Developing Competitive HMM PoS Taggers Using Small Training Corpora *

Muntsa Padró and Lluís Padró
TALP Research Center
Universitat Politècnica de Catalunya
{mpadro,padro}@lsi.upc.es

February 2004

Abstract

This paper presents a study aiming to find out the best strategy to develop a fast and accurate HMM tagger when only a limited amount of training material is available. This is a crucial factor when dealing with languages for which small annotated material is not easily available.

First, we develop some experiments in English, using WSJ corpus as a test-bench to establish the differences caused by the use of large or a small train set. Then, we port the results to develop an accurate Spanish PoS tagger using a limited amount of training data.

Different configurations of a HMM tagger are studied. Namely, trigram and 4-gram models are tested, as well as different smoothing techniques. The performance of each configuration depending on the size of the training corpus is tested in order to determine the most appropriate setting to develop HMM PoS taggers for languages with reduced amount of corpus available.

1 Introduction

PoS Tagging is a need for most of Natural Language applications such as Sumarization, Machine Translation, Dialogue systems, etc. and the basis of many higher level NLP processing tasks. It is also used to obtain annotated corpora combining automatic tagging with human supervision. These corpora may be used for linguistic research, to build better taggers, or as statistical evidence for many other language-processing related goals.

*This research has been partially supported by the European Comission (Meaning, IST-2001-34460) and by the Catalan Government Research Department (DURSI).

PoS tagging has been largely studied and many systems developed. There are some statistical implementations [1, 2, 3, 4, 5, 6] and some knowledge-based taggers (finite-state, rule-based, memory based) [7, 8, 9]. There are also some systems that combine different implementations with a voting procedure.

This work presents a thorough study aiming to establish which is the most appropiate way to train a HMM PoS tagger when dealing with languages with a limited amount of training corpora. To do so, we compare different smoothing techniques and different order HMMs.

Experiments are performed to determine the performance of the best configuration when the tagger is trained with a large English corpus (1 million words from WSJ), and comparing the results with those for a training corpus ten times smaller. Then, the experiments for the small train corpus are repeated in another language (Spanish), validating the conclusions. The tested HMM configurations vary on the order of the model (3 and 4 order HMMs are tested) and in the smoothing technique (Lidstone's law *vs.* Linear Interpolation) used to estimate model parameters.

Section 2 presents the theoretical basis of a HMM and the different smoothing techniques used in this work. Section 3 shows the realized experiments and the obtained results. Section 4 states some conclusions and further work.

2 Hidden Markov Models

We will be using Hidden Markov Models Part-of-Speech taggers of order three and four. Depending on the order of the model, the states represent pairs or triples of tags, and obviously, the number of parameters to estimate varies largely. As the states are pairs or triples of tags, the possible number of states are the possible combinations of all the tags in groups of two or three. Table 1 shows the number of tags for each language and the consequent number of potential states. This number is very large but there are many states that will never be observed. The emmited symbols are words, which we estimated to be about 100,000 for English and 1,000,000 for Spanish.

	English	Spanish
Number of tags	47	67
Number of potential states in a 3-gram HMM	2,209	4,489
Number of potential states in a 4-gram HMM	103,823	300,763

Table 1: Number of potential tags and states for English and Spanish with both models

The parameters of such models are initial state probabilities, state transition

probabilities and emission probabilities. That is:

$$\pi_i = P(q_1 = s_i)$$

is the probability that a sequence starts at state s_i,

$$a_{ij} = P(q_{t+1} = s_j | q_t = s_i)$$

is the transition probability from state i to state j (i.e. trigram probability $P(t_3|t_1t_2)$ in a 3^{rd} order model, or 4-gram probability $P(t_4|t_1t_2t_3)$ in a 4-gram HMM), and

$$b_i(k) = P(w_k|s_i)$$

is the emission probability of the symbol w_k from state s_i.

In the PoS task model, the emitted symbols are the observed words and the observed sequence is a complete sentence. Given a sentence, we want to choose the most likely sequence of states (i.e. PoS tags) that generated it. This is computed using the Viterbi algorithm [10].

