A Trigram Part-of-Speech Tagger for the Apertium Free/Open-Source Machine Translation Platform

Zaid Md Abdul Wahab Sheikh
Computer Science and Engineering
National Institute of Technology
Allahabad-211004, India
`sheikh.zaid@gmail.com`

Felipe Sánchez-Martínez
Dept. Llenguatges i Sistemes Informàtics
Universitat d'Alacant
E-03071 Alacant, Spain
`fsanchez@dlsi.ua.es`

Abstract

This paper describes the implementation of a second-order hidden Markov model (HMM) based part-of-speech tagger for the Apertium free/open-source rule-based machine translation platform. We describe the part-of-speech (PoS) tagging approach in Apertium and how it is parametrised through a tagger definition file that defines: (1) the set of tags to be used and (2) constrain rules that can be used to forbid certain PoS tag sequences, thus refining the HMM parameters and increasing its tagging accuracy. The paper also reviews the Baum-Welch algorithm used to estimate the HMM parameters and compares the tagging accuracy achieved with that achieved by the original, first-order HMM-based PoS tagger in Apertium.

1 Introduction

Part-of-speech (PoS) tagging is a well-known problem and a common step in many natural-language processing applications such as rule-based machine translation (RBMT). A PoS tagger attempts to assign the correct PoS tag or *lexical category* to all words of a given text, usually by relying on the assumption that a word can be assigned a single PoS tag by looking at the PoS tags of the neighbouring words.

In RBMT PoS tags are usually assigned to words by looking them up in a lexicon, or by using a morphological analyser (Merialdo, 1994). A large portion of the words found in a text are *unambiguous* since they can be assigned only a single PoS tag; however, there are *ambiguous* words that can be assigned more than one PoS tag. For instance, the word *chair* can be, as many other English words, either a noun or a verb.

The choice of the correct PoS tag may be crucial when translating to another language. Mainly there are two reasons why a translation can be wrong due to PoS tagging errors:

1. Because some structural transformations are applied, or not, as a result of the PoS tag assigned to a word. For instance, in the translation into Spanish of the English text *the green house*, where *green* can be either a noun or an adjective, a reordering rule `det+adj+noun` → `det+noun+adj` is only applied if PoS tag adjective is assigned to word *green*.

2. Because the translation differs depending on how the PoS ambiguity is solved; e.g., the translation into Spanish of *chair* is *silla* when it is tagged as noun, and *presidir* when it is tagged as verb.

The Apertium free/open-source RBMT platform implements a first-order hidden Markov model (HMM)-based PoS tagger (Cutting et al., 1992; Rabiner, 1989). In this paper we describe in detail the implementation of a second-order HMM-based PoS tagger for Apertium and we provide a comparative study of the tagging accuracy both when using the first-order HMM-based and the second-order HMM-based PoS taggers.

The next section overviews the Apertium free/open-source RBMT platform with emphasis on the tagging approach it implements. Then in section 3 we describe in detail our implementation of a second-order HMM-based PoS tagger for Apertium. Section 4 presents the experiments conducted to test our implementation and the results achieved. The paper ends with some conclusions and future work plans.

2 The Apertium free/open-source machine translation platform

2.1 Machine translation approach

Apertium[1] (Armentano-Oller et al., 2006) follows the typical *transfer*-based approach to MT (Hutchins and Somers, 1992) that divides the translation into three phases: analysis, transfer and generation. More precisely, it follows the shallow-transfer approach shown in Figure 1, which may be considered to be somewhere between direct translation and full syntactic transfer, i.e., a *transformer* system (Arnold et al., 1994).

Analysis in Apertium consists of the morphological analysis and PoS tagging of the source language (SL) text to translate. The *transfer* phase consists of the lexical and structural transference; structural transference is based on the detection of fixed-length patterns of lexical categories that need some processing for agreement, word reordering, introduction of prepositions where needed, etc. In the *generation* phase the TL text is generated by properly inflecting the TL lexical forms obtained after the transfer phase and by performing some operation over the inflected forms such as contractions and apostrophations.

