Practical application of one-pass Viterbi algorithm in tokenization and part-of-speech tagging

Miguel A. Molinero Álvarez
Universidad de La Coruña
Facultad de Informática
Campus de Elviña S/N
15071 - A Coruña, Spain
mmolinero@udc.es

Fco. Mario Barcala Rodríguez
Centro Ramón Piñeiro
Ctra. Santiago-Noia Km. 3. A Barcia
15896 - Santiago de Compostela, Spain
fbarcala@cirp.es

Juan Otero Pombo
Universidad de Vigo
E.S. de Ingeniería Informática
Campus das Lagoas S/N
32004 - Ourense, Spain
jop@uvigo.es

Jorge Graña Gil
Universidad de La Coruña
Facultad de Informática
Campus de Elviña S/N
15071 - A Coruña, Spain
grana@udc.es

Abstract

Sentence word segmentation and Part-Of-Speech (POS) tagging are common preprocessing tasks for many Natural Language Processing (NLP) applications. This paper presents a practical application for POS tagging and segmentation disambiguation using an extension of the one-pass Viterbi algorithm called Viterbi-N. We introduce the internals of the developed system, which is based on lattices and a stochastic model built using second order Hidden Markov Models (HMMs). Also, we present the results of an evaluation process and the analysis of the error cases. The results achieved suggest that the Viterbi-N algorithm applied on lattices allows POS tagging and segmentation disambiguation to be accomplished in a common process. Although the tests were done for the Galician language, the solution proposed could be easily exported to other languages.

Keywords

Part-Of-Speech tagging, tokenization, segmentation disambiguation, Hidden Markov Models, lattices.

1 Introduction

Current Part-Of-Speech (POS) taggers assume that their input is already correctly tokenized. This means that every token in the input is an individual linguistic component suitable for being tagged with a single POS tag. The tokenization task tends to be relatively simple, since in most cases each word corresponds to one linguistic token. However, there are cases where this segmentation can be more complex. On one hand, there are contractions and verbal forms with enclitic pronouns, where the same word contains information about two or more linguistic components which have to be split into individual tokens. On the other, there are idioms, where several words act together as one linguistic component, and must be joined to form a unique compound token.

Segmentation ambiguities arise when one or more words can be segmented into linguistic tokens in more than one way. This kind of phenomenon is quite common in languages with a rich morphology, such as Spanish or Galician. To deal with such ambiguities, several works [8] [9] use artificial tags to be assigned to compound tokens or to tokens which are part of only one linguistic reality. However, they postpone the solution of these segmentation tasks to later phases of Natural Language Processing (NLP), which in most cases are not documented.

Our approach lies in using the one-pass Viterbi algorithm extension [6] over second order Hidden Markov Models (HMMs) to carry out the segmentation just at the moment of assigning POS tags. Segmentation ambiguities are detected by a morphological preprocessor using lexicons and provided as input to the algorithm.

This way, POS tagging and segmentation disambiguation are accomplished in one unique process using a lattice structure. Lattices will allow us to represent every possible segmentation and to manage all the computations needed for the classic Viterbi algorithm at the same time, as we will explain later.

2 Segmentation Issues

As we have indicated earlier, many POS tagging environments simply ignore segmentation issues, leaving them to be solved in later steps. For example, a common approach is to use agglutinations of tags[1] which are assigned to contractions and enclitic

[1] To simplify, in this work, we use Adj for adjective, Adv for adverb, C for conjunction, Det for determiner, P for

forms. A contraction formed by a preposition and a determiner could be tagged with a compound tag like *P+Det*, instead of being split into one token tagged with *P* and another one tagged with *Det*. Given that many NLP applications need to know the linguistic information of each word component, when using this approach the contraction will need to be processed in a later step in order to extract its linguistic information. Moreover, it causes an unnecessary growth of the tagset, with its negative consequences (sparse data, larger training corpus needed, etc.) [4].

In comparison, we detect tokenization ambiguities just before the POS tagging phase with a morphological preprocessor [7]. This is done using external lexicons and some segmentation rules for verbal forms with enclitic pronouns. If a word can make sense with different segmentations, the morphological preprocessor provides every alternative to the POS tagger. Then, the POS tagger will choose the best one.