2.1 Parameter Estimation

The simplest way to estimate the HMM parameters is Maximum Likelihood Estimation (MLE), which consists in computing observation relative frequencies:

$$P_{MLE}(x) = \frac{count(x)}{N}$$

$$P_{MLE}(x|y) = \frac{count(x,y)}{count(x)}$$

For the case of the HMM we have to compute this probability estimation for each initial state, transition or emission in the training data:

$$\pi_i = \frac{count(s_i(t=0))}{count(sentences)}$$

$$a_{ij} = \frac{count(s_i \rightarrow s_j)}{count(s_i)}$$

$$b_i(k) = \frac{count(s_i, w_k)}{count(s_i)}$$

where $count(s_i(t=0))$ is the number of times that s_i is visited as an initial state, $count(sentences)$ is the number of sentences, $count(s_i \rightarrow s_j)$ is the count of

the transitions from state s_i to s_j, $count(s_i)$ is the number of visits to state s_i and $count(s_i, w_k)$ is the number of times that symbol w_k is emmited from state s_i.

Actually, computing $b_i(k)$ in this way is quite difficult because the number of occurrences of a single word will be too small to provide enough statistical evidence, so Bayes rule is used to compute $b_i(k)$ as:

$$b_i(k) = P(w_k|s_i) = \frac{P(s_i|w_k)P(w_k)}{P(s_i)}$$

where:

$$P(s_i) = \frac{count(s_i)}{count(words)} \quad ; \qquad P(w_k) = \frac{count(w_k)}{count(words)}$$

being $count(words)$ the number of words in the training corpus.

Since $P(s_i|w_k)$ would also require lots of data to be properly estimated, we approximate it as $P(t|w_k)$, where t is the last tag in the n-gram corresponding to the state. Similarly, $P(s_i)$ is approximated as $P(t)$.

2.2 Smoothing

MLE is usually a bad estimator for NLP purposes, since data tends to be sparse[1]. This leads to zero probabilities being assigned to unseen events, causing troubles when multiplying probabilities.

To solve this sparseness problem it is necessary to look for estimators that assign a part of the probability mass to the unseen events. To do so, there are many different smoothing techniques, all of them consisting of decreasing the probability assigned to the seen events and distributing the remaining mass among the unseen events. In this work two smoothing methods are compared: Lidstone's law and Linear Interpolation.

2.2.1 Laplace and Lidstone's Law

The oldest smoothing technique is Laplace's law [11], that consists in adding one to all the observations. That means that all the unseen events will have their probability computed as if they had appeared once in the training data. Since one observation for each event (seen or unseen) is added, the number of different possible observations (B) has to be added to the number of real observations (N), in order to maintain the probability normalised.

$$P_{La}(x) = \frac{count(x+1)}{N+B}$$

[1] Following Zipf's law: a word's frequency is inversely proportional to its rank order

However, if the space is large and very sparse –and thus the number of possible events (B) may be similar to (or even larger than) the number of observed events– Laplace's law gives them too much probability mass.

A possible alternative is Lidstone's law (see [12] for a detailed explanation on these and other smoothing techniques) which generalises Laplace's and allows to add an arbitrary value to unseen events. So, for a relatively large number of unseen events, we can choose to add values lower than 1. For a relatively small number of unseen events, we may choose 1, or even larger values, if we have a large number of observations (N).

$$P_{Ld}(x) = \frac{count(x) + \lambda}{N + B\lambda} \qquad \lambda > 0$$

To use Laplace's or Lidstone's laws in a HMM-based tagger we have to smooth all probabilities involved in the model:

$$\pi_i = \frac{count(s_i(t=0)) + \lambda_\pi}{count(sentences) + B_{tag}\lambda_\pi}$$

$$a_{ij} = \frac{count(s_i \rightarrow s_j) + \lambda_A}{count(s_i) + B_{tag}\lambda_A}$$

$$P(s_i) = \frac{count(s_i) + \lambda_s}{count(words) + B_{tag}\lambda_s}$$

$$P(w_k) = \frac{count(w_k) + \lambda_w}{count(words) + B_w\lambda_w}$$

where B_{tag} is the number of possible tags and B_w is the number of words in the vocabulary (obviously, we can only approximate this quantity).

Since there are different counts involved in each probability, we have to consider different λ values for each formula. In the case of Laplace's law, all λ are set to 1, but when using Lidstone's, we want to determine which is the best set of λ, and how they vary depending on the train set size, as discussed in section 3.1.

2.2.2 Linear Interpolation

A more sophisticated smoothing technique consists of linearly combine the estimations for different order n-grams:

$$\begin{aligned} P_{li}(t_3|t_1t_2) &= c_1P(t_3) + c_2P(t_3|t_2) + c_3P(t_3|t_1t_2) \\ P_{li}(t_4|t_1t_2t_3) &= c_1P(t_4) + c_2P(t_4|t_3) + c_3P(t_4|t_2t_3) + c_4P(t_4|t_1t_2t_3) \end{aligned}$$

where $\sum_i c_i = 1$ to normalise the probability. Although the values for c_i can be determined in many different ways, in this work they are estimated by deleted interpolation as described in [6]. This technique assumes that the c_i values don't depend on the particular n-gram and computes the weights depending on the counts for each i-gram involved in the interpolation.

