ARTÍCULO

Machine Learning Algorithms for Portuguese Named Entity Recognition

Ruy Luiz Milidiú[1], Julio Cesar Duarte[2], Roberto Cavalcante[1]

[1]Departamento de Informática, Pontifícia Universidade Católica
Marquês de São Vicente, 225
Gávea, Rio de Janeiro, Brazil, 22453-900
[2]Centro Tecnológico do Exército
Américas, 28705
Guaratiba, Rio de Janeiro, Brazil, 23020-470
milidiu@inf.puc-rio.br, jduarte@ctex.eb.br, rcavalcante@inf.puc-rio.br

Abstract

Named Entity Recognition (NER) is an important task in Natural Language Processing. It provides key features that help on more elaborated document management and information extraction tasks. In this paper, we propose seven machine learning approaches that use HMM, TBL and SVM to solve Portuguese NER. The performance of each modeling approach is empirically evaluated. The SVM-based extractor shows a 88.11% F-score, which is our best observed value, slightly better than TBL. This is very competitive when compared to state-of-the-art extractors for similar Portuguese NER problems. Our HMM has reasonable precision and accuracy and does not require any expert knowledge. This is an advantage for our HMM over the other approaches. The experimental results suggest that Machine Learning can be useful in Portuguese NER. They also indicate that HMM, TBL and SVM perform well in this natural language processing task.

Keywords: Machine Learning, Supervised Training, Named Entities.

1 Introduction

Named Entity Recognition (NER) is the problem of finding all proper nouns in a text and to classify them among several given categories of interest or to a default category called Others. There are three usual given categories: Person, Organization and Locations. Time, Piece, Event, Abstraction, Thing, and Value are some additional but less usual categories of interest. Here are some examples of possible Named Entities (NE):

- **Fernando Henrique Cardoso** discursou sobre o seu plano de.... (Person)
- A **Intel** lancará uma nova linha de... (Organization)
- A viagem até **Cascavel** tomará a maior parte... (Location)
- **Segunda-Feira** já estaremos em casa para... (Time)
- O **Terceiro Workshop sobre Segurança do Trabalho** será sediado ... (Event)

Here, by named entities, we mean the role the entity plays without considering its current context. For example:

- **PUC-Rio** está contratando novos professores...
- Tal evento será realizado na **PUC-Rio** a partir...

In the first sentence, we have no doubt that PUC-Rio is an organization given the context, although the second example can lead to an ambiguity whether it is an organization or a location. These phenomena can occur in various sentences like:

- **Brasil** disputará a Copa do Mundo....

These ambiguities are raised because the author omitted certain words that could disambiguate the sentence, like:

- Tal evento será realizado nas *dependências da* **PUC-Rio** a partir...
- *A seleção de futebol do* **Brasil** disputará a Copa do Mundo....

In this paper, we consider only the context free NER problem for the Portuguese language.

For the English language, NER is one of the basic tasks in complex NLP systems. In [18], a Hidden Markov Model-based Chunk Tagger is used. The performance of the proposed system shows a F-score above the 94% threshold. In [16], a decision tree built with the C4.5 algorithm is applied to the Portuguese and Spanish NER problem of identifying the boundaries of the entities.

In [12], a different approach is used for this problem. Rules of form and similarity are used to identify named entities with the aide of the REPENTINO gazetteer [13]. Palavras-NER [1], the best named entity extractor reported for the Portuguese, is based in a full parser and achieves a F-score of 80.61% in the Golden Collection of HAREM [14] for all entities.

In HAREM, the problem of finding named entities is slightly different, since every capitalized word is assumed as a NE. Here, however, we only consider proper nouns as candidates to NE. Hence, a direct comparison between our findings and HAREM's benchmarks is not possible. Nevertheless, the results show some consistent characteristics and indicate that our ML solutions are very competitive.

Here, we present our findings on seven Machine Learning modeling approaches to solve Portuguese NER. In the first one, a greedy algorithm with the help of a gazetteer is used. In the second, a pure HMM model is evaluated. In the third, we test the same HMM model with the greedy algorithm as an initial classifier. In the next two experiments, we use TBL in combination with either the greedy algorithm or our HMM model. In the sixth, a pure SVM model is evaluated. And in the last one, we test the SVM model with the help of the greedy algorithm. These Machine Learning algorithms were chosen because they have great results in similar natural language processing task.

