A SPANISH POS TAGGER WITH VARIABLE MEMORY

Triviño-Rodriguez, J.L. **Morales-Bueno, R.**
Dept. Lenguajes y Ciencias de la Computación
University of Málaga
{trivino,morales}@lcc.uma.es

Abstract

An implementation of a Spanish POS tagger is described in this paper. This implementation combines three basic approaches: a single word tagger based on decision trees, a POS tagger based on variable memory Markov models, and a feature structures set of tags. Using decision trees for single word tagging allows the tagger to work without a lexicon that lists only possible tags. Moreover, it decreases the error rate because there are no unknown words. The feature structure set of tags is advantageous when the available training corpus is small and the tag set large, which can be the case with morphologically rich languages like Spanish. Finally, variable memory Markov models training is more efficient than traditional full-order Markov models and achieves better accuracy. In this implementation, 98.58% of tokens are correctly classified.

1 Introduction

Many words in Spanish function as different parts of speech (POS). Part-of-speech tagging is the problem of determining the syntactic part of speech of an occurrence of a word in context. Because most of the high-frequency Spanish words function as several parts of speech, an automatic system for POS tagging is very important for most other high level natural language text processing.

There are many approaches to the performance of this task, but they could be classified into two main ones:

- The linguistic approach, in which the language model is written by a linguist, generally in the form of rules or constraints. An example of this approach is the tagger developed by Voutilainen [Voutilainen and Järvinen, 1995].

- The automatic approach, in which the language model is automatically obtained from corpora. This corpora can be either raw or annotated. In this way, the automatic taggers are classified into supervised and unsupervised learning:

  - Supervised learning is applied when the model is obtained from annotated corpora.
  - Unsupervised learning is applied when the model is obtained from raw corpora training.

  The most noticeable examples of automatic tagging approaches are Markov chains [Church, 1989, Charniak et al., 1993, Jelinek, 1985, Merialdo, 1994, Garside et al., 1987, Cutting et al., 1992], neural networks [Schmid, 1994], decision trees [Daelemans et al., 1996, Márquez and Rodríguez, 1995], and transformation-based error driven learning [Brill, 1994].

The most widely-used methods for POS tagging are stochastic methods based on fixed order Markov chains models and Hidden Markov models. However, the complexity of Markov chains grows exponentially with its order L, and hence only low order Markov chains could be considered in practical applications. Moreover, it has been observed that, in many natural sequences, memory length depends on the context and is not fixed. In order to solve this problem, in 1996 Dana Ron [Ron, 1996, Ron et al., 1996] described a learning model of a subclass of PFAs (*Probabilistic Finite Automatas*) called PSAs (*Probabilistic Suffix Automatas*). A PSA is like a Markov chain with variable memory length. So, this model avoids the exponential complexity relation with the order L and thus can be applied to better effect than Markov chains.

In this paper a Spanish POS tagger based on variable memory Markov chains is described (which will be called *VMM* throughout). This tagger is based on a modified version of Singer's tagger [Schütze and Singer, 1994]. The VMM training algorithm used by Singer's tagger is more efficient than the Markov chains training algorithm. Furthermore, it allows higher order chains than fixed order Markov models. This implies better accuracy and a lower error rate.

On the other hand, the described tagger makes use of a feature structure set of tags like the model described by Kempe [Kempe, 1994]. Usually, the contextual probability of a tag is estimated as the relative frequency in a given training corpus. With a large tag set (resulting from the fact that the tags contain – besides the POS – a lot of morphological information) and with only a small training corpus available, most of these probabilities are too low for an exact estimation of contextual probabilities. Moreover, a large tag set increases the complexity of the Markov chains and it implies low training efficiency. An example of this problem is the tagger developed by Sánchez [Sánchez León and Nieto Serrano, 1995].