3 Experiments and Results

The main purpose of this work is to study the behaviour of different configurations for a HMM-based PoS tagger, in order to determine the best choice to develop taggers for languages with scarce annotated corpora available.

First, we will explore different configurations when a large amount of training corpus is available. The experiments will be performed on English, using 1.1 million words from the Wall Street Journal corpus. Then, the same configurations will be explored when the training set is reduced to 100,000 words.

Later, the behaviour on the reduced train set will be validated on a manually developed 100,000 word corpus for Spanish [13].

The tested configurations vary on the order of the used HMM (trigram or 4-gram), and on the smoothing applied (Lidstone's law or Linear Interpolation).

All the experiments are done using ten fold cross-validation. In each fold, 90% of the corpus is used to train the tagger and the rest to test it.

In the following sections, we present the behaviour of the different HMM configurations for English, both with a large and a small corpus. After, we repeat the tests using a small corpus for Spanish.

3.1 Applying Lidstone's Law

As it has been explained in section 2.2.1, when Lidstone's law is used in a HMM tagger, there are four λ values to consider. Changing these values significantly affects the precision of the system.

Thus, before comparing this smoothing technique with another, we have to select the set of λ that yields the best tagger performance. After performing some experiments, we observed that λ_A is the only parameter that significantly affects the behaviour of the system. Modifying the other three values didn't change the system precision in a significant way. So λ_π, λ_s, and λ_w were set to some values determined as follows:

- λ_π is the assigned count for unobserved initial states. Since initial states depend only on the tag of the first word in the sentence, and the tag set we are using is quite reduced (about 40 tags), we may consider that in a

1,000,000 word corpus, at least one sentence will start with each tag. So, we will count one occurrence for unseen events (i.e. we are using $\lambda_\pi = 1$, Laplace's law, for this case). When the corpus is ten times smaller, we will use a proportional rate of occurrence ($\lambda_\pi = 0.1$).

- λ_s is the count assigned to the unseen states. Since we approximate $P(s_i)$ by $P(t)$ (see section 2.1), the possible events are the number of tags in the tag set, and we can reason as above, assuming at least one occurrence of each tag in a 1,000,000 word corpus (again, Laplace's law, $\lambda_s = 1$), and a proportional value for the small corpus ($\lambda_s = 0.1$)
- λ_w is the count assigned to the unseen words. Obviously, enforcing that each possible word will appear at least once would take too many probability mass out of seen events (English vocabulary is about 100,000 forms, which would represent 10% of a 1 million word corpus), so we adopt a more conservative value: $\lambda_w = 0.1$ for the large corpus, and the proportional value $\lambda_w = 0.01$ for the small one.

After setting these three λ values, we have to select the best value for λ_A. To diminish the risk of getting local maxima, we will repeatedly use hill-climbing with different starting values and step lengths ($\Delta\lambda$), and choose the value that produces better results.

In table 2 the results of this hill-climbing algorithm using the whole English corpus (1 million of words) are presented. Table 3 show the same results for the 100.000 words English corpus.

		Trigram HMM			4-gram HMM		
Initial λ_A	$\Delta\lambda$	Initial precision	Final precision	λ_A	Initial precision	Final precision	λ_A
0.05	0.01	96.98	97.00	0.22	96.72	96.88	0.28
0.05	0.005	96.98	96.99	0.085	96.72	96.81	0.125
0.5	0.1	97.008	97.009	0.6	96.91	96.93	1.0
0.5	0.05	97.008	97.009	0.4	96.91	96.93	0.95
1.0	0.5	97.00	**97.01**	0.5	96.93	**96.94**	1.5
1.0	0.1	97.00	**97.01**	0.6	96.93	96.93	1.0

Table 2: Precision obtained applying hill-climbing on the complete English corpus

As it was expected, when using a small corpus the precision falls, specially when a 4-gram HMM is used, since the evidence to estimate the model is insufficient. This point is discussed in section 3.3.

