Learning Named Entity Classifiers using Support Vector Machines

Thamar Solorio and A. López López

Computer Science Department
Instituto Nacional de Astrofísica, Óptica y Electrónica
Luis Enrique Erro #1,
72840 Puebla, México

Abstract. Traditional methods for named entity classification are based on hand-coded grammars, lists of trigger words and gazetteers. While these methods have acceptable accuracies they present a serious drawback: if we need a wider coverage of named entities, or a more domain specific coverage we will probably need a lot of human effort to redesign our grammars and revise the lists of trigger words or gazetteers. We present here a method for improving the accuracy of a traditionally-built named entity extractor. Support vector machines are used to train a classifier based on the output of an existing extractor system. Experimental results show that this approach can be a very practical solution, increasing precision by up to 11.94% and recall by up to 27.83% without considerable human effort.

1 Introduction

While hand-coded grammars, gazetteers, chunkers, contextual rules and lists of trigger words provide a valuable source of information useful for building NE extractors, [1–4], they can become obsolete if no updating is performed. Another disadvantage of relying on this information is that the coverage of these tools might be too general or overly specific, thus achieving poor precision and recall when applied to more specific or different domains. However, they can present a useful starting point in building accurate Named Entity (NE) classifiers. We believe that machine learning techniques can be used to build automated classifiers trained on traditional hand-built NE extractors. Then, instead of manually redesigning the NE extractors we can allow the classifiers to learn from tags assigned by the NE extractor. Hopefully the NEs not covered by the extractor will be successfully classified by the learner.

We present here a new methodology for NE classification that uses Support Vector Machines (SVM) in order to enhance the accuracy of a NE Extractor System (NEES). The NEES is considered as a black box, we are only interested in its output, which is used as one of the attributes in our learning scenario. Our proposed solution can be considered as a stack of classifiers where in the first stage a traditional hand-built NEES is used to assign possible tags to the corpus, then these tags are used by a SVM classifier to obtain the final NE tags.

In addition, other attribute information used are the class values assigned to the 2 words to the left and right of the instance. Experimental results show that Support Vector Machines are successfully applied to this learning task increasing F-measure and accuracy.

The organization of this paper is as follows: the next section describes some of the most recent work in NE classification. Section 3 describes our learning scenario and provides a brief introduction to Support Vector Machines. In Section 4 we present some experimental results and Section 5 summarizes our conclusions and discusses possible future work.

2 Related Work

In [5] a new method for automating the task of extending a proper noun dictionary is presented. The method combines two learning approaches: an inductive decision-tree classifier and unsupervised probabilistic learning of syntactic and semantic context. The decision tree learns to assign semantic categories to named entities using a dictionary and a training corpus. In this stage only high confidence classifications are accepted. On a second stage, unsupervised learning is used to improve recall. The attribute information selected for the experiments uses POS tags as well as morphological information whenever available. Also, in this work the original named entity words are not used, at least not completely. This method uses the first two and last two words of the NE, along with contextual information (two words to the left of the NE and two words to the right).

A very interesting system for NE recognition based on Hidden Markov Models was proposed by Zhou and Su [2]. The novelty in this system being the combination of information used to build the HMM-based tagger: they use a combination of internal and external sub-features. The system considers three internal sub-features: in the first one they consider information relevant to discriminate between dates, percentages, times, monetary amounts and capitalization information; the second sub-feature consists of triggers that the authors consider useful for NE recognition; the last internal sub-feature contains gazetteer information, gathered from look-up gazetteers that contain lists of names of people, organizations, locations and other kinds of named entities. The external evidence considered is about context of other NEs already recognized. A list is updated with every named entity that has been recognized so far; when a new candidate is found an alias algorithm is invoked to determine its relation with the NE on the list.

Sekine et al. presented a named entity hierarchy containing 150 NE types [6]. They build the hierarchy in three steps: initially they build three different hierarchies, corpus-based, based on previous systems and tasks and based on thesauri; then they merge the three hierarchies (in this stage three experts are involved in the procedure); in the final step the final hierarchy is refined by tagging additional corpora and developing automatic taggers.