2.2 Part-of-speech tagging in Apertium

The HMM-based PoS tagger in Apertium gets its input from the morphological analyser module which segments the SL text in *surface forms* (lexical units as they appear in raw corpora) and delivers, for each surface form, all its possible *lexical forms* consisting of lemma, lexical category and morphological inflection information; e.g., the morphological analysis of the Spanish word *cantábamos* ("we sang") has lemma *cantar* ("sing"), lexical category verb, and inflection information: indicative, imperfect, 1st person, plural.

The PoS tagger is trained from corpora and a tagger definition file that specifies how the lexical forms delivered by the morphological analyser must be grouped into coarse tags. Grouping lexical forms into coarse tags is needed to reduce the amount of parameters of the HMM.

To reduce the amount of observable outputs, each one is made to correspond to a *word class*. Typically a word class is an *ambiguity class* (Cutting et al., 1992), that is, the set of all possible PoS tags that a word could receive. However, sometimes it may be useful to have finer classes, such as a word class containing only a single, very frequent, ambiguous word (Pla and Molina, 2004). In addition, unknown words, i.e. words not found in the lexicon, are usually assigned the ambiguity class consisting of the set of *open* categories, i.e. the set of PoS tags which are likely to grow by addition of new words to the lexicon: nouns, verbs, adjectives, adverbs and proper nouns.

Tagger definition file. SL lexical forms delivered by the morphological analyser convey detailed information about each surface form. This information is needed in some part of the RBMT engine, such as the structural transfer and the morphological generator, to carry out the translation. Nonetheless, for the purpose of efficient disambiguation, lexical forms with similar syntactic roles may be grouped in coarser tags (such as verb in personal form). This is because, while lexical forms are more informative, at the same time they considerably increase the number of HMM parameters to estimate, worsening the problem of data sparseness, and thus increasing the number of parameters that achieve a null-frequency count. In particular, when PoS taggers are used in MT, what really counts is to be able to distinguish analyses leading to different translations.

Coarse tags are defined in a XML-based[2] tagger definition file, the format of which is specified by DTD `tagger.dtd`.[3] XML element `def-label` defines a category or coarse tag by means of a list of fine-tags defined by one or

[1]The MT engine, documentation, and linguistic data for different language pairs can be downloaded from `http://www.apertium.org`.

[2]`http://www.w3.org/XML/`

[3]DTD provided by the `apertium` package

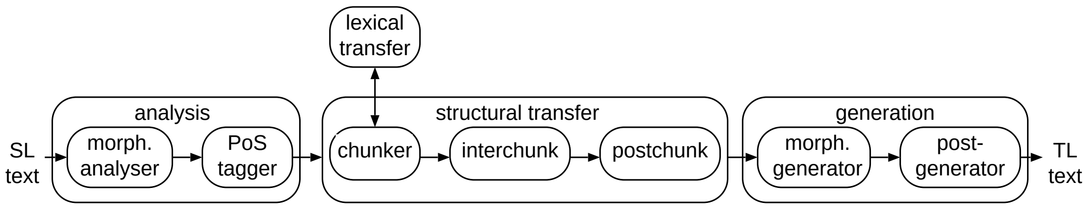

Figure 1: Architecture of the Apertium free/open-source RBMT platform.

more `tags-item` elements. It is also possible to define lexicalised coarse tags by specifying the lemma of the word in the attribute `lemma`.

In the tagger definition file it is also possible to define constraint rules in the form of forbid and enforce rules. Forbid rules define restrictions as sequences of coarse tags (`label-sequence`) that cannot occur. These sequences can be made up of only *two* `label-items` for the bigram tagger and up to *three* `label-items` in case of the trigram tagger. Forbid rules are used, for example to forbid a relative adverb at the beginning of a sentence (`SENT RELADV`), or a pronoun after a noun (`NOM PRNPERS`). Enforce rules are used to specify the set (`label-set`) of tags (`label-item`) that are allowed to occur after a particular PoS tag. These rules are applied to the HMM parameters by introducing zeroes in the state transition probabilities of forbidden PoS tag sequences and re-normalising.

