PUC

ISSN 0103-9741

Monografias em Ciência da Computação
nº 11/07

Probabilistic Classifications with TBL

Cícero Nogueira dos Santos
Ruy Luiz Milidiú

Departamento de Informática

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO

RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22451-900

RIO DE JANEIRO - BRASIL

Monografias em Ciência da Computação, No. 11/07 ISSN: 0103-9741
Editor: Prof. Carlos José Pereira de Lucena June, 2007

Probabilistic Classifications with TBL

Cícero Nogueira dos Santos, Ruy Luiz Milidiú

nogueira@inf.puc-rio.br, milidiu@inf.puc-rio.br

Abstract. The classifiers produced by the Transformation Based error-driven Learning (TBL) algorithm do not produce uncertainty measures by default. Nevertheless, there are situations like active and semi-supervised learning where the application requires both the sample's classification and the classification confidence. In this paper, we present a novel method which enables a TBL classifier to generate a probability distribution over the class labels. To assess the quality of this probability distribution, we carry out four experiments: cross entropy, perplexity, rejection curve and active learning. These experiments allow us to compare our method with another one proposed in the literature, the TBLDT. Our method, despite being simple and straightforward, outperforms TBLDT in all four experiments.

Keywords: Machine Learning, Transformation Based Learning, Natural Language Processing.

Resumo. Os classificadores produzidos pelo algoritmo Aprendizado Baseado em Transformações (TBL) não produzem, por padrão, medidas de incerteza. No entanto, existem aplicações que requerem tanto a classificação da amostra como a confiança dessa classificação, como por exemplo o Aprendizado Ativo e o Aprendizado Semi-supervisionado. Esse trabalho apresenta um novo método o qual habilita o classificador TBL a estimar distribuições de probabilidades sobre as etiquetas de classes. A qualidade destas distribuições de probabilidades é mostrada a partir de quatro experimentos: entropia cruzada, perplexidade, curva de rejeição e aprendizado ativo. Esses experimentos permitem ainda a comparação do método proposto com outro método existente na literatura, o TBLDT. Nosso método, apesar de ser mais simples, obteve melhores resultados do que o TBLDT nos quatro experimentos.

Palavras-chave: Aprendizado de Máquina, Aprendizado Baseados em Transformações, Processamento de Linguagem Natural.

1 Introduction

Since the last decade, Machine Learning (ML) has proven to be a very powerful tool to enable effective Natural Language Processing (NLP). ML has been applied to central NLP problems such as: part-of-speech tagging, word-sense disambiguation, shallow parsing and prepositional phrase attachment ambiguity resolution. Most of the more successful ML algorithms can be roughly divided into two groups: rule-based and probabilistic. One of the main advantages of the rule-based methods is that, in general, the outcome of the learning process is a small set of rules that can be interpreted by humans.

On the other hand, probabilistic algorithms have the advantage of including uncertainty measures in their outputs. These uncertainty measures are useful for situations where the application requires both the sample's classification and the classification confidence. Semi-supervised algorithms and Active Learning are methods that require the samples' classification confidence. In Semi-supervised algorithms like Co-training [Blum and Mitchell, 1998], the uncertainty measures are used to select samples where the classifier is more confident. In Active Learning, where the objective is to minimize the tagging effort, the classification confidence is used to select the more informative samples to be manually tagged.

Transformation Based error-driven Learning (TBL) is a successful symbolic machine learning method introduced by Eric Brill [Brill, 1995]. It has since been used for several Natural Language Processing tasks, such as part-of-speech (POS) tagging [Brill, 1995], text chunking [Ramshaw and Marcus, 1999], spelling correction, portuguese noun-phrase extraction [Milidiú et al., 2006] and portuguese appositive extraction [Freitas et al., 2006], achieving state-of-the-art performance in many of them. The classifiers generated by TBL, by default, do not produce uncertainty measures. In [Florian et al., 2000], Florian et al. propose a method that provides an uncertainty measure to TBL classifiers.

Here, we present a novel method that enables a TBL classifier to generate a probability distribution over the class labels. To assess the quality of this probability distribution, we carry out four experiments: active learning, rejection curve, perplexity and cross entropy. These experiments allow us to compare our method with the TBLDT algorithm proposed by Florian et al. [Florian et al., 2000]. Our method, despite being simple and straightforward, outperforms TBLDT in all four experiments.