One work focused in NE recognition for Spanish is based on discriminating among different kinds of named entities: core NEs, which contain a trigger word as nucleus, syntactically simple weak NEs, formed by single noun phrases, and syntactically complex named entities, formed by complex noun phases. Arévalo et al. focused on the first two kinds of NEs [1]. The method is a sequence of processes that uses simple attributes combined with external information provided by gazetteers and lists of trigger words. A context free grammar, manually coded, is used for recognizing syntactic patterns.

In [7] the authors experimented with a risk minimization approach. They performed several experiments, combining different features. Their best performance for NE recognition in English is achieved by a system that combines dictionaries and trigger word lists with linguistic features extracted directly from the text, such as case information, token prefix and suffix. However, in the German data set the improvement from using additional information was not that great, the authors believe that this difference between the English and German data sets may be due to the fact that for English there is a higher availability of good quality dictionaries.

3 Learning NE Classifiers

In this section we describe the setting of our learning scenario and present a brief introduction to Support Vector Machines.

As mentioned previously, we build our NE classifiers using the output of a NEES. Our assumption is that by using machine learning algorithms we can improve performance of NE extractors without a considerable effort, as opposed to that involved in extending or redesigning grammars and lists of trigger words and gazetteers. Another assumption underlying this approach is that of believing that the misclassifications of the NEES will not affect the learner. Intuitively, one may consider the incorrectly classified instances as noisy. However we believe that by having available the correct NE classes in the training corpus, the learner will be capable of generalizing error patterns that will be used to assign the correct NE. If this assumption holds, learning from other's mistakes, the learner will end up outperforming the NEES.

In this work we are considering that all the possible named entities are already identified, that is, we are not dealing with the problem of recognizing which parts of the text are NEs. Named Entities are recognized and segmented by the NEES, our only concern being the NE classification in the same four types defined in the CoNLL 2002 competition: Person (*Per*), Location (*Loc*), Organization (*Org*) and Miscellaneous (*Misc*).

In order to build a training set for the learner, each instance n is described by a vector of six attributes, $\langle a_1, a_2, ..., a_6 \rangle$, where a_1 and a_2 are the classes assigned by the NEES to the first two words to the left of n, a_3 and a_4 are the classes for the first two words to the right of n, a_5 is the class value of n predicted by the NEES and a_6 is the true class value of n. Note that a_5 and a_6 will differ only when the base NEES misclassifies a named entity. There are two types of

possible errors from the NEES that the learner will try to overcome. The first type occurs when the NEES assigns the wrong NE class to the NE; the other type occurs when the NEES fails to recognize the NE. When this is the case the NEES assigns a Part Of Speech (POS) tag to the NE, although the same thing happens when the NEES needs to classify a non-NE word. It follows that the possible values for attributes a_1 to a_5 can take any value from the set of possible NEs, this is $NE = \{Per, Loc, Org, Misc\}$, plus the set of possible POS tags, which has over one hundred different tags; so the size of the feature space of our NE classification task is 5×10^{10}.

A graphical representation of our proposed NE classifier is given in Figure 1. We can see that a NEES is used as a black box, the output of which is fed to the SVM classifier. The SVM classifier uses this information (in some cases also additional information such as POS tags, capitalization information and lemmas) and assigns the final NE classification.

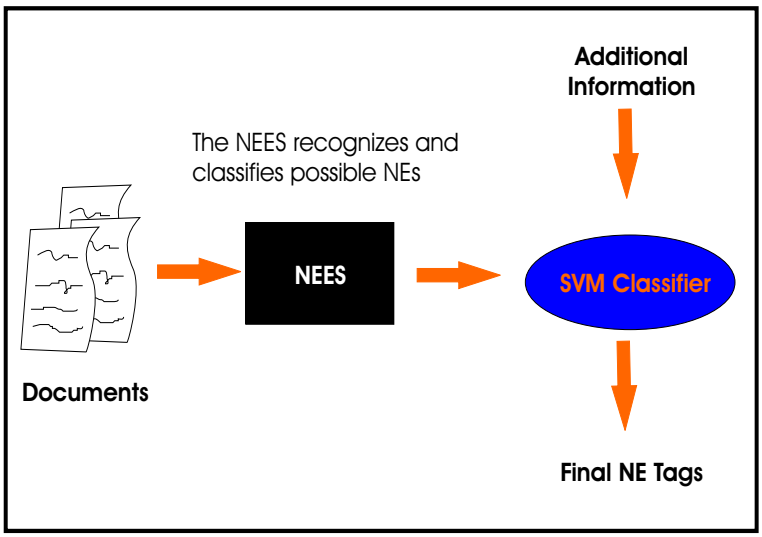

Fig. 1. A graphical representation of our NE classifier. The possible NEs together with the tags given by the NEES are used as input to the SVM classifier.