The performance of each modeling approach is empirically evaluated. The SVM-based extractor shows a 88.11% F-score, which is our best observed value, slightly better than TBL. This is very competitive when compared to state-of-the-art extractors for similar Portuguese NER problems. Our HMM has reasonable precision and accuracy and does not require any expert knowledge. This is an advantage of our HMM over the other approaches. The experimental results suggest that a Machine Learning (ML) approach can be useful in Portuguese NER. They also indicate that HMM, TBL and SVM perform well in this natural language processing task.

The paper is organized as follows. In the next section, we describe the basic ML techniques that are used in our modeling, that is, HMM, TBL and SVM. In section 3, we describe our modeling strategies for Portuguese NER. In section 4, we summarize our empirical findings. Finally, in section 5, we present our concluding remarks.

2 Techniques

Our approaches to NER use three basic machine learning techniques: Hidden Markov Models, Transformation-Based Learning and Support Vector Machines.

2.1 Hidden Markov Models

Hidden Markov Modeling (HMM) [11] is a powerful probabilistic framework used to model sequential data. HMM is widely used in Natural Language Processing tasks such-as part-of-speech (POS) tagging, text segmentation and

voice recognition.

In HMM, we have two basic concepts: observations and hidden states. In NLP tasks, the sequence of words that form a sentence are usually considered as the observed data, and the states represent semantic information related to the sentence. The HMM parameters are set to maximize the log-likelihood between the sentence and the semantic information.

With the HMM parameters, one can easily evaluate the best state sequence using the *Viterbi* algorithm [7]. The best sequence of states is the one that has the highest log-likelihood with the given sentence. The states obtained can, then, be mapped to the semantic tags generating a NLP classifier. The success in the classification process is highly dependent on the choice of states and their corresponding observables. Nevertheless, generic models can perform quite nicely in some problems.

2.2 Transformation Based Learning

Transformation Based error-driven Learning (TBL) is a symbolic machine learning method, introduced by Eric Brill [2]. It is also used in several important NLP tasks, such as part-of-speech (POS) tagging [3], parsing, prepositional phrase attachment and phrase chunking, achieving state-of-the-art performance in many of them.

The main idea in a TBL algorithm is to generate an ordered set of rules that can correct tagging mistakes in the corpus, which have been produced by an initial guess classification process called, Baseline System (BLS). The rules are generated according to a list of templates given by the developer, which are meant to capture the relevant feature combinations to the problem by successively correcting the mistakes generated by the BLS and also by TBL itself.

This learning algorithm is a mistake-driven greedy procedure which, iteratively, acquires a set of transformation rules. The TBL algorithm can be described as follows:

1. The initial guess classification is used to evaluate an un-tagged version of the training corpus;

2. The results of the classification are evaluated by a comparison with the tagged version of the corpus and, whenever an error is found, all rules that can correct it are generated by instantiating the rule templates with the current token feature's context. A new rule may correct tagging errors, but can also generate some other errors by changing correctly tagged tokens;

3. The rules' scores, that is, the number of errors repaired minus number of errors created, are computed. If there is no rule above an arbitrary threshold score value, the learning process is stopped;

4. The rule with best score is selected, stored in the ordered set of learned rules and applied to the whole corpus;

5. The process is retaken in step 2.

2.3 Support Vector Machines

Support Vector Machines (SVMs) were developed by Vapnik et al. [17] as a method for learning linear and, through the use of kernels, non-linear rules. They have successfully been used for isolated handwritten digit recognition, object recognition, speaker identification, charmed quark detection, face detection in images, and text categorization [4].

SVMs use geometrical properties in order to compute the hyperplane that best separates a set of training examples. 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 standard SVM is intended to solve binary classification problems. However, they can also solve multi-class classification problems by decomposing them in several binary problems. One possible decomposition technique is the one-against-one approach, in which $\frac{k(k-1)}{2}$ classifiers are constructed and each one trains data from two different classes. In classification, a voting strategy is used: each binary classification is considered to be a voting where votes can be cast for

all data points. In the end, data points are designated to be in a class with maximum number of votes.