The remainder of the paper is organized as follows. In section 2, we describe the TBL algorithm. In section 3, we show our proposed method. In section 4, the experimental design and the results are reported. Finally, in section 5, we present our concluding remarks.

2 Transformation Based Learning

In a classification problem setup, the application defines which feature is to be learned. This feature is represented by a set of class labels Y. For instance, in the case of part-of-speech tagging, Y is the POS tagset.

TBL uses an error correcting strategy. Its main scheme is to generate an ordered list of rules that correct classification mistakes in the training set, which have been produced by an initial guess.

The requirements of the algorithm are:

- two instances of the training set, one that has been correctly labeled with the Y's class labels, and another that remains unlabeled;

- an initial classifier, the baseline system, which classifies the unlabeled training set by trying to guess the correct class for each sample. In general, the baseline system is based on simple statistics of the labeled training set; and
- a set of rule templates, which are meant to capture the relevant feature combinations that would determine the sample's classification. Concrete rules are acquired by instantiation of this predefined set of rule templates.

The learning method is a mistake-driven greedy procedure that iteratively acquires a set of transformation rules. The TBL algorithm can be depicted as follows:

1. Starts applying the baseline system, in order to guess an initial classification for the unlabeled version of the training set;
2. Compares the resulting classification with the correct one and, whenever a classification error is found, all the rules that can correct it are generated by instantiating the templates. This template instantiation is done by capturing some contextual data of the sample being corrected. Usually, a new rule will correct some errors, but will also generate some other errors by changing correctly classified samples;
3. Computes the rules' scores (errors repaired - errors created). If there is not a rule with a score above an arbitrary threshold, the learning process is stopped;
4. Selects the best scoring rule, stores it in the set of learned rules and applies it to the training set;
5. Returns to step 2.

When classifying a new sample set, the resulting sequence of rules is applied according to its generation order.

3 Probability Estimation with TBL Classifiers

One of the TBL classifiers' disadvantages is that they make hard decisions. When a TBL rule set is applied to a sample set S, each sample $s \in S$ receives one class label $y \in Y$, $[s \leftarrow y]$. Since these hard decisions do not have an associated probability, they give no hint about the classification quality.

On the other hand, probabilistic classifiers make soft decisions by assigning to each sample a probability distribution over all possible classes. There are many situations where this kind of classification is useful. One of these situations is in pipeline systems, such as an information extractor system that performs named entity extraction on the output of a probabilistic part-of-speech tagging. Another case is the semi-supervised learning which, in general, requires uncertainty measures to select samples for which the classifier is more confident.

3.1 The proposed method

The method proposed in this paper enables a TBL classifier to make soft decisions. Using this method, when a TBL rule set is applied to a sample set S, each sample $s \in S$ receives a probability distribution over the class labels Y, $[s \leftarrow P(Y|s)]$, instead of a single class label $y \in Y$. The method relies in the use of the training set to estimate a probability model associated with the TBL classifier. The procedure to estimate such a probability model is based in the notion of equivalence classes.

We call an *equivalence class* a set of samples that share some specific characteristics. In this work, we use two characteristics to group the samples into equivalence classes:

- the class label assigned by the initial classifier; and
- the rules that change the samples.

Using these two information pieces, we can create two types of equivalence classes: (a) a set of samples that have the same initial class label and are changed by the same rules; or (b) a set of samples that have the same initial class label and no rules are applied to them.

For each equivalence class, we compute its probability distribution by using maximum likelihood estimation. This is done by using Eq. 1.

$$P(y|e) = \frac{count(e, y)}{count(e)} \;\; \forall y \in Y \tag{1}$$

where e is an equivalence class; $count(e, y)$ is the number of samples in e whose class label is y; and $count(e)$ is the number of samples in e.

We also estimate the probability distribution of a dummy equivalence class e_p that includes all the samples in the training set. We call the distribution $P(Y|e_p)$ the *prior class labels distribution*.

The whole process used to construct the probability model associated with a TBL rule set is shown in Fig. 1. The use of the probability model can be very efficient if it is stored in a hash table. The [key, value] pairs of the hash table are formed by the equivalence classes IDs and their respective probability distributions, [*initial class label + rules applied*, $P(Y|e)$].

