-3,+3]$: $w_{-3}, w_{-2}, w_{-1}, w_0, w_{+1}, w_{+2}, w_{+3}$ initiate in capitals

trigger word: w_{-1} and w_{+1} trigger word for location, person, organization

gazetteer word: w_0 belong to gazetteer list

Fig. 1. Features for NE detection and classification

For NED, we used the BIO model proposed by [7]. A word should have one of the following three tags: **B** indicating a word at the beginning of a NE, **I** for word inside a NE and **O** tag for words outside a NE.

The annotation scheme is demonstrated in the example: *Soy_O Antonio_B Guijero_I de_O Portugal_B ._O*[2]. The following features have been used for NED: p, $c[-2,+2]$[3], aC and $iC[-2,+2]$.

We classify our entities into four categories - PERson, LOCation, ORGanization and MISCellanegous. The following set of features has been considered: $c[-3,+3]$, fW, sW, aC, $iC[-3,+3]$, trigger[4] and gazetteer[5] word.

[2] the English meaning is: *I am Antonio Guijero from Portugal.*

[3] in our example the word form at position 0 is *Soy.*

[4] semantically significant word pointing to some of the categories person, location, organization; e.g. city is a trigger word for locations.

[5] collections of names of people, locations, organizations.

3 Weakly Supervised Algorithms

In this section we mention already existing co-training and self-training algorithms, and describe the proposed by us voted co-training.

3.1 Co-training

Blum and Mitchel [1] assume that there exists two independent and compatible feature sets or views of data, where each feature set defining a problem is sufficient for learning and classification purposes. A classifier learned on each of those redundant feature subsets can be used to label data for the other and thus expand each other's training set. However in a real-world application, finding independent and redundant feature splits can be unrealistic and this can lead to deterioration in performance [6].

From another side Goldman and Zhou [4], proposed a co-training strategy that doesn't assume feature independence and redundancy. They learn two different classifiers from a data set. The idea behind this strategy is that the two algorithms use different representations for their hypotheses and thus they can learn two diverse models that can complement each other by labeling some unlabeled data and enlarge the training set of the other. In order to decide which unlabeled examples a classifier should label, they derive confidence intervals. We adopt this co-training strategy, having two different basic classifiers and a third (external) classifier, that decides which labeled example to be included into the training data, when the initial classifiers disagree.

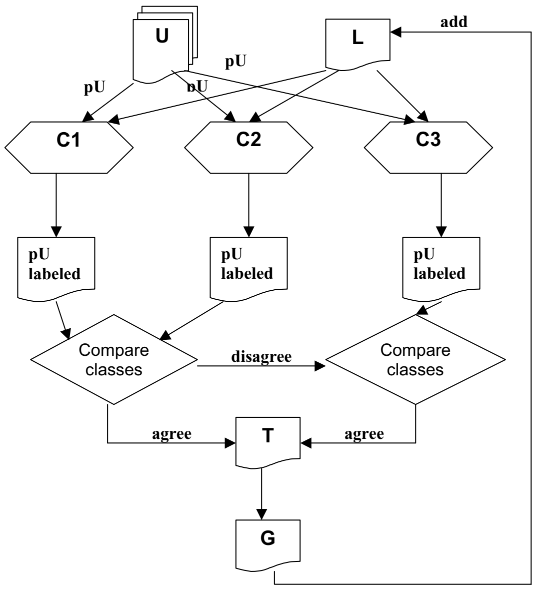
Collins and Singer [2] introduce the CoBoost algorithm to perform Named Entity classification, which boosts classifiers that use either the spelling of the named entity, or the context in which that entity occurs. Our approach differs from theirs by the co-training algorithm we use, the classification methods we worke with and the feature sets used for the NED and NEC modules.

3.2 Self-training

The definition for self-training could be found in different forms in the literature, however we adopted the definition of Nigam and Ghani [6]. We need only one classifier, without any split of features. For several iterations the classifier labels the unlabeled data and converts the most confidently predicted examples of each class into a labeled training example.