3 A second-order HMM-based PoS tagger

In this section we describe our implementation of a trigram PoS tagger based on second-order hidden Markov models (HMM). Trigram taggers based on second-order HMMs have already been implemented in various projects, the most well-know implementations are the TnT tagger (Brants, 2000) and its open-source alternative, HunPos (Halácsy et al., 2007).[4] There is also an implementation of a trigram tagger in Acopost (A Collection Of Open Source PoS Taggers).[5]

3.1 Our implementation

In a second-order HMM the transition probability matrix is three dimensional, as the probability of transitioning to a new state depends not only on the current state but also on the previous state. Similarly, the probability of the model emitting a given word class depends not only on the current state but also on the previous state. Hence the transition probability matrix has dimensions $N \times N \times N$ while the emission probability matrix has dimensions $N \times N \times M$, where N is the number of states (PoS tags) and M is the number of observable outputs (ambiguity classes).

The tagger supports both the normal (bigram) forbid and enforce rules as well as extended rules to match three consecutive PoS tags. A sequence of three PoS tags is forbidden if it matches any extended forbid rule or does not match an extended enforce rule or if any of the two consecutive tag pairs in the sequence of three PoS tags matches a bigram forbid rule or fails to match a bigram enforce rule.

3.1.1 Assumptions

1. The text sequence $O_1 \ldots O_T$ being disambiguated is always preceded by two unambiguous word $O_{-1} = \{I\}$ and $O_0 = \{I\}$.

2. The text to disambiguate is ended by two unambiguous words $O_T = \{E_1\}$ and $O_{T+1} = \{E_2\}$.

3. All word classes (observable outputs) contain, at least, the correct PoS tag.

We have chosen the end-of-sentence tag for I, E_1 and E_2; this makes the PoS tagging of each sentence independent of the position of the text in which it appears.

3.1.2 Parameter estimation

The process of estimating the HMM parameters consists in finding the set of parameters that maximises the mathematical expectation of the

[4]`http://code.google.com/p/hunpos/`
[5]`http://acopost.sourceforge.net`

observed sequences. The classic methods to estimate these parameters are:

1. Training the model in an unsupervised way from untagged corpora via the Baum-Welch expectation-maximisation (EM) algorithm (Baum, 1972). In an untagged corpus each word has been assigned the set of all possible PoS tags that it can receive; untagged corpora can be easily obtained if a morphological analyser is available.

2. Training the model in a supervised manner with hand-tagged corpora via the maximum-likelihood estimate (MLE; Gale and Church 1990). In a hand-tagged corpus all ambiguities have been manually solved; thus, this is a direct method to estimate the HMM parameters that consists of collecting frequency counts from the training corpus and using these counts to estimate the HMM parameters.

3.1.3 Baum-Welch EM algorithm

The Baum-Welch algorithm is an iterative algorithm that works by giving the highest probability to the state transitions and output emissions used the most. After each Baum-Welch iteration the new HMM parameters may be shown to give a higher probability to the observed sequence (Baum, 1972).

A description of the Baum-Welch algorithm for a first-order HMM is provided by Rabiner (1989), formulas for a second-order HMM are described by Kriouile et al. (1990). What follows is a revision of the Baum-Welch algorithm using the notation used by Rabiner (1989) and Sánchez-Martínez (2008, Appendix B).

Along this section we assume a second-order HMM λ whose parameters are a_{ijk} (transition probability) and $b_{jk}(O_t)$ (emission probability) for the sequence of states (PoS tags) $s_i s_j s_k$ and the observable output O_t.

Forward probabilities. The forward function is defined as the probability of the partial observation sequence from 1 to time $t+1$, and transition $s_j s_k$ at times $t, t+1$, given the model λ :

$$\alpha_{t+1}(j,k) = \left[\sum_{i=1}^{N} \alpha_t(i,j) a_{ijk}\right] b_{jk}(O_{t+1})$$

$$\alpha_0(i,j) = \begin{cases} 1 & s_i, s_j = unambiguous \\ 0 & otherwise \end{cases}$$

Backward probabilities. The backward function is defined as the probability of the partial observation sequence from $t+1$ to T, given transition $s_i s_j$ at times $t-1, t$ and the model λ :

$$\beta_t(i,j) = \sum_{k=1}^{N} a_{ijk} b_{jk}(O_{t+1}) \beta_{t+1}(j,k)$$

$$\beta_{T+1}(i,j) = 1$$

Other probabilities. The probability of a sequence $O = O_1 ... O_T$ can be calculated using the forward and backward probabilities in the following way:

$$P(O|\lambda) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i,j) \beta_t(i,j)$$

where t can be freely chosen from the interval $[0,T]$; in particular,

$$P(O|\lambda) = \alpha_T(E_1, E_2)$$

This equality is a direct consequence of the second assumption (see Section 3.1.1).