Probabilistic classifications using a TBL classifier and a probability model associated with it must be done as described below.

1. Classify the samples using the TBL rule set. Record, for each sample, the class label assigned by the initial classifier and the rules applied to it;
2. Using the hash table containing the probability model, assign the following items for each sample:
   - the probability distribution whose key is the initial class label of the sample and the rules applied to it; or
   - if the hash table does not contain such a key, assign to the sample the distribution $P(Y|e_p)$.

3.2 Equivalence class partitioning

When creating a probability model using the proposed method, some equivalence classes could be very dense, mainly the ones formed by samples that are not changed by any rule. These cases are inefficient in terms of probability estimation. In fact, if a large portion of the training set is in the same equivalence class, then the distribution assigned to it is nearly the prior class distribution. For instance, in the case of base noun-phrase identification showed in Sect. 4 almost 50% of the the training set falls in the same equivalence class.

```
Algorithm: GenerateProbabilityModel

    Input:
        • $R$: a TBL rule set;
        • $T$: the training set used in the generation of the rule set $R$.

    Do:
        1. apply the rule set $R$ to the training set $T$. For each sample $t \in T$,
           record the class label assigned by the initial classifier and the rules
           applied to it;
        2. create equivalence classes by using the information recorded in
           step 1;
        3. estimate the probability distribution for each equivalence class by
           using Eq. 1;
        4. estimate the prior class labels distribution $P(Y|e_p)$ by using Eq. 1.

    Output:
        • the probability model associated with the rule set $R$.
```

Figure 1: Algorithm used to construct a probability model associated with a TBL rule set

To overcome this inefficiency, we use an *auxiliary feature* to partition the equivalence classes that are composed by samples not changed by any rule. This feature must be suitably chosen for the classification problem being solved. In the two classification problems shown in Sect. 4, we use the POS tag as the auxiliary feature. In the case of base noun-phrase identification, for instance, the densest equivalence class is partitioned into 20 new equivalence classes whose ID includes the samples' POS tag.

3.3 Smoothing

Smoothing is very useful when estimating probability models for sparse data. Smoothing deals with events that have been observed zero times and also tends to improve the accuracy of the model. In this work, we apply two smoothing techniques to the probability model generated from a TBL classifier.

3.3.1 Additive Smoothing

We use the *plus-delta* version [Chen and Goodman, 1996] of the additive smoothing, which is computed by using Eq. 2.

$$P(y|e) = \frac{count(e,y) + \delta}{count(e) + \delta \cdot |Y|} \quad \forall y \in Y \tag{2}$$

where e is an equivalence class; δ is a number between 0 and 1; $count(e,y)$ is the number of samples in e whose class labels are y; $count(e)$ is the number of samples in e; and $|Y|$ is the number of class labels.

3.3.2 Backoff Smoothing

The idea of this technique is to smooth a most specific estimate $P(y|e_1)$ with a less specific one $P(y|e_2)$ by computing a mixture of the two estimates [Kalt, 2005]

$$\hat{P}_{1,2}(y|e_1) = \lambda P(y|e_1) + (1-\lambda)P(y|e_2) \tag{3}$$

using a mixing coefficient λ. The mixing coefficient varies between 0 and 1 according to our confidence in the first estimate. We say that the estimate $P(y|e_1)$ is most specific than $P(y|e_2)$ because the equivalence class e_1 contains less samples than e_2, and $e_1 \subset e_2$.

Given a sequence of estimates $(P(y|e_1), P(y|e_2), ..., P(y|e_k))$, where $P(y|e_1)$ is the most specific one and $P(y|e_k) = P(y|e_p)$ is the less specific one, we can recursively compute a linear combination of these estimates by

$$\begin{aligned} \hat{P}_k(y|e_k) &= P(y|e_p) = P(y) \\ \hat{P}_i(y|e_i) &= \lambda_i P(y|e_i) + (1-\lambda_i)\hat{P}_{i+1}(y|e_{i+1}) \end{aligned} \tag{4}$$