Finally, most taggers deal with the problem of determining the syntactic part of speech of an occurrence of a word in context, but they can not determine the set of correct tags for a word without any such context. Instead, most taggers use a lexicon that lists possible tags. In morphologically rich languages, this lexicon must be very large in order to represent the language correctly. In order to solve this problem, a single word tagger has been added to the system described. This single word tagger [Triviño, 1995] is based on a modified version of the ID3 algorithm described by Quinlan [Quinlan, 1986].

This paper is organised as follows: first the single word tagger is described. Next, the VMM is shown and how it is applied to POS tagging. Finally, the results obtained from this model are analysed.

2 Single word tagging

The single word tagger is the algorithm that computes all valid tags for a word if the word is considered out of context. The information used to carry out this task is the morphology of the word. Therefore, this kind of algorithm must be applied to morphologically rich languages where a word may have many derived words and where the lexicon must be very large in order to hold all these words.

Most single word taggers like Padro's tagger [Padró, 1996] are based on the two-level morphology described by Kimmo Koskenniemi [Koskenniemi, 1985]. Koskenniemi's model of two-level morphology was based on the traditional distinction that linguists make between morphotactics, which enumerates the inventory of morphemes and specifies in what order they can occur, and morphophonemics, which accounts for alternate forms or "spellings" of morphemes according to the phonological context in which they occur. For example, the word *chased* is analyzed morphotactically as the stem *chase*

followed by the suffix *-ed*. However, the addition of the suffix *-ed* apparently causes the loss of the final *e* of *chase*; thus *chase* and *chas* are allomorphs or alternate forms of the same morpheme. Koskenniemi's model is "two-level" in the sense that a word is represented as a direct letter-for-letter correspondence between its lexical or underlying form and its surface form.

Instead of this, the single word tagger that we have developed is based on a modified version of the ID3 algorithm. This algorithm generates a decision tree for a single out attribute (called *class* in TDIDT terminology). However, the feature structure set of tags of the tagger needs a decision tree for several out attributes. In order to solve this problem the ID3 algorithm has been modified. So, the modified version of the algorithm builds up the tree recursively as follows: first, the algorithm chooses the best out attribute in order to generate the tree; next, the tree is built up recursively following the traditional ID3 algorithm until the selected out attribute is classified; finally, the algorithm goes back and chooses another out attribute for each leaf to build up the tree from that leaf.

The single word tagger has been trained with an 87830 words corpus. From this corpus, the learning algorithm has computed a decision tree with 48317 rules. Next, an example of a generated rule is shown (the symbol '*' means any symbol):

```
**********************************b***dad
  [NomN_ComFemSg#1]
```

A word matches in this rule if the word ends with the symbol 'b' followed by any three symbols followed by the string 'dad'. This rule has been computed from the words: *accesibilidad, aceptabilidad, adaptabilidad, admisibilidad, afabilidad, agregabilidad, amabilidad, amigabilidad, antiobesidad, aplicabilidad, asociabilidad, autoestabilidad, barbaridad, combustibilidad, comensurabilidad, compatibilidad, comunicabilidad, conductibilidad, confortabilidad, conmutabilidad, contabilidad, contrastabilidad, convertibilidad, corresponsabilidad, corruptibilidad, credibilidad, cuestionabilidad, culpabilidad, debilidad, deseabilidad, disponibilidad, elegibilidad, estabilidad, excitabilidad, factibilidad, falibilidad, fiabilidad, flexibilidad, flotabilidad, globalidad, gobernabilidad, habilidad, habitabilidad, heredabilidad, honorabilidad, impasibilidad, impenetrabilidad, impermeabilidad, imposibilidad, imprevisibilidad, improbabilidad, imputabilidad, inaccesibilidad, incompatibilidad, incomunicabilidad, incorruptibilidad, inestabilidad, infalibilidad, ingobernabilidad, ininteligibilidad, insaciabilidad, insensibilidad, inteligibilidad, inviabilidad, invisibilidad, irresponsabilidad, irreversibilidad, irritabilidad, maniobrabilidad, morbilidad, morbosidad, mutabilidad, nubosidad, obesidad, obviedad, perdurabilidad, permeabilidad, posibilidad, probabilidad, rentabilidad, respetabilidad, responsabilidad, reversibilidad, sensibilidad, sobriedad, sociabilidad, solubilidad, subunidad, susceptibilidad, urbanidad, verbosidad, viabilidad, volubilidad, vulnerabilidad.*