3.3 Voted Co-training Algorithm

After the resume of the existing co-training algorithm, here we introduce our method. Voted co-training starts with a small set of hand-labeled examples and three classifiers that learn the same pool of unlabeled examples. For each iteration the unlabeled data set is turned into labeled and instances with some growing size predefined by the user are added into the training set. In order to guarantee the labeling confidence of the unlabeled examples, for each example



- C_1 and C_2 two different classifiers
- C_3 external different classifier used in voting
- L a set of labeled training examples
- U a set of unlabeled examples
- T a temporal set of instances

Loop for I iterations:

1. do a pool pU of P randomly selected examples e_j from U
2. use L to individually train classifiers C_1, C_2, C_3 and label examples in pU
3. $\forall e_j \in pU$ whose classes agree by C_1 and C_2, do $T = T \cup \{e_j\}$
4. $\forall e_i \in pU$ whose classes disagree by C_1 and C_2, use C_3 and apply voting; select examples whose classes agree between two classifiers and add them to T
5. take randomly G examples from T and add them to L, while maintaining the class distribution in L
6. empty T

Fig. 2. the proposed Voted Co-training Algorithm

a voting strategy has been applied. Two initial classifiers compare the predicted class for each instance. When they agree, this instance is added directly into a temporal set T, when they disagree, the prediction of an external classifier is considered and voting among them is applied. To the already existing temporal set T are added only those instance on which the external classifier agrees with at least one of the initial classifiers.

We incorporate voting, knowing that such technique is used to examine the outputs of several various models and select the most probable category predicted by the classifiers. From another side by voting, the performance of a single classifier is outperformed.

The voted co-training scheme is presented in Figure 2. For further notations in this paper, we refer to I as the number of iterations, to P as the pool size (e.g. number of examples selected from the unlabeled set U needed for annotation at each iteration), to G as the growing size (e.g. the number of most confidently labeled examples added at each iteration to the set of labeled data L).

3.4 Data and Evaluation

The experimental data we worked with is part of the Spanish Efe corpus, used in the competitions of Clef [6]. The test file contained around 21300 tokens of which 2000 have been annotated by human as NEs. The correct classes of the test file have been used only when the results are calculated with the *conlleval* [7] evaluation script.

4 Self-training for Named Entity Detection (NED)

Considering the time-performance disadvantage of self and co-training techniques, and the complexity of each one of the tasks we have to resolve, we decided to use self-training for NEDetection.

The self-training algorithm starts with small set of 20 hand-labeled instances as a training set[8]. At each iteration a pool of P unlabeled examples is made, and one single classifier turns them into labeled ones. Only the most confident examples are added into L, keeping a constant ratio in the training data, for avoiding to introduce imbalance of the class set. The classifier we used has been K-nn algorithm[9], known with its property of taking into consideration every single training example when making the classification decision and significantly useful when training data is insufficient.

4.1 Experiments and Results

The parameter settings are: growing size $G = \{10, 20, 30, 40, 50, 100, 200, 500\}$, pool $P = \{30, 50, 60, 70, 80, 200, 500, 1000\}$, $I = 40$. The constant ratio in the training data is 5:3:2 for **O**, **B** and **I** tags, due to the frequent appearance of tag **O** compared to the other two categories.

The achieved performances with these settings can be found in Table 1. Note that the exposed results are after BIO error analysis, which we discuss in section

[6] http://clef.isti.cnr.it/

[7] the script contained the following measures: Precision (of the tags allocated by the system, how many were right), Recall (of the tags the system should have found, how many did it spot) and $F_{\beta=1}$(a combination of recall and precision).

[8] in our case, the first sentences in the unlabeled data.

[9] with one nearest neighbour.

4.2. We indicate with G10[10] the f-score obtained for all **BIO** tags with growing size G equal to 10, for each one of the 40 iterations.

In Table 1, the best performance per setting is denoted in bold. For instance, G10 reached 81.71% with 340 training size. Comparing G10's best performance with 340 training examples to the other settings, it can be seen that performance is not the same[11]. The difference comes from the examples added to L, which for each iteration and experimental setting did not repeat. For the next growing sizes of 20, 30, 40 and 50, the maximum performance is around 80.43%, 76.9%, 79.88% and 78.88%. For $G = 100$ with 1320 training examples 82.54% have been obtained; compared to the best performance of G10 this result is higher, however the number of examples participating in the training set is bigger. For all experimental settings, the best performance of 84.41% has been reached with 1620 examples and growing size 200.

Summarizing, we can say that satisfactory results for NE detection can be reached, using small sized training data. From another side for instance-based learning, it is not necessary to be stored all training instances, rather few examples inside stable regions. It can happen that an instance correctly classified by the self-training algorithm and added to L may not contribute with new information. Thus the performance will not improve. In future, we would like to measure the informativeness of each individual instance and include only those bringing novel information.