3 NER Modeling

3.1 Corpus

We use a corpus with 2,100 sentences taken from the SNR-CLIC corpus [8], already annotated with part-of-speech tags. The ner-tags are manually added following the Active Learning (AL) [6] scheme described below

a small quantity of sentences are randomly chosen and manually tagged;

repeat

- using the current manually tagged sentences, a classifier is built;
- the remainder of the corpus is classified using the current classifier, and ranked according to a classification confidence measure;
- the worst n sentences according to the confidence measure are selected, manually tagged and incorporated to the current corpus;

until the example set is large enough.

Through a preprocessing step, all consecutive proper nouns appearing in the corpus are concatenated in order to generate a single entity. Similarly, all proper nouns appearing in the corpus connected by a preposition or an article are also concatenated. Also some Portuguese contractions, mainly prepositions plus articles are splitted. For instance, the following transformation in the corpus is observed:

- um informe **do** *Conselho Nacional da População* .
- um informe **de o** *Conselho=Nacional=da=População* .

The following tag set is used to enconde NER: {PER, ORG, LOC, O}. The PER, ORG and LOC tags are used to respectively tag the entities Person, Organization and Location. Whereas the O tag is used otherwise.

Examples of the encoding are shown below.

- ... presidente/O de/O a/O instituição/O ,/O **Lewis=Preston**/PER ./O
- ... de/O o/O sudoeste/O de/O os/O **EUA**/LOC onde/O ...
- ... ,/O a/O **Mazda**/ORG rompeu/O negociações/O com/O ...

With these tagging conventions, we find 3,325 NE examples in the corpus.

3.2 Baseline System

The Baseline System (BLS) is an initial classifier. It is usually based on a set of simple heuristics susceptible to errors that should be identified and corrected.

It is also an essential component in the TBL approach, since it provides the initial classification guess for the TBL error correcting scheme.

For Portuguese NER, our BLS was built in order to capture the immediate knowledge available for the task. It can be described by the four main components below.

- *Location Gazetteer* - a gazetteer of names of continents, countries and their capitals, states and their capitals from Brazil extracted form the Web;
- *Person Gazetteer* - a gazetteer of popular English and Portuguese baby names extracted from the Web;
- *Organization Gazetteer* - a gazetteer of the top 500 enterprises measured by gross revenue extracted from the Fortune magazine;
- *Preposition Heuristic* - a greedy heuristic based on the last preposition previous to a proper noun. Based on a small portion of the corpus, we create a simple rule relating each preposition to the entity that most followed it. For instance, every proper noun that follows the preposition *em* is tagged as a Location.

word	pos-tag	ner-tag	hmm state
A	ART	O	O
informação	N	O	O
é	V	O	ODPER
de	PREP	O	OBPER
Norbert=Gmuer	NPROP	PER	PER
,	,	O	OAPER
60	NUM	O	OCPER
,	,	O	O
presidente	N	O	O
de	PREP	O	ODCOM
a	ART	O	OBCOM
Ciba-Geigy	NPROP	COM	COM
,	,	O	OACOM
que	PRO-KS-REL	O	OCCOM
comemora	V	O	O
,	,	O	O
...			

Table 1: Relabel procedure for HMM state generation

Whenever a proper noun is found, we apply the BLS in the order above until a match is made.

3.3 HMM Model

Our HMM based models are very similar to the one proposed in [9, 10]. A simple way to model NER using HMM is to use the ner-tags (PER, ORG, LOC, O) as the hidden states and the pos-tags as the observations. Each sentence is then mapped to its pos-tag sequence. The HMM probabilities are estimated by the relative frequencies obtained through feature counting in the training data. A especial symbol, _UNKNOWN_, which can be emitted in any state, is created to deal with unobserved data.

When applying the model to classify an instance, the sentence is first mapped to its pos-tag sequence. Next, the Viterbi algorithm is applied to find the best ner-tag sequence.