Fig. 1: *Ambiguous segmentations of 'polo' and 'sin embargo' for Galician and Spanish languages respectively.*

Contractions, verbal forms with enclitic pronouns, idioms and proper nouns are the categories which are able to generate segmentation ambiguities. For example, as we can see in figure 1, the Galician word '*polo*' could be treated as a noun (chicken), as a contraction of the preposition '*por*' and the determiner '*o*' (by the) or even as a verbal form '*pos*' with the enclitic pronoun '*o*' (put it). On the other hand, a sequence of words like the Spanish expression '*sin embargo*' could be joined together and tagged as a conjunction (however) or it could be tagged individually as a preposition and a noun (without seizure).

Once a sentence has been preprocessed and segmentation ambiguities detected, a tagging model is used to assign the correct POS tag to each of the tokens. The model is built as a second order Hidden Markov Model (HMM) and its parameters are estimated from a training corpus using linear interpolation of uni-, bi- and trigrams as our smoothing technique [5].

3 Lattices

In the context of POS tagging with HMMs, the classic version of the Viterbi algorithm is applied on trellises [3], where the first row contains the words of the sentence to be tagged, and the candidate tags appear in columns below the words. However, this structure does not allow the representation of ambiguous segmentations.

A practical solution lies in using lattices to represent sentences. Figure 2 shows a Galician language sentence which contains several types of ambiguous segmentations: '*Non poden verse a causa de certo individuo*' (they cannot meet each other because of a certain person). The gaps between the words are enumerated and an arc can span one or more words. Such an arc is labelled with the words spanned and their corresponding POS tag. For example, gap 3 marks the begining of an ambiguous segmentation for the word '*verse*'. It could be segmented into verb '*ver*' (to meet) and reflexive enclitic pronoun '*se*', or as verb '*verse*' (it may deal with). In gap 5, the idiom '*a causa de*' (because of) could be also segmented into several different tokens and the same in gap 7 for '*de certo*' (certainly).

Although there are 40 possible paths in this sentence, only the one formed by the arcs drawn in the upper part of the lattice shows the correct segmentation. Each arc represents a token, so the correct segmentation is seven tokens long, while the longest possible segmentation of this sentence is nine tokens long.

Therefore, lattices will allow us to represent all the information about ambiguous segmentations. Now we will see how an extension of the Viterbi algorithm can use them to tag sentences without repeating computations for each path.

4 Viterbi-N: the one-pass Viterbi algorithm with normalization

The Viterbi algorithm [10] is a dynamic programming algorithm for finding the most likely sequence of hidden states (called the Viterbi path) that explains a sequence of observations for a given stochastic model. In the context of POS tagging, we are looking for the most likely sequence of tags that explains a sequence of words in a sentence. In order to do so, a trellis is built from the sentence to be tagged. For each state (tag) in that trellis the cumulative probability for all paths reaching that state must be computed but, given that such paths in trellises have the same length, it is only necessary to store the cumulative probability of the best one. At the end, the most likely sequence of tags for the sentence is obtained by comparing cumulative probabilities of final states and going backwards.

On the contrary, the Viterbi-N algorithm is applied on lattices [6], so it is possible to reach one state coming from paths of different length. Thus, for each state in the lattice, it will be necessary to store as many cumulative probabilities as there are different lengths of path reaching that state. Therefore, let $\Delta_{t,t',l(q)}$ be an accumulator which collect the maximum probability of state q covering words from position t to t', and with length l, l being the number of states from first state to state t'.

Only accumulators with the same length would be directly comparable, because of the different number of factors involved in their computation.

preposition, Pro for pronoun, N for noun, V for verb, Id for idiom and Q for punctuation mark.

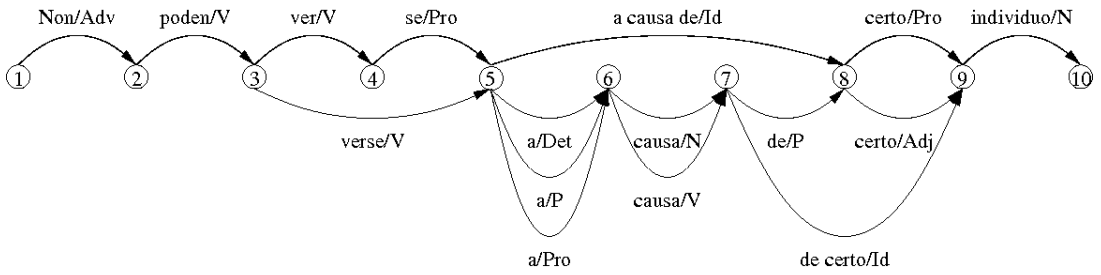

Fig. 2: *Ambiguous segmentations represented on a lattice.*

As accumulators are computed by products of probabilities, longer segmentations are penalized by the higher number of factors, making them less likely than shorter ones. In practice, this means that alternatives which imply the joining of words would be chosen more often than others which imply segmentation into several words. To solve this, a normalization step must be accomplished in order to compare segmentation paths of different lengths. Moreover, it must be noted that the algorithm works using only one lattice and performing only one pass of the Viterbi algorithm.