The number of times the transition $s_i s_j$ is seen during the generation of the sequence of observable outputs O is defined as:

$$\Gamma_{ij} = \frac{1}{P(O|\lambda)} \sum_{t=1}^{T} \alpha_t(i,j) \beta_t(i,j)$$

The expected number of times that the transition $s_i s_j s_k$ is performed during the generation of the sequence of observable outputs O, Ξ_{ijk} , is:

$$\frac{1}{P(O|\lambda)} \sum_{t=1}^{T} \alpha_t(i,j) a_{ijk} b_{jk}(O_{t+1}) \beta_{t+1}(j,k)$$

The number of times the model emits the word class v_k from hidden state s_j with the previous state being s_i when generating the sequence of observable outputs O is defined as:

$$\Phi_{ijk} = \frac{1}{P(O|\lambda)} \sum_{t=1}^{T} \alpha_t(i,j) \beta_t(i,j) \delta_{v_k, O_t}$$

$$\delta_{v_k,O_t} = \begin{cases} 1 & v_k = O_t \\ 0 & otherwise \end{cases}$$

New parameters. From the previous equations, and after processing the whole corpus, the HMM parameters can be updated through the following Baum-Welch general equations:

$$a_{ijk} = \frac{\Xi_{ijk}}{\Gamma_{ij}}$$

$$b_{ij}(k) = \frac{\Phi_{ijk}}{\Gamma_{ij}}$$

Here, Ξ_{ijk} is an estimation of $\tilde{n}(s_i s_j s_k)$, the number of times the state s_i is followed consecutively by s_j and s_k in the training corpus. Γ_{ij} is an estimation of $\tilde{n}(s_i s_j)$, the number of times the state s_i is followed by s_j. Φ_{ijk} is an estimation of $\tilde{n}(v_k, s_i s_j)$, the number of times word class v_k is emitted from tag s_j when the previous tag is s_i. These count estimations of bigrams, trigrams and ambiguity classes are used to calculate the bigram and trigram probabilities and smoothing coefficients during parameter smoothing (See Section 3.1.4). Number of occurrences $\tilde{n}(s_i)$ of PoS tags s_i can be estimated as

$$\Gamma_i = \sum_{j=1}^{N} \Gamma_{ij}$$

The number of times $\tilde{n}(v_k, s_j)$ ambiguity class v_k is emitted from PoS tag s_j can be estimated as

$$\Phi_{jk} = \sum_{i=1}^{N} \Phi_{ijk}$$

Finally, the word-class frequency count $m(v_k)$ can be calculated from the training corpus.

Segmentation. When a given text contains a sequence of two unambiguous words, the HMM can only be in the state corresponding to the PoS tag of the second ambiguity class with the previous state being the PoS tag corresponding to the first ambiguity class. This property allows for a more efficient implementation, because it is not necessary to store the whole text, but the sequence of words between two unambiguous word pairs (both included) and treat that sequence of words as the whole text. Thus the above equations can be applied locally to each segment g as if they were completely independent texts. Then the needed values can be calculated as:

$$\Xi_{ijk} = \sum_{g=1}^{G} \Xi_{ijk}^{(g)}$$

$$\Phi_{ijk} = \sum_{g=1}^{G} \Phi_{ijk}^{(g)}$$

$$\Gamma_{ij} = \sum_{g=1}^{G} \Gamma_{ij}^{(g)}$$

where G is the total number of segments.

Parameter initialisation. Baum-Welch algorithm assumes that the HMM parameters have been previously initialised. In absence of knowledge the HMM parameters can be initialised using the method proposed by Kupiec (1992).