4.2 Error Analysis of the Detected Entities

We made analysis of the obtained results and managed to resolve and correct around 16% of the occurred errors. The postprocessing consist of:

BIO substitution, applied when tag **O** is followed by **I**. We simply replace **I** by **B** when the analyzed word starts in capital letter, in the other case we put **O**. As can be seen from the example in Section 2 and regarding the definition given in [7], tag **O** cannot be followed by tag **I**. The first 6 columns in Table 1 demonstrate the performance of each individual tag (e.g. B and I). With BIO substitution, performance is improved with 10%.

Statistical representation, compares $f(w_i)$, the frequency of a word w_i starting in capital letter and tagged as **B**, versus f^*, the frequency of the same word w_i taking its lowercased variant frequency from the test file. *Tambien_B* is detected 63 times as possible entity, but its variant $f^*(tambien) = 265$, shows that the word tends to be written in lowercased letters. In this case tag B is transformed into **O**.

For entities as *La_B Mancha_I*, this substitution will be erroneous. First *La* is part of the entity and is correctly classified, but its lowercase variant *la* is

[10] For the other notations, the number next to G indicates the growing size per iteration. The first 5 columns represent the scores before and after postprocessing. With iB, iI and iG10 are denoted the initial results of self-training without BIO error analysis and with cB, cI and G10, the results after BIO error analysis.

[11] the comparison for the same training size with the other settings is denoted in italic in Table 1.

Table 1. *Self-training performance for NED task*

I	iB	cB	iI	cI	iG10	G10	G20	G30	G40	G50	G100	G200	G500
1	64.40	79.78	34.31	47.01	52.94	69.26	73.51	70.23	74.00	66.32	71.80	73.91	*77.76*
2	66.10	81.92	33.63	52.35	53.69	72.43	78.41	70.69	71.83	71.13	73.16	*71.93*	78.47
3	67.69	81.53	34.84	53.88	55.69	72.85	74.29	72.57	78.30	72.73	*73.91*	75.39	76.18
4	72.10	81.52	41.10	53.98	61.20	72.92	73.88	71.59	78.12	73.9	79.39	76.73	80.26
5	72.17	80.86	41.98	54.95	61.52	72.81	70.85	76.62	79.04	76.71	76.64	74.72	81.73
6	72.07	80.79	39.72	54.38	60.39	72.56	66.53	76.10	72.93	*74.93*	78.40	76.00	81.38
7	72.29	81.09	38.25	55.56	60.11	73.09	73.74	74.19	76.39	75.76	77.28	76.67	81.91
8	73.62	81.12	38.14	55.37	61.67	73.04	74.10	73.44	*78.24*	76.65	79.29	77.40	81.42
9	72.65	80.97	38.10	52.36	60.66	71.95	70.42	73.48	76.06	76.01	79.91	77.49	81.23
10	72.56	80.75	37.11	50.92	60.24	71.32	73.75	*73.94*	77.15	76.81	80.27	78.06	81.83
11	74.11	80.95	32.94	50.04	60.15	70.96	75.23	72.82	68.91	76.84	81.99	77.84	81.76
12	76.46	80.14	35.57	50.19	63.54	70.66	74.56	73.20	71.18	76.99	81.82	79.22	81.83
13	76.25	80.20	38.33	56.63	64.81	72.89	75.57	75.16	72.77	77.90	**82.54**	80.60	82.22
14	76.53	80.20	32.96	57.44	61.71	73.18	76.33	75.21	74.96	**78.88**	82.17	80.93	82.40
15	76.75	79.89	36.15	62.61	63.29	74.78	78.24	75.12	75.45	78.29	80.44	81.32	82.94
16	81.85	85.47	39.20	65.68	67.02	79.36	*78.30*	75.24	75.17	78.26	80.43	82.03	82.80
17	82.68	87.06	42.62	66.72	68.43	80.82	78.34	75.17	75.07	78.38	79.63	81.59	83.00
18	83.15	86.79	44.16	67.83	69.43	80.99	78.20	74.94	76.03	77.63	79.75	81.41	83.16
19	84.41	86.79	48.22	67.83	71.53	80.99	77.97	74.87	76.34	76.42	80.80	81.54	83.27
20	84.60	86.55	47.69	66.55	71.49	80.44	77.99	74.56	76.00	76.43	80.73	81.16	83.43
21	83.71	86.64	44.85	66.72	70.10	80.55	**80.43**	73.76	76.06	77.27	80.92	81.23	83.51
22	83.55	86.58	43.37	67.08	69.09	80.62	78.72	74.05	76.22	77.27	80.88	80.87	83.63
23	82.54	85.57	50.73	67.08	71.70	79.97	76.99	74.33	76.72	77.31	80.41	80.90	83.56
24	81.74	84.55	50.85	68.15	71.40	79.67	77.55	74.01	77.49	77.04	79.87	81.83	83.85
25	81.67	84.25	47.30	68.20	69.56	79.46	77.53	74.07	77.48	77.03	80.28	81.49	83.50
26	76.29	78.99	47.76	69.29	66.89	76.31	79.32	75.07	76.80	77.59	79.82	81.06	83.90
27	75.84	78.42	46.43	66.61	66.14	75.13	79.14	76.00	77.36	77.54	80.06	81.09	83.79
28	81.70	85.08	46.76	68.45	69.43	80.17	77.95	76.12	77.99	77.40	79.18	81.49	83.87
29	81.60	85.01	46.95	68.68	69.44	80.18	78.08	76.02	77.93	76.36	78.78	81.37	83.94
30	81.70	85.01	47.30	68.74	69.60	80.20	78.10	76.22	78.24	76.72	78.28	81.74	83.48
31	81.74	85.27	47.46	68.37	69.74	80.25	78.03	76.18	78.22	77.16	78.48	82.22	83.71
32	81.62	86.03	49.14	71.63	70.15	**81.71**	77.69	76.12	78.57	77.16	78.35	82.72	83.59
33	81.23	85.38	46.23	67.43	68.85	80.02	78.24	76.58	78.96	77.97	79.03	83.63	83.81
34	81.35	85.46	46.24	66.84	68.94	79.87	78.32	76.18	78.99	77.58	79.36	83.81	**84.39**
35	76.25	79.96	47.67	68.26	66.83	76.70	78.45	76.25	78.79	77.72	79.55	83.80	84.11
36	76.32	80.01	47.55	67.60	66.80	76.53	78.38	75.20	78.85	77.64	79.81	83.96	83.87
37	76.31	80.01	47.36	68.01	66.68	76.65	79.39	76.57	79.32	77.62	80.13	83.68	84.03
38	76.35	80.03	46.59	65.92	66.44	76.02	79.13	76.76	79.61	77.36	80.05	84.05	83.81
39	76.01	79.82	45.66	64.90	65.84	75.56	79.32	**76.90**	79.80	77.93	79.65	84.25	83.93
40	76.74	79.80	45.75	64.79	66.55	75.51	77.75	**76.90**	**79.88**	77.96	79.75	**84.41**	84.02