The estimate $\hat{P}_1(y|e_1)$ in this sequence is a linear combination of all k estimates. To compute the λ_i coefficient used in Eq. 4 we use the method described in [Collins, 2003]. In this method, λ_i is a function of the samples associated with the equivalence class e_i and is computed using Eq. 5.

$$\lambda_i = \begin{cases} \frac{count(e_i)}{count(e_i)+c\cdot div(e_i)} & \text{if } count(e_i) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

where $count(e_i)$ is the number of samples in e_i; c is an adjustable parameter which is tuned using held out data; and $div(e_i)$ is the number of distinct outcomes observed in e_i:

$$div(e_i) = |\{y \text{ s.t. } e_i \text{ contains at least a sample whose class label is } y\}|$$

We generate intermediary estimates in order to apply backoff smoothing to our probability model. These intermediary estimates are obtained by creating less specific equivalence classes from the ones generated in the process[1].

We partition the sample set into equivalence classes by using the following as common characteristics: the initial class label and the rules applied to the samples. We can construct less specific equivalence classes by gradually including more samples in an existing one. This is done by reducing the common information shared between the samples. To construct intermediary equivalence classes we gradually reduce the ID (initial class label and rules) of the existing one until it contains all the samples in the training set (e_p). This process is shown in the following example.

$$\begin{aligned} e_1 : ID &= [y, R1, R2, R3] \\ e_2 : ID &= [y, R1, R2] \\ e_3 : ID &= [y, R1] \\ e_4 : ID &= [y] \\ e_5 : ID &= e_p \end{aligned}$$

where ID is the set of characteristics that the samples in the equivalence class e_i have in common; y is the initial class label; and $\{R1, R2, R3\}$ are rules.

[1]Described in Sect. 3.1

After the intermediary equivalence classes are generated, we estimate their probability distribution using Eq. 1 and then we apply backoff smoothing. In the previous example, the probability estimation $\hat{P}(Y|e_1)$ assigned to the equivalence class e_1 is a mixture of the five estimates.

3.4 Related work

Florian et al. [Florian et al., 2000] proposed a method which enables TBL classifiers to produce soft decisions. The method involves dividing the training set into equivalence classes and computing distributions over each equivalence class. The way used to divide the training set is to transform the TBL rule set into a decision tree. Once the decision tree is constructed and applied to the training set, the samples attached to the same leaf define an equivalence class. When a new sample set is to be classified, the decision tree is used.

The main difference between the method proposed here and the one of Florian et al. [Florian et al., 2000], is that we do not need to construct a decision tree. We only need to record the trace of the rules applied to each sample. Then, the rule application process remains the same. Two other differences are: (1) in [Florian et al., 2000], a further growth of the decision tree is used to partition the equivalence classes (leaves) that contains a large number of examples; and, (2) as a kind of smoothing, Florian et al. prunes the decision tree by removing the nodes that contain less examples than a given threshold.

4 Experiments

The effectiveness and quality of the proposed probabilistic extension to TBL are demonstrated by four experiments presented in this section. The experiments are performed on two tasks: text chunking and base noun phrase chunking. Text chunking consists in dividing a text into syntactically correlated of words. Base noun phrase chunking consists in recognizing non-overlapping text segments that consist of noun phrases (NPs).

The data used in the text chunking is the CoNLL-2000 corpus [CoNLL, 2000]. This corpus consists of sections 15-18 and section 20 of the Penn Treebank, and is pre-divided into 8936-sentence (211727 tokens) training set and a 2012-sentence (47377 tokens) test. This corpus is tagged with POS tags and with chunk tags. The data used in the base NP chunking is the one of Ramshaw & Marcus [Ramshaw and Marcus, 1999]. It is composed by the same texts as the CoNLL-2000 corpus. The difference is that the chunk tags only identify base NPs.

We have developed TBLprob which implements the probability estimation method described in Sect. 3. To compare our results with those produced by TBLDT, the method of Florian et al. [Florian et al., 2000], we use the fnTBL toolkit. fnTBL implements probability estimation using TBLDT - *conversion of TBL rules to Decision Tree*.

To perform a fair evaluation, we produce the TBL models by using only the fnTBL toolkit. Therefore, the same TBL rule sets are used to generate estimates using TBLprob and TBLDT methods. Only the Active Learning test is carried out with the rules generated by our system. The template set used in all the experiments, for both tasks, is the one proposed by Ramshaw & Marcus [Ramshaw and Marcus, 1999].