Kupiec's (1992) method estimates the frequency counts needed to calculate the HMM parameters by processing the training corpus and collecting: observable output occurrences $m(v_l)$, observable output pair occurrences $m(v_l v_m)$ and observable output triple occurrences $m(v_l v_m v_n)$. Then the frequency counts aforementioned are estimated:

$$\tilde{n}(s_i s_j s_k) = \sum_{v_l : s_i \epsilon v_l} \sum_{v_m : s_j \epsilon v_m} \sum_{v_n : s_k \epsilon v_n} \frac{m(v_l v_m v_n)}{|v_l||v_m||v_n|}$$

$$\tilde{n}(s_i s_j) = \sum_{v_l : s_i \epsilon v_l} \sum_{v_m : s_j \epsilon v_m} \frac{m(v_l v_m)}{|v_l||v_m|}$$

$$\tilde{n}(s_i) = \sum_{v_l : s_i \epsilon v_l} \frac{m(v_l)}{|v_l|}$$

$$\tilde{n}(v_k, s_i s_j) = \begin{cases} \sum_{v_l : s_i \epsilon v_l} \frac{m(v_l v_k)}{|v_l||v_k|} & s_j \epsilon v_k \\ 0 & otherwise \end{cases}$$

$$\tilde{n}(v_k, s_j) = \begin{cases} \frac{m(v_k)}{|v_k|} & s_j \epsilon v_k \\ 0 & otherwise \end{cases}$$

3.1.4 Parameter Smoothing

The data sparseness problem which occurs due to the number of parameters to estimate in relation to the size of the training data, becomes even more acute in the case of a trigram tagger. To avoid null probabilities for those state-to-state transitions and output emissions that have not been seen in the training corpus, we employ a form of deleted interpolation (Jelinek, 1997) for parameter smoothing in which weighted estimates are taken from second-order and first-order models and a uniform probability distribution. The smoothed trigram probabilities consist of a linear combination of trigram and bigram probabilities where the values of the smoothing coefficients are computed using the *successive linear abstraction* method (Brants and Samuelsson, 1995).

3.1.5 Viterbi algorithm

The Viterbi algorithm for a second-order HMM, as described by Thede and Harper (1999) is used to disambiguate a sentence using the model we have trained above. It can be applied to text segments delimited by two unambiguous words.

4 Experiments

We studied the PoS tagging performance of the second-order HMM-based PoS tagger on the Spanish language when it is trained both in a supervised way from hand-tagged corpora and in an unsupervised way from untagged corpora of different sizes. We compare the tagging accuracy achieved with that achieved by the first-order HMM-based PoS tagger in Apertium. In addition, we report the tagging accuracy both when using forbid and enforce rules and when these rules are not used.

For the supervised training, a Spanish hand-tagged corpus with 21,803 words was used.[6] The PoS tagging error rate was evaluated using an independent Spanish hand-tagged corpus having 8,068 words; the percentage of ambiguous words (according to the lexicon) in this corpus is 23.71%, while the percentage of unknown words is 3.88%.

For the unsupervised training we used Spanish raw corpora coming from texts of the Spanish news agency EFE.[7] Different corpus sizes ranging from 6 to 145 million words have been used; larger corpora including shorter ones. The PoS tagging accuracy of the PoS taggers trained in an unsupervised way was evaluated using the hand-tagged corpus with 21,803 words aforementioned; the percentage of ambiguous words and unknown words in this corpus are 22.41% and 3.25%, respectively.

The tagger definition file used is the one provided in the Apertium language-pair package `apertium-es-ca-1.0.2`. The tagset defines 99 coarse tags (85 single-word and 14 multi-word tags for contractions, etc) grouping the 2,116 fine tags (377 single-word and 1,739 multi-word tags) delivered by the morphological analyser. The tagger definition file also defines 357 bigram forbid rules and one bigram enforce rule. The number of ambiguity classes is 291.

Table 1 reports the PoS tagging error rate over the ambiguous words only, over all the ambiguous words (including unknown words) and over all words (ambiguous and unambiguous). Both the trigram tagger and the bigram tagger performs better when using the forbid and enforce rules defined in the tagger definition file. Notice that the bigram tagger provides better results than the trigram tagger in both cases.

Figure 2 shows the PoS tagging error rate of the bigram tagger and the supervised tagger both when using the forbid and enforce rules defined in the tagger definition file, and when this rules are not used. The results reported show that the trigram tagger performs worse than the bigram tagger even when a large corpus with around 145 million words is used for training.