3 Training algorithm

3.1 Variable memory Markov models (VMM)

In this section, the variable memory Markov model is described. This model is called PSA (*Probabilistic Suffix Automata*) and has been described by Ron [Ron, 1996]. This model is hard to learn. Another model exists: the PST model (*Prediction Suffix Trees*) that can be learned more easily. It can be shown that every distribution generated by a PSA can be equivalently generated by a PST which is not much larger. So, the PST model is used as learning hypothesis instead of the PSA model.

Definition 3.1 (PST (*Prediction Suffix Tree*)) *A PST T over an alphabet Σ is a tree of degree $|\Sigma|$. Each edge in the tree is labeled by a single symbol in Σ such that from every internal node there*

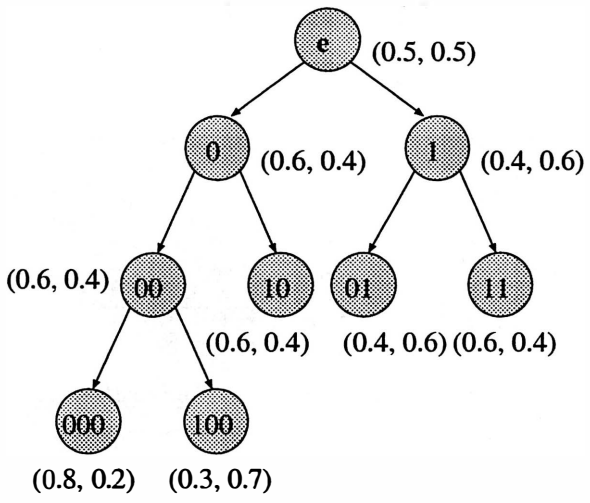

Figure 1: Example of PST over the alphabet $\{0, 1\}$. The prediction probabilities of the symbols '0' and '1' are next to the nodes, in parentheses, respectively.

is exactly $|\Sigma|$ edges each one labeled with a different symbol. The nodes of the tree are labeled by pairs (s, γ_s) where s is the string associated with the walk starting from that node and ending in the root of the tree, and $\gamma_s : \Sigma \rightarrow [0, 1]$ is the next symbol probability function related with s. It is necessary that for every string s labeling a node in the tree, $\sum_{\sigma \in \Sigma} \gamma_s(\sigma) = 1$. An example of PST can be seen in figure 1.

An interesting feature of the PST model is that the transition probabilities are smoothed along the learning task. This avoids transition probabilities with value 0. For this reason, the model must not be interpolated after the learning task unlike the Markov model.

The PST learning algorithm generates a PST hypothesis that holds the ϵ-good hypothesis property with respect to a given PSA.

Definition 3.2 (ϵ-good hypothesis) *Let M be a PSA and let T be a PST and let P_M and P_T be the two probability distributions they generate respectively. We say that T is an ϵ-good hypothesis with respect to M if $\forall N > 0$, it is held that:*

$$\frac{1}{N} * \underset{KL}{D}[P_M^N || P_T^N] \leq \epsilon$$

Where

$$\underset{KL}{D}[P_M^N || P_T^N] \stackrel{def}{=} \sum_{r \in \Sigma^N} \underset{M}{P}(r) * \log \frac{P_M^N(r)}{P_T^N(r)}$$

Definition 3.3 (Parameters of the learning model)

- *L is the maximum length of the string labeling the states.*
- *n is an upper bound on the number of states in M.*
- *δ is the confidence parameter, $0 < \delta < 1$.*
- *ϵ is the approximation parameter, $0 < \epsilon < 1$*
- *The training sample is compounded by m' strings generated by M each of length $L + 1$ at least.*