4.0.1 Tuning the smoothing parameters:

The usual way of tuning the smoothing parameters is to maximize the likelihood of a held-out data set. According to Kalt [Kalt, 2005], this is equivalent to minimizing the cross entropy of the model with a held-out set. In our case, we use the *conditional cross entropy* which is defined by

$$H_c(Y|X) = -\sum_{x \in X} q(x) \cdot \sum_{y \in Y} q(y|x) \log_2 p(y|x) \tag{6}$$

where X is the test set (held-out), Y is the set of class labels, q is the probability distribution on the test set and p is the probability distribution estimated on the training set. Since the distribution $q(x)$ is unknown, we estimate the conditional cross entropy by computing an "empirical" expectation [Kalt, 2005]:

$$H_c(Y|X) \approx \frac{-1}{|X|} \sum_{(x,y)} \log_2 p(y|x) \tag{7}$$

In our experiments, we tune the smoothing parameters by performing a 5-fold cross validation with the training set. At each cross validation iteration, the best parameter value is selected. This value is the one that minimizes Eq. 7 when the probability model using it is applied to the corresponding test set. This process is done for both tasks, text chunking and base NP chunking. Table 1 displays the mean of the best values for each test fold. These values are used in all of the following experiments.

Table 1: Best smoothing parameters by technique and task.

	δ - Additive parameter	c - Backoff parameter
Text Chunking	0.05	1.0
Base NP Chunking	0.34	1.1

4.1 Cross Entropy and Perplexity

Cross entropy is typically used to measure the quality of a probability model. It takes into account the accuracy of the estimates as well as the classification accuracy of the system [Florian et al., 2000]. We utilize conditional cross entropy to compare our TBLprob estimates with the TBLDT estimates. The experiment uses Eq. 7 to estimate the cross entropy in the test set. The p distribution that appears in Eq. 7 is the one estimated by using the training set.

Perplexity is a measure closely related to cross entropy, defined as

$$P = 2^{H(Y|X)} \tag{8}$$

Table 2 presents the cross entropy and perplexity results for the various estimators. TBLprob using backoff smoothing outperforms the other schemes in both text chunking and base NP chunking. Nevertheless, the results using additive smoothing are very close to the ones using backoff.

The observed results indicate that the overall probability distribution of the TBLprob method better matches the true probability distribution. The method TBLprob without smoothing is outperformed by TBLDT only in the text chunking task. This occurs mainly because TBLDT has a pruning factor that works as a kind of smoothing.

Table 2: Cross Entropy and Perplexity results for the two tasks.

	Text Chunking		Base NP Chunking	
Classifier	Cross Entropy	Perplexity	Cross Entropy	Perplexity
TBLprob	0.3891	1.3096	0.1847	1.1366
TBLprob + backoff	**0.3350**	**1.2614**	**0.1668**	**1.1226**
TBLprob + additive	0.3388	1.2647	0.1672	1.1229
TBLDT	0.3818	1.3030	0.2338	1.1759

4.2 Rejection curve

There are many applications where a probabilistic classifier needs to reject samples with low confidence classification. In such situations, the probability $P(y|s)$ can be used as a confidence score of the sample s to be correctly classified with the class label y. Hence, it is very useful to assess the quality of the class probability estimates of a classifier. One way to do this is to compute a rejection curve which shows the percentage of correctly classified test cases whose confidence level exceeds a given value [Dietterich and Bakiri, 1995]. A rejection curve that increases smoothly demonstrates that the confidence scores produced by the classifier are reliable.

We use the entropy H of the class probability distribution assigned to a sample s (Eq. 9) as a confidence measure in the construction of the rejection curve. The higher the value of H, the more uncertain the classifier is of its classification.

$$H(p(Y|s)) = -\sum_{i=1}^{|Y|} p(y_i|s)\log_2 p(y_i|s) \tag{9}$$

The rejection curve is constructed by classifying the test set and then gradually rejecting the samples that have an entropy value greater than the current value of θ. θ takes values between the greatest entropy in the samples and 0. A point (x, y) in the rejection curve indicates that if the $x\%$ less confident samples are excluded, the remaining samples have an accuracy residual of $y\%$.