These results show that the value selected for λ_A is an important factor when

		Trigram HMM			4-gram HMM		
Initial λ_A	$\Delta\lambda$	Initial precision	Final precision	λ_A	Initial precision	Final precision	λ_A
0.05	0.01	96.56	96.63	0.09	95.79	96.30	0.33
0.05	0.005	96.56	96.63	0.1	95.79	96.20	0.2
0.5	0.1	96.69	96.69	0.5	96.36	96.43	0.8
0.5	0.05	96.69	96.69	0.5	96.36	96.43	0.75
1.0	0.5	96.70	96.70	1.0	96.44	**96.51**	3.5
1.0	0.1	96.70	**96.71**	0.9	96.44	96.46	1.2

Table 3: Precision obtained applying hill-climbing on the reduced English corpus

using this smoothing technique. As can be seen in table 3, the precision of the tagger varies up 0.7% depending on the value used for λ_A.

After performing the hill climbing search, we choose the λ_A that gives better results in each case, as the optimal parameter to use with this smoothing technique. So, for the whole corpus using a Trigram HMM, λ_A is set to 0.6 and the tagger yields a precision of 97.01%, while if we use a 4-gram HMM, $\lambda_A = 1.5$ leads to a precision of 96.94%. When the experiments are performed over the reduced corpus, the best results are obtained with $\lambda_A = 0.9$ for a trigram HMM (96.71%) and with $\lambda_A = 3.5$ for a 4-gram model (96.51%).

3.2 Applying Linear Interpolation

The performance of the taggers when using Linear Interpolation to smooth the probability estimations has been also tested. In this case, the coefficients c_i are found via the deleted interpolation algorithm (see section 2.2.2).

When using Linear Interpolation, the precision obtained by the system with the whole corpus is 97.00% with a trigram HMM, and 97.02% with a 4-gram HMM. For the reduced corpus the precision falls slightly and we obtain 96.84% for the trigram model and 96.71% for the 4-gram HMM.

The results obtained using Linear Interpolation and a trigram model should reproduce those reported by [6], where the maximum precision reached by the system on WSJ is 96.7%. In our case we obtain a higher precision because we are assuming the nonexistence of unknown words (i.e. the dictionary contains all possible tags for all words appearing in the test set. Obviously, word-tag frequency information from the test corpus is not used when computing $P(s_i|w_k)$).

3.3 Best Configuration for English

Best results obtained for each HMM tagger configuration are summarized in table 4. Results are given both for the large and small corpus.

1,1 Mword English corpus

	Lidstone's law	Linear Interpolation
trigram	97.01%	97.00%
4-gram	96.94%	**97.02%**

100 Kword English corpus

	Lidstone's law	Linear Interpolation
trigram	96.71%	**96.84%**
4-gram	95.51%	96.71%

Table 4: Obtained results for all HMM PoS tagger configurations using large and small sections of WSJ corpus

Comparing the results for the two smoothing methods used with different order models, we can draw the following conclusions:

- In general, Linear Interpolation produces taggers with higher precision than using Lidstone's law.
- For the case of the large corpus, the results are not significantly different for any combination of n-gram order and smoothing technique. While for the reduced corpus it is clearly better to use a trigram model than a 4-gram HMM, and Linear Interpolation yields slightly better results.
- Using Linear Interpolation has the benefit that the involved coefficients are computed using the training data via deleted interpolation, while for Lidstone's law the precision is very dependent on the λ_A value, which has to be costly optimised (e.g. via hill-climbing).

3.4 Behaviour in Spanish

The same experiments performed for English were performed with a Spanish corpus (CLiC-TALP Corpus[2]) which has about 100,000 words. This corpus is manually validated so, although it is small, it is more accurately tagged than WSJ.

In this case the tagger relies on FreeLing morphological analyser [14] instead of using a dictionary built from the corpus. Nevertheless, the situation is comparable to the English experiments above: Since the corpus and the morphological

[2] more information in `http://www.lsi.upc.es/~nlp/`

analyser have been hand-developed and cross-checked, they are mutually consistent, and so we don't have to care about unknown words in the test corpus.

3.4.1 Applying Lidstone's Law

In the same way than for the English corpus, a hill-climbing search is performed to study the influence of λ_A value in the precision of the system. The λ_π, λ_s and λ_w values are fixed to the same values used for the reduced WSJ.

		Trigram HMM			4-gram HMM		
Initial λ_A	$\Delta\lambda$	Initial precision	Final precision	Final λ_A	Initial precision	Final precision	Final λ_A
0.05	0.01	96.54	96.68	0.18	95.49	96.00	0.35
0.05	0.005	96.54	96.58	0.065	95.49	95.82	0.15
0.5	0.1	96.79	96.80	0.5	96.06	96.16	1.6
0.5	0.05	96.79	96.81	0.55	96.06	96.14	1.05
1.0	0.5	96.80	**96.85**	1.5	96.14	**96.22**	2.5
1.0	0.1	96.80	96.84	1.1	96.14	96.16	1.6

Table 5: Precision obtained with the hill-climbing algorithm on the Spanish corpus

Table 5 shows the results of these experiments. The best λ_A for the trigram HMM is $\lambda_A = 1.5$, yielding a precision of 96.85%. The best value for a 4-gram model is $\lambda_A = 2.5$, which produces a precision of 96.22%

3.4.2 Applying Linear Interpolation

The coefficients for Linear Interpolation are computed for Spanish in the same way than for English (section 3.2). The precision of the obtained taggers is 96.90% for the trigram HMM and 96.73% for the 4-gram model.