Definition 3.4 (Observation probabilities) *The probability of a state s and a symbol σ in the learning sample is calculated as the relative frequency of the state s (number of times the state s*

appears in the sample divided by the length of the sample) and the relative frequency of the symbol σ in the context of the state s (number of times the symbol σ appears in the sample after the state s divided by the number of times the state s appears in the sample). This is computed by the following equations:

$$\widetilde{\Pr}(s) = \frac{1}{m'(\ell - L)} \sum_{i=1}^{m'} \sum_{j=L}^{\ell-1} \chi_j^i(s)$$

$$\widetilde{\Pr}(\sigma|s) = \frac{\sum_{i=1}^{m'} \sum_{j=L}^{\ell-1} \chi_{j+1}^i(s\sigma)}{\sum_{i=1}^{m'} \sum_{j=L}^{\ell-1} \chi_j^i(s)}$$

Where m' is the number of strings in the sample and ℓ is the length of the strings, such that $\ell \geq L+1$ and $\chi_j^i(s)$ is defined as 1 if $r_{j-|s|+1}^i \ldots r_j^i = s$ and 0 otherwise.

Definition 3.5 (Internal parameters) *The following parameters are used internally by the learning algorithm:*

$$\epsilon_2 = \frac{\epsilon}{48 * L}$$

$$\gamma_{min} = \frac{\epsilon_2}{|\Sigma|}$$

$$\epsilon_0 = \frac{\epsilon}{2 * n * L * \log(\frac{1}{\gamma_{min}})}$$

$$\epsilon_1 = \frac{\epsilon_2 * \gamma_{min}}{8 * n * \epsilon_0}$$

The learning algorithm generates a PST from a training sample following a top-down approach. It starts with a tree consisting of a single root node labeled with the empty string. Progressively, new nodes and edges are added to the tree. Each node is added depending on its observation probability. In this way, a new node is added when it has a significant probability of being in the sample and the transition probabilities from this node differ substantially from the transition probability of its parent. An example of the learning task can be seen in figure 2.

3.2 Tagging with VMM (Variable Memory Markov Models)

Part-of-speech tagging is the problem of determining the syntactic part of speech of an occurrence of a word in context. Therefore, given a sequence of words $w_1, \ldots, w_n$, POS tagging is the problem of determining the tag sequence $t_1, \ldots, t_n$ that is most likely for $w_1, \ldots, w_n$. This could be computed by maximizing the joint probability of $w_1, \ldots, w_n$ and $t_1, \ldots, t_n$:

$$\begin{aligned} \tau(w_1, \ldots, w_n) &= \underset{t_1,\ldots,t_n}{argmax} Pr(t_1, \ldots, t_n | w_1, \ldots, w_n) \\ &= \underset{t_1,\ldots,t_n}{argmax} \frac{Pr(t_1, \ldots, t_n, w_1, \ldots, w_n)}{Pr(w_1, \ldots, w_n)} \\ &= \underset{t_1,\ldots,t_n}{argmax} Pr(t_1, \ldots, t_n, w_1, \ldots, w_n) \end{aligned} \quad (1)$$

The equation 1 could be laid out as follows:

$$Pr(t_1, \ldots, t_n, w_1, \ldots, w_n) = \prod_{i=1}^{n} Pr(t_i | t_1, \ldots, t_{i-1}) * Pr(w_i | t_1, \ldots, t_i, w_1, \ldots, w_{i-1}) \quad (2)$$

Algorithm 3.1 (Learning a PST)