Figure 3 shows how the algorithm is applied on the Galician language sentence '*Non poden verse.*' (they cannot meet each other.). Lattices can be implemented as graphs in which each node is a probability accumulator associated to one linguistic token and one POS tag. In the figure we can see the accumulators needed to tag and disambiguate this sentence. Such accumulators are written with the format $\Delta(t, t', l, q)$, where t and t' are the instants where the current token starts and ends, l is the number of tokens from the beginning of the sentence until the current token and q is the associated POS tag. As there are two possible segmentation paths reaching the last token of the sentence, it has two accumulators, with lengths 5 and 4. The algorithm will normalize both accumulators by their lengths and choose the best one. Then, the sequence of tags compounding the best segmentation path can be obtained by going backwards in the lattice.

The equations of the classic Viterbi algorithm can be adapted to process lattices [2]. Assuming the use of logarithmic probabilities to avoid problems of precision with factors less than 1, we replace products by sums and adapt the Viterbi-N algorithm's equations as follows:

Let's use $\delta_{i,j}(q)$ to denote the probability of the derivation emitted by state q having a terminal yield that spans positions i to j.

- Initialization:

$$\Delta_{0,t,1}(q) = P(q|q_s) + \delta_{0,t}(q)$$

- Recursion:

$$\Delta_{t,t',l}(q) = \max_{(t'',t,q')\in \text{Lattice}} \Delta_{t'',t,l-1}(q') + P(q|q') + \delta_{t,t'}(q) \quad (1)$$

$$\text{for } 1 \leq t < T$$

- Termination:

$$\max_{Q\in\mathcal{Q}^*} P(Q, \text{Lattice}) = \max_{l} \frac{\max\limits_{(t,T,q)\in \text{Lattice}} \Delta_{t,T,l}(q) + P(q_e|q)}{l}$$

Additionally, it is also necessary to keep track of the elements in the lattice that maximized each $\Delta_{t,t',l}(q)$. When reaching time T, we get the length of the best path in the lattice:

$$L = \arg\max_{l} \frac{\max\limits_{(t,T,q)\in \text{Lattice}} \Delta_{t,T,l}(q) + P(q_e|q)}{l}$$

Next, we get the best last element of all paths of length L in the lattice:

$$(t_1^m, T, q_1^m) = \arg\max_{(t,T,q)\in \text{Lattice}} \Delta_{t,T,L}(q) + P(q_e|q)$$

Setting $t_0^m = T$, we collect the arguments $(t'', t, q') \in$ Lattice that maximized equation (1) by going backwards in time:

$$(t_{i+1}^m, t_i^m, q_{i+1}^m) =$$

$$\arg\max_{(t'',t_i^m,q')\in \text{Lattice}} \Delta_{t'',t_i^m,L-i}(q') + P(q_i^m|q') + \delta_{t_i^m,t_{i-1}^m}(q_i^m)$$

for $i \geq 1$, until we reach $t_k^m = 0$. Now, $q_1^m \ldots q_k^m$ is the best sequence of phrase hypothesis (read backwards).

To sum up, the normalized probabilities calculated by the Viterbi-N are directly compared and the highest one is chosen to build the best segmentation path for current sentence.

5 Defining alternatives

The input for the algorithm is based on the input format of classic taggers [3]. That is, one word per line, optionally followed by its candidate POS tags. However, this classic representation does not allow the inclusion of segmentation alternatives.