Figures 2 and 3 present the rejection curves for the text chunking and base NP chunking tasks, respectively. In both cases the TBLprob with smoothing clearly outperforms the TBLDT method at all rejection levels. In the case of text chunking, the two smoothing techniques generate classifiers with very similar behavior, producing a rejection curve that increases smoothly.

4.3 Active learning

This section presents experimental results which show the usefulness and effectiveness of the probabilities generated by TBLprob. The experiment is to use the generated probabilities as a measure of uncertainty in an Active Learning process. The objective of the *active learning* approach is to minimize the annotation effort by intelligently selecting the samples to be annotated. The active learning algorithm we use is the same as described in [Florian et al., 2000]:

1. Label an initial set of sentences of the corpus;
2. Train a TBL model and use the TBLProb method to obtain the class label probabilities on the rest of the corpus;

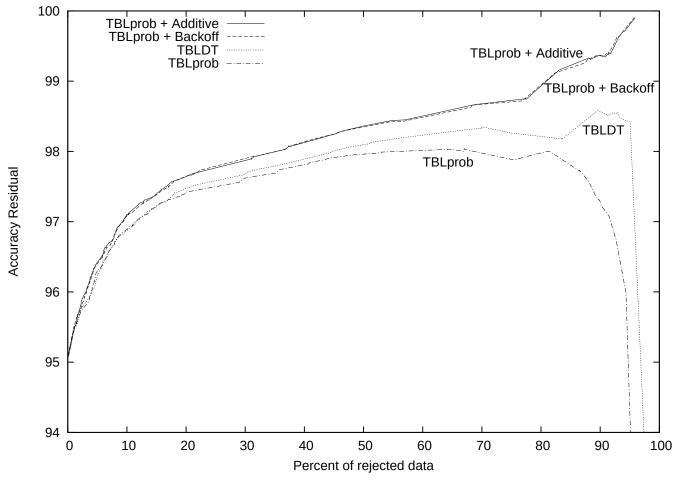


Figure 2: Rejection curve: Text chunking.

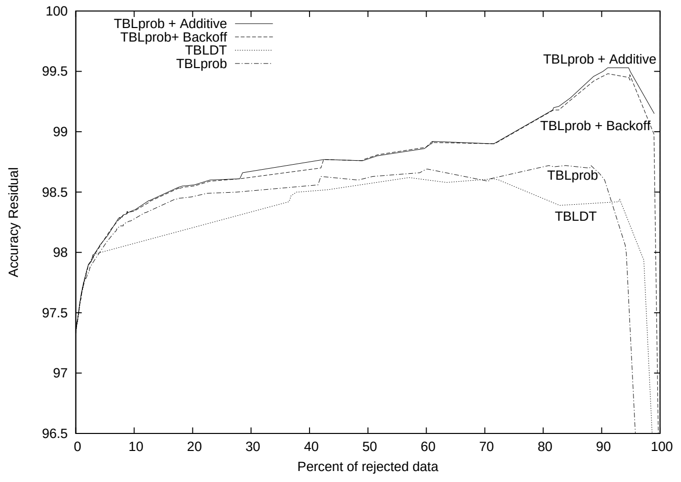


Figure 3: Rejection curve: Base NP chunking.

3. Choose T samples from the rest of corpus, specially the samples that optimize an evaluation function f, based on the class distribution probability of each sample;

4. Add the samples, including their correct class label to the training pool and retrain the system;

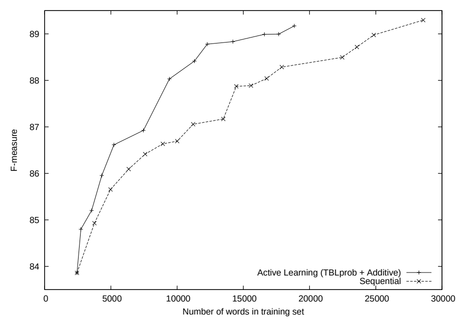
5. If the desired number of samples is reached, stop, otherwise repeat from Step 2.