1. *Let $\bar{T}$ be a single root node (labeled with* **e***) and let*
   $\bar{S} = \{\sigma \mid \sigma \in \Sigma \wedge \widetilde{\Pr}(\sigma) \geq (1-\epsilon_1) * \epsilon_0\}$.
2. *While $\bar{S} \neq \emptyset$, pick up any $s \in \bar{S}$ and do:*
   (a) *Remove s from $\bar{S}$*
   (b) *If there exists $\sigma \in \Sigma$ such that*

$$\widetilde{\Pr}(\sigma|s) \geq (1+\epsilon_2) * \gamma_{min} \quad \wedge \quad \frac{\widetilde{\Pr}(\sigma|s)}{\widetilde{\Pr}(\sigma|\, suffix(s))} > 1 + 3 * \epsilon_2$$

   then add to $\bar{T}$ the node s and all the nodes on the path from the deepest node in $\bar{T}$ that is a suffix of s, to $\bar{S}$.
   (c) *If $|s| < L$ then, for every $\sigma' \in \Sigma$, if $\widetilde{\Pr}(\sigma' \cdot s) \geq (1-\epsilon_1) * \epsilon_0$ then add $\sigma' \cdot s$ to $\bar{S}$.*
3. *Let $\hat{T} = \bar{T}$*
4. *Extend $\hat{T}$ by adding all missing sons of internal nodes.*
5. *For each s labeling a node in $\hat{T}$ let:*

$$\hat{\gamma}_s(\sigma) = \widetilde{\Pr}(\sigma|s') * (1 - |\Sigma| * \gamma_{min}) + \gamma_{min}$$

   where s' is the longest suffix of s in $\bar{T}$

Up to this point no assumptions about the probabilities have been made, but the probabilities required by equation 2 are not empirically collectible. In order to solve this problem, it is necessary to make the following assumptions about these probabilities [1]:

$$Pr(t_i|t_1,\ldots,t_{i-1}) = Pr(t_i|t_{i-L},\ldots,t_{i-1}) \quad (3)$$

$$Pr(w_i|t_1,\ldots,t_i,w_1,\ldots,w_{i-1}) = Pr(w_i|t_i) \quad (4)$$

By equations 3 and 4, equation 2 can be expressed as follows:

$$Pr(t_1,\ldots,t_n,w_1,\ldots,w_n) = \prod_{i=1}^{n} Pr(t_i|t_{i-L},\ldots,t_{i-1}) * Pr(w_i|t_i) \quad (5)$$

Where L is the maximum length of the model.

This tagger is based on a feature structure set of tags. Therefore, the term t_i in equation 5 is a vector of a symbols[2] (a symbol for each attribute). However, the Viterbi algorithm [Viterbi, 1967] can only compute the probability of a sequence of a single attribute. In order to solve this problem, a set of a trees (a tree for each attribute) has been computed independently. The equation 5 is computed from this set of trees as follows:

$$Pr(t_1,\ldots,t_n,w_1,\ldots,w_n) =$$

[1] These assumptions were described by Charniak [Charniak et al., 1993]. That is, that the probability of the tag given the past depends only on the last two tags, and the probability of the word given the past depends only on its tag.

[2] We take into account 18 attributes for each word, thus $a = 18$ in our tagger