a determiner for female gender in Spanish language and has higher frequency than *La*. To avoid such substitutions, pairwise frequency of bigrams is considered. When the frequency of *La Mancha* is higher than the frequency of *la Mancha*, tag **B** for *La* is kept, otherwise is changed into *La_O Mancha_B*.

This transformation improved the f-score with 6%, for the experimental settings of P=500, G=200 at iteration 40. The final f-score for BIO classification achieved 90.37%, tag **B** was detected with 93.97% f-score and tag **I** reached 81.94%.

5 Co-training for Named Entity Classification (NEC)

The instances detected by the self-training algorithm are classified with the voted co-training method, as described in Figure 2. The learning process started with 10 hand-labeled examples in the following ratio 3:3:3:1, for ORG, PER, LOC and MISC classes. The instances representing MISC class tend to have rare appearance in text and the probability of encountering such instance is lower. As main classifiers have been used decision trees and k-nn, all implemented in TiMBL's package[3]. The HMM toolkit [8] has been developed for post tagging purposes, but we adopted it for NER purposes.

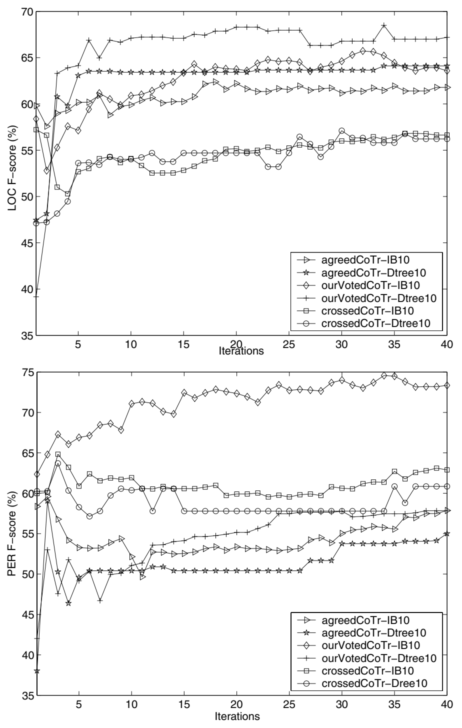


Fig. 3. From top to bottom and from left to right are represented the performances of three different co-training algorithms using k-nn and decision tree, for classifying the entities into location, organization, person and miscellaneous classes. The growing size used has been 10 for each of the 40 iterations.