The results in Table 1 and Figure 2 clearly show that the use of forbid and enforce rules considerably increases the PoS tagging accuracy, especially when the unsupervised training is applied. In addition, the improvement in accuracy is much larger in the case of the trigram tagger.

5 Discussion and future work

The experiments conducted show that the bigram tagger performs better than the trigram tagger in all cases. The fact that the trigram tagger performs worse than the bigram tagger when both are trained in a supervised way seems reasonable given the small amount of hand-tagged corpora used and the number of parameters to estimate. However, in the case of the unsupervised training one would expect the trigram tagger to out-

[6]We have not used larger, freely-available annotated corpora such as the Spanish Ancora corpus (`http://clic.ub.edu/ancora/`) because they are not fully compatible with the lexical forms and multiword units defined in the Spanish monolingual dictionary used.

[7]`http://www.efe.com`

training method	**errors over**	**bigram**		**trigram**	
		with rules	no rules	with rules	no rules
supervised	ambiguous words	7.37 %	7.42 %	7.99 %	8.47 %
	ambiguous & unknown words	14.37 %	14.37 %	14.96 %	15.45 %
	all words	4.76 %	4.76 %	4.92 %	5.06 %
unsupervised	ambiguous words	22.76 %	24.98 %	24.94 %	32.28 %
	ambiguous & unknown words	26.83 %	30.70 %	31.77 %	38.50 %
	all words	8.64 %	9.12 %	9.39 %	11.12 %

Table 1: PoS tagging error rate over the ambiguous words, over all ambiguous words (including unknown words) and over all words (ambiguous and unambiguous) when the Spanish PoS tagger is trained in a supervised manner on a 21,803 word hand-tagged corpus and in an unsupervised manner on a 145 million raw text corpus.

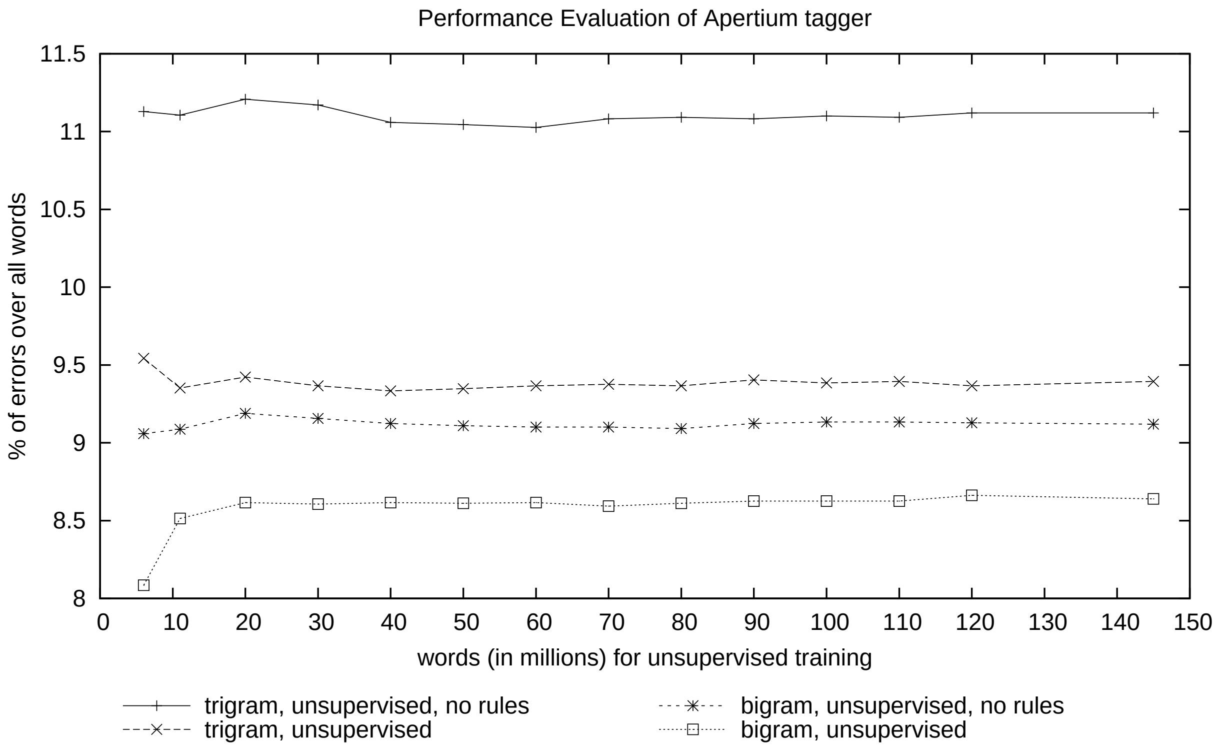

Figure 2: PoS tagging error rate over all words when the taggers are trained in an unsupervised way from untagged corpora of different sizes (horizontal axis).