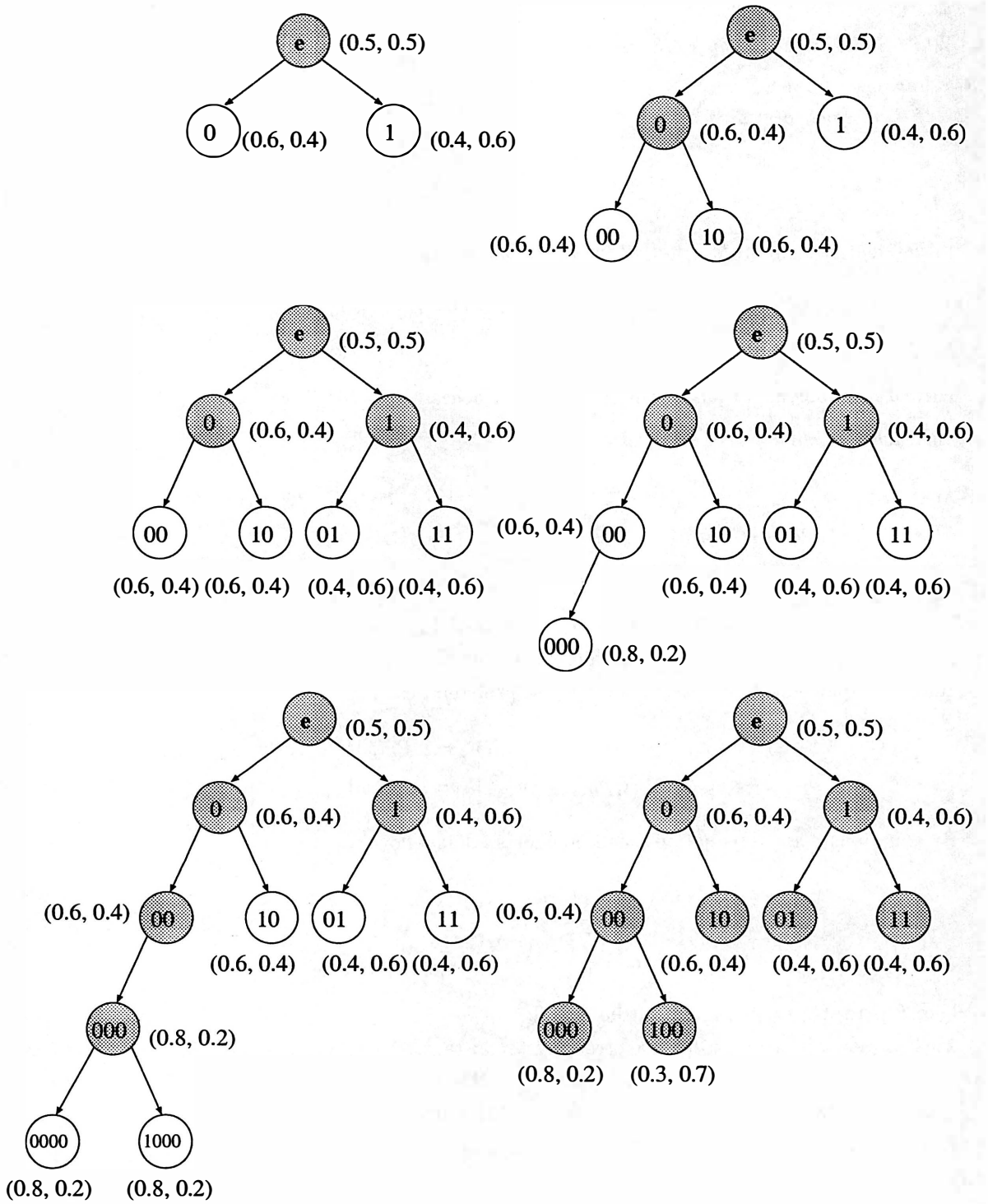

Figure 2: An illustrative run of the learning algorithm

$$= \prod_{j=1}^{a} Pr(t_1^j, \ldots, t_n^j, w_1, \ldots, w_n)$$
$$= \prod_{i=1}^{n} (Pr(w_i|t_i) * \prod_{j=1}^{a} Pr(t_i^j | t_{i-L}^j, \ldots, t_{i-1}^j)) \quad (6)$$

Given a set of PSTs T, $Pr(t_i^j|t_{i-L}^j, \ldots, t_{i-1}^j)$ is estimated by $\gamma_s^j(t_i^j)$ where s is the longest suffix of $t_{i-L}^j, \ldots, t_{i-1}^j$ labeling a state in T^j and γ_s^j is the next symbol probability function related with the state s in T^j.

The static probability $Pr(w_i|t_i)$ is estimated indirectly from $Pr(t_i|w_i)$ using Bayes' Theorem:

$$Pr(w_i|t_i) = \frac{Pr(w_i) * Pr(t_i|w_i)}{Pr(t_i)}$$

The terms $Pr(w_i)$ are constant for a given sequence w_i and can therefore be omitted from the maximization. The values of $Pr(t_i)$ can be computed as follows:

$$Pr(t_i) = \prod_{j=1}^{a} \gamma_{\mathbf{e}}^j(t_i^j) \quad (7)$$

The static parameters $Pr(t_i|w_i)$ are computed by the single word tagging.