3.4.3 Best Configuration for Spanish

Results for Spanish are –as it may be expected– similar to those obtained with the reduced English corpus. Again, working with a trigram HMM gives higher precision than working with a 4-gram one, for both smoothing techniques, and using Linear Interpolation gives a slightly better results than using Lidstone's law. Table 6 summarizes the obtained results for both smoothing methods.

Nevertheless, some important remarks can be extracted from these results:

– Competitive HMM taggers may be build using relatively small train sets, which is interesting for languages lacking large resources.

100 Kword Spanish corpus

	Lidstone's law	Linear Interpolation
trigram	96.85%	**96.90%**
4-gram	96.22%	96.73%

Table 6: Obtained results for all HMM PoS tagger configurations using Spanish 100 Kwords corpus

- The best results are obtained using trigram models and Linear Interpolation.
- Lidstone's law may be used as an alternative smoothing technique, but if λ_A is not tuned, results are likely to be significantly lower.

4 Conclusions and Further Work

We have studied how competitive HMM-based PoS taggers can be developed using relatively small training corpus.

Results point out that accurate taggers can be build provided the appropriate smoothing techniques are used. Between both techniques studied here, in general the one that gives a higher precision is Linear Interpolation but Lidstone's law can reach, in many cases, similar precision rates if a search is performed through the parameter space to find the most appropriate λ_A.

The model proposed in [6] (trigram tagger, Linear Interpolation smoothing) is not only the more suitable for big training corpus but also it gives the best results for limited amounts of training data.

The use of four-gram models may result in a slight increase in precision when using large corpus. Nevertheless, the gain is probably not worth the increase in complexity and size of the model.

Further work to be performed includes dealing with unknown words, and study their influence on the taggers developed on small corpus. Also, we plan to port the same experiments to other languages (namely, Catalan) to further validate the conclusions of this paper.

References

[1] Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the 1st Conference on Applied Natural Language Processing, ANLP, ACL (1988) 136–143

[2] Cutting, D., Kupiec, J., Pedersen, J.O., Sibun, P.: A practical part–of–speech tagger. In: Proceedings of the 3rd Conference on Applied Natural Language Processing, ANLP, ACL (1992) 133–140

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

[4] Schmid, H.: Improvements in part–of–speech tagging with an application to german. In: Proceedings of the EACL SIGDAT Workshop, Dublin, Ireland (1995)

[5] Ratnaparkhi, A.: A maximum entropy part–of–speech tagger. In: Proceedings of the 1st Conference on Empirical Methods in Natural Language Processing, EMNLP. (1996)

[6] Brants, T.: Tnt - a statistical part- of-speech tagger. In: Proceedings of the 6th Conference on Applied Natural Language Processing, ANLP, ACL (2000)

[7] Karlsson, F.: Constraint grammar as a framework for parsing running text. In: Proceedings of 13th International Conference on Computational Linguistics, COLING, Helsinki, Finland (1990) 168–173

[8] Brill, E.: A Corpus–based Approach to Language Learning. PhD thesis, Department of Computer and Information Science, University of Pennsylvania (1993) http://www.cs.jhu.edu/~brill/acadpubs.html.

[9] Daelemans, W., Zavrel, J., Berck, P., Gillis, S.: Mbt: A memory–based part–of–speech tagger generator. In: Proceedings of the 4th Workshop on Very Large Corpora, Copenhagen, Denmark (1996) 14–27

[10] Viterbi, A.J.: Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. IEEE Transactions on Information Theory (1967) 260–269

[11] Laplace, P.S.m.: Philosophical Essay on Probabilities. Springer-Verlag (1995)

[12] Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. The MIT Press (1998)

[13] Civit, M.: Criterios de etiquetación y desambiguación morfosintáctica de corpus en español. PhD thesis, Linguistics Department, Universitat de Barcelona (2003)

[14] Carreras, X., Chao, I., Padró, L., Padró, M.: Freeling: An open-source suite of language analyzers. In: Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04), Lisbon, Portugal (2004)