We have decided to use XML-like tags for the definition of such alternatives. An alternative structure starts with a line containing only the tag `<alternatives>`. Then, each segmentation alternative starts with a line containing only the tag `<alternative>` and ends with the tag `<\alternative>`. Between those tags, the segmentation alternatives are presented using the classic format. Finally, the alternative structure ends with

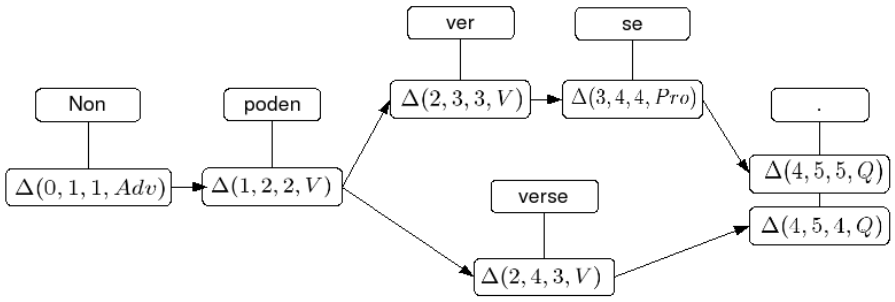

Fig. 3: *Viterbi-N algorithm applied on a lattice*

the tag <\alternatives>. For example, the alternative structure for the galician word '*polo*' (see figure 1) would be as follows:

```
<alternatives>
  <alternative>
       polo      N
  </alternative>
  <alternative>
       por       P
       o         Det
  </alternative>
  <alternative>
       pos       V
       o         Pro
  </alternative>
</alternatives>
```

Alternative structures may appear at any place inside a sentence. Their construction is a task that should be accomplished by the previously mentioned morphological preprocessor[2]. It should build every alternative and assign candidate tags in each branch. It must be noted that some branches may already have already the correct POS tags (e.g. contractions have usually unique tags when they are segmented), providing valuable information that can be used to choose the correct alternative.

6 Evaluation

We have performed three experiments using Galician language texts obtained from the "Reference Corpus from Present-day Galician Language" project [1] to test the accuracy of our approach. We have implemented the Viterbi-N algorithm over a lattice structure, and fed it with the input described in section 5. The main goal of these tests is to establish both how accurate the segmentation disambiguation process is, and how dependent it is from the trained model.

We worked with a manually tagged corpus, containing 115754 words and organized in 3920 sentences. In this corpus, our morphological preprocessor detects 1967 sentences with at least one ambiguous segmentation. The whole number of segmentation ambiguities in the corpus is 3037.

Our first experiment (E1), only to figure out the possibilities of our system, consisted in tagging ambiguous sentences from the training corpus. A high degree of accuracy would be expected in this experiment, since there are no unknown words in the text. We performed this experiment on a set of 434 sentences randomly extracted from the training corpus, with the only requisite of containing at least one ambiguous segmentation. This set contained 702 cases of ambiguous segmentations.

For the second experiment (E2), we randomly extracted 185 sentences, again containing at least one ambiguous segmentation. These sentences are formed by 6073 words, and were used as a testing corpus. The remaining 109681 words were used as a training corpus.

As a high number of segmentation ambiguities remained undetected in experiment E2, we decided to carry out a third experiment avoiding this problem. Thus, in the third experiment (E3), we again tagged the extracted testing corpus, but with an improved version of the morphological preprocessor, which is able to detect new ambiguous segmentations, not detected in experiment E2.

Although the size of the testing corpus could seem a little small, we have chosen such a size for three reasons. First, the Galician language is a less-resourced language, so the amount of tagged text available was small. Second, it is difficult to align manual and automatic tagged text to compare results when alternative segmentation options are given. Therefore, with a small corpus errors could be easily detected and checked. Third, we wanted to make a detailed study of the error cases in order to determine where they come from, and how to avoid them.

Table 1 shows the experimental results. The first column shows the number of ambiguous segmentations detected by the preprocessor. The second column shows the number of segmentations where the correct segmentation was chosen. The third shows the number of ambiguous segmentations not detected by the preprocessor. The next column shows the percentage accuracy of the segmentation disambiguation taking cases of the third column as errors, and the last one shows the accuracy of the segmentation disambiguation process when the cases of the third

[2] Details about how the preprocessor accomplishes this lie outside the scope of this work [7].

	Cases	Good choice	No option given	Total Accuracy	Real Accuracy
E1	702	662	8	94.30%	95.39%
E2	309	241	41	77.99%	89.92%
E3	309	255	5	82.52%	83.88%

Table 1: *Test results for experiments E1, E2 and E3.*

column are not treated as errors.