4 Analysis of results

The order L of the VMM tagger has been set to 4. The tagger has been trained on a Spanish 45000 words corpus tagged with the set of tags shown in table 1. An example of tagged text [3] can be seen in figure 3.

The accuracy obtained is 98.58%. This accuracy is better than that achieved by any other tagger we know of. In table 2 a comparison between the accuracy of several taggers is shown.

Our results (98.58%) are better than Singer's tagger (95.81%) based on VMM because the single word tagger has added lexical information to our tagger. This can decrease the error rate when errors due to bad tags for rare words are avoided by the single word tagger. However, it is difficult to compare these results to other works, since the accuracy varies greatly depending on the corpus, tag set, etc. More exact performance could be measured by increasing the size of training a testing text. Thus, the performance could decrease or increase if different corpus styles are used.

5 Conclusions and future work

In this paper, a high accuracy Spanish tagger has been presented. This tagger has three main features: a single word tagger based on decision trees, a feature structure set of tags, and a variable memory Markov model.

The results obtained show that joining an accurate single word tagger with a feature structure set of tags and a high order VMM produces an improvement in the performance of the tagger.

[3] Many spanish verbs have the same form to several times. In example, the verb *'tomar'* have the same word (*'toma'*) for the third person in present time and second person in imperative time. The rest of persons in these times are different. In order to determine the valid tag, it must be necessary a syntactic analysis. Therefore, the POS tagger keeps two different tags to this word. In order to measure the accuracy of the tagger, this word is considered to be correctly tagged only if it has its two tags.

Word	Valid tags	Selected tag
Él	[ProP_PerMasP3Sg#1]	[ProP_PerMasP3Sg#1]
toma	[NomN_ComFemSg#0.333333] [VerImvImpSim2Sg#0.333333] [VerIndPrs3Sg#0.333333]	[VerImvImpSim2Sg#0.5] [VerIndPrs3Sg#0.5]
el	[ArtAr_DetMasSg#1]	[ArtAr_DetMasSg#1]
tomo	[VerIndPrs1Sg#0.5] [NomN_ComMasSg#0.5]	[NomN_ComMasSg#1]
del	[Pre#1]	[Pre#1]
libro	[VerIndPrs1Sg#0.5] [NomN_ComMasSg#0.5]	[NomN_ComMasSg#1]
bajo	[Pre#0.2] [AdvAv_Modo#0.2] [NomN_ComMasSg#0.2] [AdjAd_CalMasSg#0.2] [VerActIndPrs1Sg#0.2]	[Pre#1]
la	[ArtAr_DetFemSg#0.333333] [ProP_PerFemP3Sg#0.333333] [NomN_ComMasSg#0.333333]	[ArtAr_DetFemSg#1]
mesa	[NomN_ComFemSg#0.333333] [VerImvImpSim2Sg#0.333333] [VerIndPrs3Sg#0.333333]	[NomN_ComFemSg#1]
del	[Pre#1]	[Pre#1]
hombre	[NomN_ComMasSg#1]	[NomN_ComMasSg#1]
bajo	[Pre#0.2] [AdvAv_Modo#0.2] [NomN_ComMasSg#0.2] [AdjAd_CalMasSg#0.2] [VerActIndPrs1Sg#0.2]	[AdjAd_CalMasSg#1]
.	[SimbSPunto]	[SimbSPunto]

Figure 3: An example of the output of the tagger

Attribute	# Tags
Class	12
Class of noun	4
Class of adjective	13
Class of adverb	9
Class of pronoun	13
Function of pronoun	4
Class of conjunction	13
Class of Article	3
Gender	5
Active/Passive	3
Form	6
Tense	19
Person	4
Number	4
Prefix	2
Suffix	2
Class of Symbol	22

Table 1: Set of tags in the corpus

Tagger	**Language**	**Corpus (# words)**	**Accuracy (%)**
Triviño	Spanish	45000	98.58
Padró	Spanish/english	10^6	97.45
Charniak	English	10^6	96.45
Kempe	French	$2 * 10^6$	96.16
Brill	English	350000	96.0
Singer	English	10^6	95.81
Kempe	French	10000	88.89

Table 2: Comparison between several taggers

On the other hand, most of the errors arise because the tagger does not take into account relations between out attributes. Thus, the main development line of this work is to change the variable memory model for a model that takes relations between attributes into account.