3.1 Low-Dimensionality Features

Having the high-dimensionality feature space described above has some serious drawbacks: the number of training examples needed in order to achieve good accuracy is proportional to the size of the feature space; but increasing the amount of training examples to meet this requirement might be unfeasible. Another disadvantage is the high computational resources needed to use SVM with a feature space such as this.

In order to overcome these difficulties we decided to reduce the size of the feature space. We achieve this by generalizing the POS tags, i.e. instead of having tags for all the possible kinds of pronouns, we kept one tag P that encapsulates the set of pronouns. We did the same for each POS. Table 1 shows the resulting reduced set of feature values. By doing this our reduced feature space has a size of $16^5 * 4 = 4,194,304$.

Feature Value	Description
Per	Person
Org	Organization
Loc	Location
Misc	Miscelaneous
N	Noun
A	Adjective
P	Pronoun
F	Punctuation mark
D	Determiner
V	Verb
S	Preposition
R	Adverb
T	Article
C	Conjunction
M	Numeral
I	Interjection

Table 1. Reduced Feature Values Set

3.2 Support Vector Machines

SVM have been successfully used in classification and regression tasks. This technique uses geometrical properties in order to compute the hyperplane that best separates a set of training examples [8]. When the input space is not linearly separable SVM can map, by using a kernel function, the original input space to a high-dimensional feature space where the optimal separable hyperplane can be easily calculated. This is a very powerful feature, because it allows SVM to overcome the limitations of linear boundaries. They also can avoid the over-fitting problems of neural networks as they are based on the structural risk minimization principle. The foundations of these machines were developed by Vapnik, for more information about this algorithm we refer the reader to [9].

4 Experimental Results

Having introduced our proposed solution we continue describing in this section the experimental setting used to evaluate it. We begin by describing the data sets

used, and continue presenting the results achieved. In our experiments we used the WEKA implementation of SVM [10]. In this setting multi-class problems are solved using pairwise classification. The optimization algorithm used for training the support vector classifier is an implementation of Platt's sequential minimal optimization algorithm [11]. The kernel function used for mapping the input space was a polynomial of exponent one.

Two data sets were used in the experiments, one was gathered by people in the NLP lab at our Institution. It consists of news in Spanish acquired from different newspapers from Mexico that cover disaster-related events. This collection contains a total of 285 NEs. Although it is a small corpus (which is not so bad as it gives us the opportunity of manual revision) it is large enough for our experimental evaluation. The other corpus is that used in the CoNLL 2002 competitions for the Spanish NE extraction task. This corpus is divided in three sets: a training set consisting of 20,308 NEs and two different sets for testing, *testa* which has 4,634 NEs and *testb* with 3,948 NEs, the former was designated to tune the parameters of the classifiers (development set), while *testb* was designated as the one used to compare the results of the competitors. As in our setting there is no parameter tuning we performed experiments with the two sets.

In order to evaluate our solution we used the Named Entity extractor developed by Carreras and Padró [12]. They have developed a set of NLP analyzers for Spanish, English and Catalan that include practical tools such as POS taggers, semantic analyzers and NE extractors. This NEES is based on hand-coded grammars and lists of trigger words and gazetteer information.

Classifiers	**Precision**	**Recall**	F_1	**Accuracy**
Baseline	**90.80%**	66.2%	76.62%	62.10%
SVM Features	78.60%	89.95%	83.91%	73.38%
SVM Features+NEE	87.02%	90.83%	88.88%	80.00%
SVM NEE (*)	88.30%	**91.73%**	**89.96%**	**81.75%**
% of improvement (best vs baseline)	-2.83%	27.83%	14.83%	24.04%

Table 2. Results of NE categorization with our data set. The last row shows the percentage of highest improvement comparing the best result with SVM, marked with "*", against the baseline.