This simple model is quite inefficient. Since it has a small number of states, it does not take advantage of the inherent local structure of the sentence near to a NE. This limitation can be reduced by the introduction of new enhanced states, generated online and based on the tags manually introduced. Hence, the following tags are used:

- OAT, a tag immediately after a given T tag;
- OBT, a tag immediately before a given T tag;
- OCT, a tag immediately after a OAT tag;
- ODT, a tag immediately before a OBT tag;
- OET, a tag immediately before and after the same T tag;
- OHT, a tag immediately after a given T tag and before another T' tag;

where T tag is one of the ner-tags. For instance, for the PER tag, we obtain the following new tags: OAPER, OBPER, OCPER, ODPER, OEPER and OHPER. As we can map a tag to two or more different states, we add an extra relabeling procedure, which uses an order of preference for the states. An example of this mapping is shown in Table 1.

With this relabeling procedure we enhance our results, as a consequence of the O tag refining. Now we can improve our model by taking advantage of the available lexical information. Normally, in NLP tasks, treating all prepositions as the same can lead to many errors. Whenever a preposition appears in a sentence, we replaced it by its corresponding lexical information.

To help the classification process, the BLS can be evaluated before the HMM classification (BLS + HMM) and its evaluated non null ner-tags used instead of the pos-tags as the HMM observations.

3.4 TBL Model

To apply TBL, some of its components must be specialized to the current task. Our key modeling decisions are described below.

Initial Classification - we tested two different initial classifiers: the Baseline System, and the HMM Model.

Templates - several sets of templates were tested in combination with the features word, pos and ner-tags. The best template set that we found consists of some generic templates, together with some specific ones. The generic templates use a combination of the features in a neighborhood of two tokens. On the other hand, the specific templates look for specific patterns, mainly for sequences of named entities, prepositions, articles, precedent verbs, adverbs and nouns.

Examples that illustrate our template set are:

1 ner[0] word[-1] pos[-1] word[-2] pos[-2];

2 ner[0] word[-3,-1]_where{pos=ART} pos[-1];

3 ner[0] ner[-2,-2]_where{ner=LOC} pos[-1].

The first template creates good rules whenever a mistake can be corrected by using the two previous word and pos-tags. The second one generates rules based on the precedent article, the word[-3,-1]_where{pos=ART} term instantiates previous words that are tagged as article in the pos-tag. The last one tries to catch sequences of Location entities.

For instance, when training TBL with the full corpus. The two top score rules are:

1 ner[0]=COM pos[-1]=PREP → ner=PER;

2 ner[0]=COM word[-3,-1]_where{pos=ART}=o pos[-1]=N → ner=PER.

3.5 SVM Model

SVM is designed to classify data points in a vector space. Therefore, our model needs to map each token in the corpus to a n-dimensional vector. The following paragraphs describe this conversion process.

First, similarly to what is described in [15], we select which neighbor tokens to use by defining a window of size 5. This means that the classification of a token takes into account the token itself, the 2 preceding tokens and the 2 proceeding tokens. After this, we decide which features are interesting. For each relevant neighbor token we chose the following features: the word, its pos-tag and an initial classification, when provided.

We observe that all the chosen features store categorical data. Therefore, we have to represent each of them as a vector of zero-one variables where each coordinate refers to a possible feature value. In such vector the coordinate related to the observed feature value is set to one, while all the others set to zero.

Finally, we obtain an unique vector to represent each token in the corpus by concatenating all the vectors described above. When an initial classification is provided, such unique vectors have 44,844 coordinates each; otherwise, they have 44,824 coordinates each. Just a few of these coordinates have non-zero value. Hence, we adopted the sparse format representation used in [5].

SVM can learn non-linear classification models through the use of kernel functions. However, we train a *soft margin* linear classification model which accepts an amount of training errors. We chose this model because it takes less time to be trained, while leading to fairly good results.

4 Experimental Results

Validation of the chosen approaches is conducted with a 10 holdout cross-validation. For each sample, the corpus is randomly divided into 70% of the sentences for training and 30% for test. In each iteration of the cross-validation, each machine learning approach is trained using the training set, the result of the training process is a classifier that is applied to the test set. The results reported are the means obtained through the 10 iterations.