5.1 Comparative Study and Discussion of the Obtained Results

A brief comparative study with other co-training algorithms has been made. The obtained results for each category are represented in separate graphics in Figure 3. By *crossedCoTr* is denoted the co-training algorithm where two classifiers simply exchange the instances they learn, feeding each other's input. *ourVotedCoTr* represents the proposed by us algorithm. The performance of a co-training where only the instances on which two classifiers agree and have been added into the training set L is denoted as *agreedCoTr*.

In the graphics can be seen how for each one of the four classes, our voted co-training outperforms the other two algorithms. For location class $agreedCoTr - IB10$ and $ourVotedCoTr - IB10$ start with similar efficiency, however after the 7th iteration, the curve of the voted co-training starts improving. For the same class $ourVotedCoTr - Dtree$ keeps 5% higher score than $agreedCoTr - Dtree$.

In general K-nn and decision tree dealt well with LOC and ORG classes reaching 68-70% performance. The contribution of the external classifier has been for PER and MISC classes. Compared to the other two methods, MISC class gained 8-9% better performance with *ourVotedCoTr*. This was due to the additional information provided with the agreement of the external classifier. In future we would like to work with more discriminative feature set for MISC class, since this was the class that impeded classifier's performance. As can be seen in the graphic, for many iterations two classifiers could not agree with an instance belonging to MISC class and the performance kept the same score.

6 Conclusions and Future Work

In this paper we demonstrated the building of a complete Named Entity Recognition system, using small set of labeled and large amount of unlabeled data, by the help of self-training and co-training. The obtained results are encouraging, reaching 90.37% for detection phase and 65% for classification[12].

The detection task was easily managed only with self-training. The Named Entity Classification task is more difficult, but with the proposed by us voted co-training algorithm, an outperformance of 5% per class was obtained compared to other co-training algorithms. The features used for miscellaneous class have not been so discriminative, and we would like to repeat the same experiment with a better set.

In future we would like to make a comparative study of self-training, active learning and the proposed by us voted co-training, while dealing with the NEC task. More challenging will be to investigate how voted co-training behaves compared to a supervised machine learning NER system. Finally for evaluating the effectiveness of the proposed method, it will be applied to other natural language processing task such as word sense disambiguation.

[12] considering the performance of person, location, organization and miscellaneous classes altogether.

Acknowledgements

This research has been partially funded by the Spanish Government under project CICyT number TIC2003-0664-C02-02 and PROFIT number FIT-340100-2004-14 and by the Valencia Government under project numbers GV04B-276 and GV04B-268.

References

1. A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *COLT: Proceedings of the Workshop on Computational Learning Theory*, pages 92–100, 1998.
2. M. Collins and Y. Singer. Unsupervised models for named entity classification. In *Proceedings of the Joint SIGAT Conference on EMNLP and VLC*, pages 100–11, 1999.
3. W. Daelemans, J. Zavrel, K. Sloot, and A. van den Bosch. TiMBL: Tilburg Memory-Based Learner. Technical Report ILK 04-02, Tilburg University, 2004.
4. S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 327–334, 2000.
5. Z. Kozareva, O. Ferrandez, A. Montoyo, R. Muñoz, and A. Suárez. Combining data-driven systems for improving named entity recognition. In *Proceedings of Tenth International Conference on Applications of Natural Language to Information Systems*, pages 80–90, 2005.
6. K. Nigam and R. Ghani. Analyzing the effectiveness and applicability of co-training. In *Proceedings of Ninth International Conference on Information and Knowledge Management*, pages 86–93, 2000.
7. T. K. Sang. Introduction to the conll-2002 shared task: Language independent named entity recognition. In *Proceedings of CoNLL-2002*, pages 155–158, 2002.
8. I. Schroder. A case study in part-of-speech tagging using the icopost toolkit. Technical Report FBI-HH-M-314/02, Department of Computer Science, University of Hamburg, 2002.