As expected, experiment E1 produced very good results. Only 8 cases of ambiguous segmentation were not detected by the morphological preprocessor. We cannot consider these cases as real errors, since no alternatives are given to the algorithm and they could be detected just by upgrading the lexicons used by the morphological preprocessor. The real accuracy achieved in this experiment is over 95%.

Experiment E2 is a more natural one, because unknown words appear in the testing corpus. As can be seen, there is a high number of ambiguous segmentations not given by the preprocessor. This fact has a simple explanation: idioms which are in the corpus but not included in the morphological preprocessor lexicons, unknown enclitic forms, etc. A human linguist is able to detect them, but our preprocessor simply does not have the necessary information to do so. Once again, if we do not take these cases as errors, the accuracy is 89.92%. This accuracy descends to 77.99% if we treat them as errors.

For experiment E3, we added to the lexicons of the morphological preprocessor many of the unknown cases of experiment E2. In fact, all but those that do not meet the usual criteria for inclusion in a lexicon (Latin or foreign idioms, etc.). Now, we have to keep in mind that these new added cases are not in the trained model, so some branches of an alternative segmentation could be an unknown word. In these conditions, which could be considered as the worst case for our system, we achieved 83.88% accuracy. We judge this value as a real approximation to the overall accuracy of the system in segmentation disambiguation and we adopt it as a baseline for future developments.

Although the results obtained were not outstanding, we believe it is a very promising technique. We must note that the training corpus used is very small for the size of the tagset[3] and at the moment we have no more corpora available. In fact the training corpus is still under development and the one used here has a lack of coherence. So we think most errors come from the training corpus and not from the technique itself. Unfortunately, we have no other approaches to compare with, or we do not know any other work which explains and tests the segmentation disambiguation for Western European languages.

Concerning the pure POS tagging results, they are subordinate to the success of the tokenization task. Taking each segmention error as one POS tagging error, we achieved 87.14% accuracy in experiment E3. We have checked that this poor result comes once again from the poverty of the training corpus.

[3] The tagset used has near 300 different tags. It can be consulted in http://corpus.cirp.es/xiada/etiquetario.html

6.1 Error analysis

In a detailed analysis of the errors, we became aware of some interesting points. First, we have detected two different kinds of error, which we could classify as soft and hard:

- Soft errors are those from idioms. Such errors arise when several words are not joined into an idiom, but are correctly tagged individually, or when they are joined into an idiom when they should not be. These kinds of error choose segmentations that commonly make sense with the rest of the sentence. In some cases it is not even clear for linguists when some idioms should be built, so the information of the model is limited for this purpose.

- Hard errors are those from contractions, enclitic forms, etc. If the correct segmentation is not chosen in such cases, the error is very hard, since it could even start a cascade error for the rest of the sentence. As a result of this kind of error, the tagged sentence makes no sense and it could be considered a whole tagging error. For example, in the Galician sentence '*o polo comeu millo*' (the chicken ate corn), if polo is segmented as a contraction, we will have '*o por o comeu millo*' (the by the ate corn), a completely meaningless sentence.

Table 2 shows the rates of soft and hard errors detected in experiment E3. As can be seen, we achieved 63.56% accuracy for idioms. Further analysis of the training corpus revealed that it was very poor in idioms. Linguists who tagged it, usually chose not to join several words to make an idiom, even when it was possible. Therefore, the training corpus had very little information about idioms.

However, we achieved outstanding results for the rest of segmentations. It is worth noting that every segmentation ambiguity was detected for such categories, and only two among 175 cases were wrongly solved, giving us 98.85% accuracy.

Moreover, we realized that most soft errors come from the fact that some idioms contain very common words. This means that the alternative branches where words are not joined have a very high probability according to the training model. For example, the preposition '*a*' (to, at, on) is one of the most common words in the model. It has a very high occurrence probability and it is also very common to find it inside idioms.

The real problem is that idioms themselves appear little in the training corpus. So the trained model will give more weight to the segmented branch over the joined branch when the word '*a*' appears in the idiom.

	Cases	Good choice	No option given	Total Accuracy	Real Accuracy
Soft errors	134	82	5	61.19%	63.56%
Hard errors	175	173	0	98.85%	98.85%

Table 2: *Test results for experiment E3 classified by kind of errors.*

This happends with several very common words as '*que*' (that, which, than), '*de*' (of, from), etc. A possible solution would be to upgrade the size of the training corpus.