	Precision (%)			Recall (%)			F-score (%)		
Experiment	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min
BLS	73.11	-	-	80.21	-	-	76.50	-	-
Plain HMM	65.22	66.32	63.33	67.10	69.25	65.16	66.14	67.76	64.23
BLS + HMM	77.36	79.39	72.78	78.79	82.04	75.45	78.07	80.18	74.09
HMM + TBL	75.88	78.31	70.68	74.67	78.01	70.82	75.27	78.01	70.75
BLS + TBL	85.84	87.61	84.01	88.74	90.08	87.37	87.26	88.58	85.66
Plain SVM	83.70	84.80	81.91	86.30	87.50	85.02	84.98	86.02	83.44
BLS + SVM	**86.98**	89.34	85.57	**89.27**	91.95	87.72	**88.11**	90.63	87.07

Table 2: Results for the seven experiments.

We assumed F-score as our key statistics. **F-score** is the harmonic mean between precision and recall. **Precision** informs how many good classifications the model predicted amongst all predictions made. **Recall** informs how many good classifications were predicted amongst all true entities.

Here, we report the results we found on the seven most important experiments. Their corresponding settings are described below

1 **BLS**: the application of the Baseline System.

2 **Plain HMM**: HMM with the addition of the enhanced states.

3 **BLS + HMM**: HMM with the Baseline System as the Initial Classifier.

4 **HMM + TBL**: TBL with previous HMM Extractor as the Initial Classifier.

5 **BLS + TBL**: TBL with Baseline System as the Initial Classifier.

6 **Plain SVM**: soft margin linear SVM with window size five.

7 **BLS + SVM**: SVM with the help of the Baseline System.

The HMM and TBL algorithms used implementations developed in LEARN laboratory at PUC-Rio. The SVM algorithm used a public implementation of SVM called libsvm [5].

Table 2 shows the results for each experiment. Bold values indicate the best statistic in each column.

Plain HMM shows very good initial F-scores, although below BLS, since it has very little expert knowledge, indicating that it is a good initial alternative when no specific domain knowledge is available. BLS + SVM outperformed the others integrating SVM with a good heuristic and the help of a little gazetteer, this was slightly better than the TBL approach.

The most common errors made by our best extractors (SVM and TBL) are proper nouns that are preceded by:

- a definite article, as in "O sub-prefeito de a **Barra=da=Tijuca** ... ", which are identified as Organizations
- a noun, as in " ... de o partido **Sakigake** ... ", which are identified as People
- the preposition *em*, as in "Em Furnas (Rio de Janeiro) os ...", which are identified as Locations

Some of these errors are related to the possible roles the same entity can have in different contexts, what generates an ambiguity very hard to distinguish without extra information.

In Table 3, we show the best results for specialized NE extractors. We build one specific extractor for each NE category.

We notice here that the easiest NE to be recognized is Location, mainly because of the easiness of building an efficient gazetteer of this kind of entity. On the other hand, the most difficult one is Organization, mainly because many entities can take an Organization value in some contexts.

		Precision (%)			Recall (%)			F-score (%)		
Experiment	Entity	Mean	Max	Min	Mean	Max	Min	Mean	Max	Min
BLS + TBL	PER	**87.78**	90.69	83.95	81.13	83.84	76.96	84.28	85.51	82.80
	ORG	75.35	79.08	72.98	**91.93**	94.62	89.47	82.79	85.16	81.34
	LOC	93.10	96.40	90.69	81.85	86.27	75.89	87.08	89.54	83.48
BLS + SVM	PER	87.71	89.89	84.13	**89.15**	92.82	86.33	**88.41**	90.50	85.48
	ORG	**84.36**	89.56	80.79	88.52	91.18	85.06	**86.36**	88.17	83.50
	LOC	**96.18**	98.60	93.88	**82.09**	85.95	78.08	**88.55**	90.63	86.38

Table 3: Best specialized extractors for each NE category.

5 Concluding Remarks

This work shows some promising ML approaches to Portuguese NER.

The SVM and TBL methods appear as an excellent alternative when linguistics experts can provide their expertise to the system, either by building a specific BLS, by choosing the right features to use or by formulating the templates that capture the domain knowledge. This can be viewed as the premium price solution. On the other hand, the plain HMM alternative gives good values for precision and accuracy, without the support of any specific linguistic intelligence. This can be viewed as the cheap solution.