We begin describing the results using our disaster corpus. We perform four different experiments and report the results in Table 2. We experimented here using 10-fold cross-validation. The first experiment, labeled Baseline is the result of the NEES. As can be seen, it has very high precision, while recall and thus F-measure, are not so good. The Baseline system achieved an accuracy of 62.10%. A different experiment was performed using SVM trained on features like POS tags, lemma and capitalization information. These features are acquired automatically from the text using the Spanish analyzers mentioned above. We used a five word

window, where we consider 2 words to the left and 2 words to the right of the target word, each of these words is described by its POS tag, its lemma and the capitalization of the word (all letters capitalized, first letter capitalized, digits or other when non of these is true). Each target word is then described by a vector of 16 attributes: 5 times 3 features plus the real class value. Results from this experiment are also in Table 2 under label SVM Features. We can see that even though we are not using the NEES tags we achieve higher recall, F-measure and accuracy than the NEES.

The results named SVM Features+NEE were obtained using the same features described in the previous experiment plus the NE tags assigned by the extractor system. This combination of features outperformed the previous results in all but one figure, achieving an accuracy of 80%. The last experiment performed with this corpus uses as features the output of the NEES, it does not use POS tags or additional information. Results are labeled SVM NEE, also shown in Table 2. As we can see, the methods using SVM outperformed the NEES in recall, F-measure and accuracy, the best results being those from training SVM on the extractor system tags. Precision of the NEES was the only figure that remained higher. However the improvements achieved by using SVM are as high as 27% in recall, 14.83% in F-measure and 24.04% in accuracy.

In Tables 3 and 4 we present results using the corpora from the CoNLL 2002 competition. In both tables we used the designated training set to build the classifiers. The first table shows the results of using the development set for testing. It can be seen that we achieve higher precision, F-measure and accuracy by using the NEES tags and additional features (POS tags, capitalization and lemmas), while the NEES by itself has the highest recall for this set. Table 4 shows the results of testing with the test set of CoNLL 2002, the SVM Feature+NEES classifier outperformed the Baseline method in three figures: precision, F-measure and accuracy.

Classifiers	**Precision**	**Recall**	F_1	**Accuracy**
Baseline	77.4%	**95.49%**	85.48%	74.64%
SVM Features+NEE (*)	**86.00%**	92.27%	**88.97%**	**80.14%**
SVM Features	67.00%	83.02%	74.12%	58.89%
SVM NEE	85.60%	91.31%	88.36%	79.15%
% of improvement (best vs baseline)	10%	-3.49%	3.92%	6.86%

Table 3. Results of NE categorization for CoNLL 2002 *development* set. The last row shows the percentage of highest improvement comparing the best result with SVM, marked with "*", against the baseline.

By comparing the results from the three tables it is interesting to note that for the CoNLL 2002 test sets the best results are achieved by combining features such as POS tags, capitalization and lemma information with the output of the NEES. While in our disaster data set the best results are achieved by using

Classifiers	Precision	Recall	F_1	Accuracy
Baseline	70.50%	**88.08%**	78.33%	64.38%
SVM Features+NEE (*)	80.06%	87.40%	**83.86%**	**72.21%**
SVM Features	71.60%	81.55%	76.23%	61.60%
SVM NEE	**80.20%**	86.57%	83.28%	71.35%
% of improvement (best vs baseline)	11.94%	-0.78%	6.59%	10.84%

Table 4. Results of NE categorization for CoNLL 2002 *test* set. The last row shows the percentage of highest improvement comparing the best result with SVM, marked with "*", against the baseline.

only the output of the extractor system. This is not surprising if we consider an important difference between our corpus and those from the CoNLL 2002: as our disaster-related corpus was carefully checked the NE tags are nearly error-free, while the other corpora are very large, which makes it unfeasible to manually correct any misclassification. There are some inconsistencies in these corpora that we were unable to correct. We believe that, by using additional information, SVM can achieve better results when there may be noise in the examples, as we suspect is the case. However, the three learning tasks share a common characteristic, the average best results were achieved by a method using SVM. The NE extractor system was always outperformed by some method based on machine learning in at least three out of the four measures.