However, almost these errors could still be solved with morphosyntactic information, leading us to think that it is possible to upgrade the accuracy of the system with some rules. This approach would be a less expensive solution than increasing the training corpus. Such rules may act in the lattice structure itself, pruning segmentation branches that prove impossible for the current context. From our point of view, a small set of rules could greatly improve the accuracy of the system for idioms and bring it near to 100% for other categories. In this case we would have a hybrid system with a very high degree of accuracy in the tokenization task.

7 Conclusions and future work

The tokenization task is usually simplified, leaving segmentation ambiguities to be solved in later steps of the NLP applications. In our case, we chose to accomplish segmentation tasks in the POS tagging phase, making it more complex, but the benefits will affect all successive applications.

In this paper, we have presented a practical application of the Viterbi-N algorithm for segmentation disambiguation and POS tagging. Segmentation ambiguities arise when one or several words can be segmented into linguistic tokens in more than one way. These are the cases of some contractions, verbal forms with enclitic pronouns, idioms, etc. The underlying idea for this combination of tasks is that POS categories provide a lot of information that can be used when choosing the correct alternative for an ambiguous segmentation. In the end, we have developed a POS tagger able not only to decide the tag to be assigned to every token, but also to choose the best sequence of tokens from a set of possible segmentation paths as well. Since the approach is purely stochastic, the technique could be easily exported to other languages.

Another advantage of the approach used, is that segmentation disambiguation could be considered a costless add-on for the POS tagging environment. If the training corpus is built carrying out the segmentations, they will be included in the learned model automatically.

The developed system was tested in the context of the Galician language, which has a very rich morphology, that is, the worst scenario for our system, and quite good results were achieved in the segmentation disambiguation task. We believe that they will be improved when the training corpus will be mature enough.

Indeed, another way to improve results is to use rules based on linguistic information which could prune some erroneous segmentation candidates. This would be particularly useful when the training corpus is of small size or low quality.

Acknowledgments

This paper arises out of a project developed at Centro Ramón Piñeiro para a Investigación en Humanidades and is partially supported by Ministerio de Educación y Ciencia (TIN2004-07246-C03-1), Xunta de Galicia (PGIDIT05PXIC30501PN) and "Galician Network for Language Processing and Information Retrieval" 2006-2009.

References

[1] http://corpus.cirp.es/corgaxml. *Reference Corpus from Present-day Galician Language (CORGA).*

[2] T. Brants. Cascaded markov models. *Proceedings of the Ninth Conference of the European chapter of the Association for Computational Linguistics (EACL-99)*, 1999.

[3] T. Brants. Tnt – a statistical part-of-speech tagger. *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000)*, 2000.

[4] D. Elworthy. Tagset design and inflected languages. *Proceedings of EACL SIGDAT workshop From Texts to Tags: Issues in Multilingual Language Analysis*, 1995.

[5] J. Graña. *Técnicas de Análisis Sintáctico Robusto para la Etiquetación del Lenguaje Natural*. Doctoral thesis, Universidad de La Coruña, Spain, 2000.

[6] J. Graña, M. Alonso, and M. Vilares. A common solution for tokenization and part-of speech tagging: One-pass viterbi algorithm vs. iterative approaches. *Petr Sojka, Ivan Kopecek and Karel Pala (eds.), Text, Speech and Dialogue, volume 2448 of Lecture Notes in Artificial Intelligence, pp. 3-10, Springer-Verlag*, 2002.

[7] J. Graña, F. M. Barcala, and J. Vilares. Formal methods of tokenization for part-of-speech tagging. *Computational Linguistics and Intelligent Text Processing, volume 2276 of Lecture Notes in Computer Science, pp. 240-249, Springer-Verlag*, 2002.

[8] J. L. A. Moreno, A. Álvarez Lugrís, and X. G. Guinovart. Aplicación do etiquetario morfosintáctico do sli ó corpus de traduccións tectra. *Viceversa: Revista Galega de Traducción, 7/8, pp. 189-212*, 2003.

[9] M. Vilares Ferro, A. Valderruten Vidal, J. Graña Gil, and M. Alonso Pardo. Une approche formelle pour la génération d'analyseurs de langages naturels. In P. Blache, editor, *Actes de la Seconde Conférence Annuelle sur le Traitement Automatique du Langage Naturel*, Marseille, France, June 1995.

[10] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory, vol. IT-13*, 1967.