We use the same evaluation function f which is used in [Florian et al., 2000]:

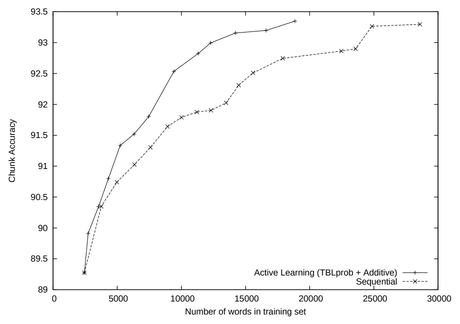
$$f(S) = \frac{1}{|S|} \sum_{i=1}^{|S|} H(Y|S, i) \tag{10}$$

where $H(Y|S, i)$ is the entropy of the class label probability distribution associated with the word index i in the sentence S.

Figure 4 shows the performance of the TBLprob classifier [2] for the text chunking task when trained with samples selected by using active learning and when trained with sequentially selected samples. The plots show the following information: (a) the F-measure and (b) the chunk accuracy versus the number of words in the annotated training set. Using less training data, the TBLprob classifier trained with active learning can obtain the same performance than the one trained with sequential data. Overall, the TBLprob + active learning can yield the same performance as the sequential system with 52% less data, a reduction greater than the one of 45% reported in [Florian et al., 2000]. The experiments with base NP chunking show similar results.



(a) F-measure vs. number of words in training set



(b) Chunk Accuracy vs. number of words in training test

Figure 4: Performance of the TBLprob classifier versus sequential choice.

5 Conclusions

In this paper, we presented TBLprob, a novel method which enables a TBL classifier to produce probabilistic classifications. We show four experiments that demonstrate the quality and the effectiveness of the proposed method. In these experiments, the text chunking and the base noun phrase chunking tasks were used as test cases.

The experimental results indicate that our probabilistic classifier performs at least as well as TBLDT. Moreover, using smoothing techniques, we outperformed TBLDT in our four experiments.

References

[Blum and Mitchell, 1998] Blum, A. and Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *COLT*. Morgan Kaufmann.

[Brill, 1995] Brill, E. (1995). Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging. *Comput. Linguistics*, 21(4):543–565.

[Chen and Goodman, 1996] Chen, S. F. and Goodman, J. (1996). An empirical study of smoothing techniques for language modeling. In Joshi, A. and Palmer, M., editors, *Proceedings of the Thirty-Fourth Annual Meeting of the Association for Computational Linguistics*, pages 310–318, San Francisco. Morgan Kaufmann Publishers.

[2] The additive smoothing ($\delta = 0.05$) is used in this experiment.

[Collins, 2003] Collins, M. (2003). Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

[CoNLL, 2000] CoNLL (2000). Shared task for computational language learning (conll).

[Dietterich and Bakiri, 1995] Dietterich, T. G. and Bakiri, G. (1995). Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286.

[Florian et al., 2000] Florian, R., Henderson, J. C., and Ngai, G. (2000). Coaxing confidences from an old friend: Probabilistic classifications from transformation rule lists. In *Proceedings of Joint Sigdat Conference on Empirical Methods in NLP and Very Large Corpora*, Hong Kong University of Science and Technology.

[Freitas et al., 2006] Freitas, M. C., Duarte, J. C., Santos, C. N., Milidiú, R. L., Renteria, R. P., and Quental, V. (2006). A machine learning approach to the identification of appositives. In *Proceedings of Ibero-American AI Conference*, Ribeirï¿½ Preto, Brazil.

[Kalt, 2005] Kalt, T. (2005). *Control Models of Natural Language Parsing*. PhD thesis, University of Massachusetts Amherst.

[Milidiú et al., 2006] Milidiú, R. L., Santos, C. N., Duarte, J. C., and Renteria, R. P. (2006). Semi-supervised learning for portuguese noun phrase extraction. In *Proceedings of 7th Workshop on Computational Processing of Written and Spoken Portuguese*, pages 200–203, Itatiaia, Brazil.

[Ramshaw and Marcus, 1999] Ramshaw, L. and Marcus, M. (1999). Text chunking using transformation-based learning. In Armstrong, S., Church, K., Isabelle, P., Manzi, S., Tzoukermann, E., and Yarowsky, D., editors, *Natural Language Processing Using Very Large Corpora*. Kluwer.