References

[Brill, 1994] Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *Proceedings of AAAI94*, page 6.

[Charniak et al., 1993] Charniak, E., Hendrickson, C., Jacobson, N., and Perkowitz, M. (1993). Equations for part-ofspeech tagging. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, pages 784–789.

[Church, 1989] Church, K. (1989). A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of ICASSP*.

[Cutting et al., 1992] Cutting, D., Kupiec, J., Pederson, J., and Sibun, P. (1992). A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing.* ACL.

[Daelemans et al., 1996] Daelemans, W., Zavrel, J., Beck, P., and S.Gillis (1996). Mtb: A memory-based part-of-speech tagger generator. In *Proceedings of 4th Workshop on Very Large Corpora*, Copenhagen, Denmark.

[Garside et al., 1987] Garside, R., Leech, G., and Sampson, G. (1987). *The Computational Analysis of English.* London and New York: Longman.

[Jelinek, 1985] Jelinek, F. (1985). Robust part-of-speech tagging using a hidden markov model. Technical report, IBM.

[Kempe, 1994] Kempe, A. (1994). Probabilistic tagging with feature structures. In *Coling-94*, volume 1, pages 161–165.

[Koskenniemi, 1985] Koskenniemi, K. (1985). A general two-level computational model for word-form recognition and production. In Departament of Linguistics, editor, *Karlsson, F.*, pages 1–18. Computational Morphosyntax, University of Helsinki.

[Merialdo, 1994] Merialdo, B. (1994). Tagging english text with a probabilistic model. *Computational Linguistics*, 2(20):155–171.

[Márquez and Rodríguez, 1995] Márquez, L. and Rodríguez, H. (1995). Towards learning a constraint grammar from annotated corpora using decision trees. ESPRIT BRA-7315 Acquilez II, Working Paper.

[Padró, 1996] Padró, L. (1996). Pos tagging using relaxation labelling. In *Proceedings of 16th International Conference on Computational Linguistics*, Copenhagen, Denmark.

[Quinlan, 1986] Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, (1):81–106.

[Ron, 1996] Ron, D. (1996). *Automata Learning and its Applications.* PhD thesis, MIT.

[Ron et al., 1996] Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia: Learning probabilistic automata with variable memory length. *Machine Learning*, page 34.

[Schmid, 1994] Schmid, H. (1994). Part-of-speech tagging with neural networks. In *Proceedings of 15th International Conference on Computational Linguistics*, Kyoto, Japan.

[Schütze and Singer, 1994] Schütze, H. and Singer, Y. (1994). Part-of-speech tagging using a variable memory Markov model. In *ACL 32'nd.*

[Sánchez León and Nieto Serrano, 1995] Sánchez León, F. and Nieto Serrano, A. (1995). Development of a spanish version of the Xerox Tagger. Technical report, Facultad de Filosofía y Letras (Universidad Autónoma de Madrid).

[Triviño, 1995] Triviño, J. (1995). SEAM. Sistema experto para análisis morfológico. Master's thesis, Universidad de Málaga.

[Viterbi, 1967] Viterbi, A. (1967). error bounds for convolutional codes and an asymptotical optimal decoding algorithm. In *Proceedings of IEEE*, volume 61, pages 268–278.

[Voutilainen and Järvinen, 1995] Voutilainen, A. and Järvinen, T. (1995). Specifying a shallow grammatical representation for parsing purposes. In *Proceedings of the 7th meeting of the European Association for Computational Linguistics*, pages 210–214.