5 Discussion

Most approaches to NE classification involve the use of hand-coded grammars, lists of trigger words and gazetteers, as observed by [7] in their introduction to the CoNLL-2003 Shared Task. While the accuracies of such systems may be acceptable in some domains, i.e. the domains covered by those lists and grammars, it is very likely that their accuracy will suffer when used to classify NEs in documents from a more specific subject. Consider then the effort of extending the coverage of these approaches: we need to revise the grammar, or regular expressions, to adapt them in order to cover new NE instances. In addition, it is very likely that the lists of trigger words will need some changes, together with gazetteers. We have presented here an alternate solution where instead of redesigning internally the NEES for new instances we need only to use a machine learning classifier, a SVM in this case, using the outputs of the current NEES correcting them if pertinent and retrain. In this way, we can take advantage of previous efforts to build the extractor. Our experimental results show that we can increase the overall performance of the NEES with this approach as assessed by F-measure and accuracy. We observed from the experiments that there is a minor penalty either in precision or recall although it is easily override by the improvement in the other measures.

We have presented here an alternative, based on machine learning techniques, for NE classification. This alternative can be applied when we need to have a

wider coverage of named entities, i.e. if the NEES was built from a very specific domain and we want to extend its usability, or if the NEES is very general and we need a more specified coverage of NEs, such as the case of our disaster related corpus. We believe that NE recognition can also be improved by using a similar approach. Some interesting possibilities of future work include:

- *Evaluating this approach with a NEES for English* We believe that this solution can be applied with similar success to the problem of NE classification for English. We are working now in finding a NEES that we can use in order to experiment.
- *Automatic Correction of Errors in Corpus* A major difficulty of doing research related with natural language processing is the lack of error-free annotated corpora. It can be a very extenuating task to verify the correctness of the tagging. We believe that machine learning can be used for building error detectors in corpora, and we are planning to explore this topic.
- *Using Unlabeled Data* Given that unlabeled data are more easily gathered, as opposed to labeled, there have been some interesting methods proposed in problems such as text categorization that exploit information contained in unlabeled data. We believe that using unlabeled data can benefit NE categorization and that machine learning algorithms such as SVM can be an interesting alternative in this direction.

Acknowledgements

We would like to thank CONACYT for partially supporting this work under grants 166934 and U39957-Y.

References

1. M. Arévalo, L. Márquez, M.A. Martí, L. Padró, and M. J. Simón. A proposal for wide-coverage spanish named entity recognition. *Sociedad Española para el Procesamiento del Lenguaje Natural*, 28, May 2002.
2. G. Zhou and J. Su. Named entity recognition using an hmm-based chunk tagger. In *Proceedings of ACL'02*, pages 473–480, 2002.
3. R. Florian. Named entity recognition as a house of cards: Classifier stacking. In *Proceedings of CoNLL-2002*, pages 175–178, 2002.
4. T. Zhang and D. Johnson. A robust risk minimization based named entity recognition system. In *Proceedings of CoNLL-2003*, 2003.
5. G. Petasis, A. Cucchiarelli, P. Velardi, G. Paliouras, V. Karkaletsis, and C. D. Spyropoulos. Automatic adaptation of proper noun dictionaries through cooperation of machine learning and probabilistic methods. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 128–135. ACM Press, 2000.
6. C. Nobata S. Sekine, K. Sudo. Extended named entity hierarchy. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, 2002.

7. E. F. Tjong Kim Zhang and F. De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the CoNLL-2003*, 2003.
8. M. O. Stitson, J. A. E. Wetson, A. Gammerman, V. Vovk, and V. Vapnik. Theory of support vector machines. Technical Report CSD-TR-96-17, Royal Holloway University of London, England, December 1996.
9. V. Vapnik. *The Nature of Statistical Learning Theory*. Number ISBN 0-387-94559-8. Springer, N.Y., 1995.
10. I. H. Witten and E. Frank. *Data Mining, Practical Machine Learning Tools and Techniques with Java Implementations*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 1999.
11. J. Platt. Fast training of support vector machines using sequential minimal optimization. In *Advances in Kernel Methods –Support Vector Learning, (B. Scholkopf, C. J. C. Burges, A. J. Smola, eds.)*, pages 185–208, Cambridge, Massachusetts, 1999. MIT Press.
12. X. Carreras and L. Padró. A flexible distributed architecture for natural language analyzers. In *Proceedings of LREC'02*, Las Palmas de Gran Canaria, Spain, 2002.