perform the bigram tagger. The differences in the output of the trigram tagger and the bigram tagger when tagging the test corpus do not give any insights on this. To further investigate this phenomenon we plan to use even larger corpora and a better smoothing method. We also plan to train PoS taggers for other languages in Apertium such as French or English. Source code is available through the Apertium SVN repository (`https://apertium.svn.sf.net/svnroot/apertium/branches/gsoc2009/disismt/`)

Acknowledgements

We acknowledge Google for sponsoring this project through the Google Summer of Code 2009 program, and the Spanish Ministry of Science and Innovation for partially supporting this work under project TIN2009-14009-C02-01. We also thank Mikel L. Forcada for his help with the Baum-Welch expectation-maximisation algorithm.

References

Armentano-Oller, C., Carrasco, R. C., Corbí-Bellot, A. M., Forcada, M. L., Ginestí-Rosell, M., Ortiz-Rojas, S., Pérez-Ortiz, J. A., Ramírez-Sánchez, G., Sánchez-Martínez, F., and Scalco, M. A. (2006). Open-source Portuguese-Spanish machine translation. In *Computational Processing of the Portuguese Language, Proceedings of the 7th International Workshop on Computational Processing of Written and Spoken Portuguese, PROPOR 2006*, volume 3960 of *Lecture Notes in Computer Science*, pages 50–59. Springer-Verlag.

Arnold, D., Balkan, L., Meijer, S., Humphreys, R., and Sadler, L. (1994). *Machine translation: An introductory guide*. NCC Blackwell, Oxford.

Baum, L. E. (1972). An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process.

Brants, T. (2000). Tnt - a statistical part-of-speech tagger.

Brants, T. and Samuelsson, C. (1995). Tagging the Teleman corpus. In *Proceedings of the 10th Nordic Conference of Computational Linguistics*.

Cutting, D., Kupiec, J., Pedersen, J., and Sibun, P. (1992). A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing. Association for Computational Linguistics. Proceedings of the Conference.*, pages 133–140.

Gale, W. A. and Church, K. W. (1990). Poor estimates of context are worse than none. In *HLT '90: Proceedings of the workshop on Speech and Natural Language*, pages 283–287, Morristown, NJ, USA. Association for Computational Linguistics.

Halácsy, P., Kornai, A., and Oravecz, C. (2007). HunPos - an open source trigram tagger. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume. Proceedings of the Demo and Poster Sessions*, pages 209–212, Prague, Czech Republic.

Hutchins, W. J. and Somers, H. L. (1992). *An Introduction to Machine Translation*. Academic Press.

Jelinek, F. (1997). *Statistical Methods for Speech Recognition*. The MIT Press.

Kriouile, A., Mari, J.-F., and Haon, J.-P. (1990). Some improvements in speech recognition algorithms based on hmm. In *Acoustics, Speech, and Signal Processing, 1990. ICASSP-90., 1990 International Conference on*, pages 545–548 vol.1.

Kupiec, J. (1992). Robust part-of-speech tagging using a hidden markov model.

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

Pla, F. and Molina, A. (2004). Improving part-of-speech tagging using lexicalized HMM. *Journal of Natural Language Engineering*, 10(2):167–189.

Rabiner, L. R. (1989). A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286.

Sánchez-Martínez, F. (2008). *Using unsupervised corpus-based methods to build rule-based machine translation systems*. PhD thesis, Universitat d'Alacant.

Thede, S. M. and Harper, M. P. (1999). A second-order hidden markov model for part-of-speech tagging. In *In Proceedings of the 37th Annual Meeting of the ACL*, pages 175–182.