Our SVM approach outperformed the other solutions, showing a 88.11% F-score, which is slightly better than the one obtained by PALAVRAS-NER. Although this comparison cannot be fully taken into account, since there are some differences in the definition of the two problems, the results of the NER evaluation are similar when comparing each one of the entities separately. In both cases, the best algorithms dealt better with Locations than Organizations, with People showing medium difficulty.

The experimental results suggest that extra contextual information can be used to increase performance. For instance, extra syntactic information can be used in the definition of the TBL templates or as new features for the SVM.

A next step in this work is to evaluate the same model using the Golden Collection from HAREM [14]. Preliminary straightforward tests do not show good performance, since there are some major differences in the definition of NE. For instance, in the HAREM NER problem, any capitalized word must be classified as an entity, our classifiers, on the other hand, only consider proper nouns as candidates for an entity.

We showed that our extractors can have a great benefit in the automatic construction of entity gazetteers that could aide various other NLP tasks. We shall continue tuning the parameters and enhancing our template system to catch other kinds of named entities, as well as to be able to evaluate its performance for the Golden Collection.

References

[1] Eckhard Bick. Functional aspects in portuguese ner. In *Proc. of the 7th Intl. Workshop, PROPOR*, Lecture Notes in Artificial Inteligence. Springer-Verlag, Heidelberg, 2006.

[2] Eric Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Process*, Trento, Italy, 1992. Association for Computational Linguistics.

[3] Eric Brill. Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Computational Linguistics*, 21(4), 1995.

[4] Christopher J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):121–167, 1998.

[5] C. Chang and C. Lin. Libsvm: a library for support vector machines, software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm, 2001.

[6] David A. Cohn, Zoubin Ghahramani, and Michael I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky,

and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

[7] G. D. Forney. The viterbi algorithm. In *Proceedings IEEE*, volume 61, March 1973.

[8] M. C. Freitas, M. Garrão, C. Oliveira, C. N. Santos, and M. Silveira. A anotação de um corpus para o aprendizado supervisionado de um modelo de sn. In *Proceedings of the III TIL / XXV Congresso da SBC*, São Leopoldo - RS, 2005.

[9] Maria Claudia Freitas, Julio Cesar Duarte, Cicero Nogueira Santos, Ruy Luiz Milidiú, Violeta Quental, and Raúl Rentería. A machine learning approach to the identification of appositives. In *Ibero-American AI Conference – Proc. of the 10th Intl. Conference, IBERAMIA'2006*, 2006.

[10] Ruy Luiz Milidiú, Julio Cesar Duarte, Cícero Nogueira Santos, and Raúl Rentería. Semi-supervised learning for portuguese noun phrase extraction. In *Proc. of the 7th Intl. Workshop, PROPOR*, Lecture Notes in Artificial Inteligence. Springer-Verlag, Heidelberg, 2006.

[11] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. In *Proc. of IEEE*, pages 257–286, 1989.

[12] Luis Sarmento. Siemês: a named entity recognizer for portuguese. In *Proc. of the 7th Intl. Workshop, PROPOR*, Lecture Notes in Artificial Inteligence. Springer-Verlag, Heidelberg, 2006.

[13] Luis Sarmento and Luís Cabral Ana Sofia Pinto. Repentino a wide-scope gazetteer for entity recognition in portuguese. In *Proc. of the 7th Intl. Workshop, PROPOR*, Lecture Notes in Artificial Inteligence. Springer-Verlag, Heidelberg, 2006.

[14] Nuno Seco, Diana Santos, Rui Vilela, and Nuno Cardoso. A complex evaluation architecture for harem. In *Proc. of the 7th Intl. Workshop, PROPOR*, Lecture Notes in Artificial Inteligence. Springer-Verlag, Heidelberg, 2006.

[15] Thamar Solorio and Aurelio López-López. Learning named entity classifiers using support vector machines. In *CICLing*, pages 158–167, 2004.

[16] Thamar Solorio and Aurelio López-López. Learning named entity recognition in portuguese from spanish. In *CICLing*, pages 762–768, 2005.

[17] V. Vapnik. *Statistical Learning Theory*. Wiley, New York, NY, 1998.

[18] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480, Morristown, NJ, USA, 2001. Association for Computational Linguistics.