Comparing Two Markov Methods for Part-of-Speech Tagging of Portuguese

Fábio N. Kepler and Marcelo Finger

Institute of Mathematics and Statistics, University of São Paulo (USP)
{kepler,mfinger}@ime.usp.br

Abstract. There is a wide variety of statistical methods applied to Part-of-Speech (PoS) tagging, that associate words in a text to their corresponding PoS. The majority of those methods analyse a fixed, small neighborhood of words imposing some form of Markov restriction. In this work we implement and compare a fixed length hidden Markov model (HMM) with a variable length Markov chain (VLMC); the latter is, in principle, capable of detecting long distance dependencies. We show that the VLMC model performs better in terms of accuracy and almost equally in terms of tagging time, also doing very well in training time. However, the VLMC method actually fails to capture really long distance dependencies, and we analyse the reasons for such behaviour.

1 Introduction

An empiric approach to Natural Language Processing (NLP) suggests that we can learn the complicated and extensive structure of language by searching common patterns in its use, identified via Probability Theory. This leads to probabilistic models of linguistic phenomena whose parameters are induced by the application of statistical and machine learning methods.

This work concentrates on *part-of-speech analysis*, that is, attributing to words in context their correspondent part-of-speech (PoS) tag. Several computational linguistic tasks benefit from such analysis, such as parsing, automatic translation, grammar correction and information extraction.

Many words have more than one possible PoS tag, which is disambiguated by the context in which these words occur. However, even with contextual information some words remain ambiguous.

Several computational approaches use statistical methods to model relations between words, tags and contexts, which are normally computed by employing some form of supervised learning from a manually tagged corpus. Other approaches exist in literature like neural networks and symbolic methods.

A traditionally used method is the Hidden Markov Model (HMM) [1], that tags a word generally by analysing the tags of the two previous words. This method works quite well, classifying correctly more than 93% of the words of a Portuguese corpus. This relatively great accuracy is due to the fact that most contextual dependencies is short distanced, so that just the two previous tags can determine the tag of a word. But considering that the tagging accuracy of several existing limited-context methods have practically converged to a value inferior to 100% (∼97%), an extra portion of efficiency could be reached developing methods that consider longer distance dependencies. However, due to data sparseness, contexts with size greater than two can hardly improve accuracy on these methods, since several contexts occur only once in the corpus, and therefore cannot capture any dependency. This can be seen even with contexts of size two, as shown in the results (Section 4). Moreover, the time necessary to train a HMM model, that is, to count and to calculate the probabilities of tags in context, is usually too large, with an exponential growth on the number of words considered in the context. The other method we study is Variable Length Markov Chains (VLMC), and we use it to try to capture *variable* long distance dependecies, thus overcoming these problems and reaching that extra efficiency.

This work presents a comparison between two methods applied to PoS tagging of Portuguese: the classically used HMM method, and the not so known VLMC method. It also presents a study about the treatment of short, average, and long distance dependencies in the Brazilian Portuguese when using VLMC. We obtained accuracy results very near the best ones reported for Portuguese, and also with training and execution times well below the existing ones in literature.

The article is organized as the following. Section 2 briefly presents the application of VLMC to the PoS tagging problem. Section 3 describes the taggers' implementation. The obtained results are analysed in Section 4, and a comparison with similar results in literature is shown in Section 5. Conclusions about the application of the methods are then discussed in Section 6.

2 Variable Length Markov Chains

The idea is to allow the memory of a Markov chain to have variable length, depending on the observed past values. We will explain the concept of Variable Length Markov Chains considering the PoS tagging context. For a formal description see [2,3].

Consider a Markov chain with a finite, large order k. Let l_i be a tag, and $l_{i-k,i-1}$ be the tags preceding l_i. The idea of a variable length memory can be seen as a cut of irrelevant states from the $l_{i-k,i-1}$ history. We call the set of these states the *context* of l_i.

Given a tag l_i, its context $l_{i-h,i-1}$, $h \leq k$, is given by the *context function* $c(l_{i-k,i-1})$. The states that determine the probabilities of the VLMC are given by the values of the context functions of the tags. These states are represented as a tree.

A *context tree* is a tree with a root node in the top, from which ramifications go down, such that each internal node has at most $|\mathcal{L}|$ sons, where $\mathcal{L}$ is the tagset. Each value of a context function $c(\cdot)$ is represented as a branch of such tree. For example, the context given by $c(l_{i-k,i-1})$ is represented as a branch whose sub-branch in the top is determined by l_{i-1}, the next sub-branch by l_{i-2}, and so on, until the leaf, determined by l_{i-h}.

The parameters of a VLMC are the underlying functions $c(\cdot)$ and their probabilities. To obtain these parameters we use a version of the context algorithm of Rissanen [4]. First, it builds a big context tree, using a training corpus. For a tag l_i, its maximal history $l_{i-k,i-1}$ is placed as a branch in the tree. Then, the algorithm uses a pruning function considering a local decision criterion. This pruning cuts off the irrelevant states from the tags' histories. For each leaf u of the context tree, and branch v that goes from the root to the upper node of u, u is pruned from the tree if

$$\varDelta_{vu} = \sum_{l \in \mathcal{L}} P(l|vu) \log \left(\frac{P(l|vu)}{P(l|v)} \right) C(vu) < K, \tag{1}$$

where $C(vu)$ is the number of occurrences of the sequence vu in the training corpus, and K is a threshold value, called the *cut value* of the context tree,

If the probability of a tag does not change much between considering the entire branch together with the leaf (all past history) and considering only the branch (the history without the oldest tag), then the leaf does not need to be considered, and can be removed from the tree.

3 Implementation

We have implemented two PoS taggers, one based on VLMC and one based on HMM.

The HMM tagger uses second order Markov models — that is, the tag of a word depends on the word itself and on the two previous tags — where the states represent the tags and the observations represent the words. The transition probabilities depend on the states, in this case pairs of tags, and the outcome probabilities depend on the destination state. Formally, we wish to find the best sequence of tags $l_1 \ldots l_T$ for a given sequence of words $w_1 \ldots w_T$ of size T, that is,

$$\arg \max_{l_1 \ldots l_T} \left[\prod_{i=1}^{T} P(l_i | l_{i-1}, l_{i-2}) P(w_i | l_i) \right] \tag{2}$$

where l_0 and l_{-1} are sentence starting markers.

The probabilities are estimated from a tagged corpus. We use maximum likelihood probabilities $\hat{P}$, which are derived from the relative frequencies of words and sequences of tags. For example, the probability of a tag l_3 given the sequence l_1, l_2 is equal to the number of times the three tags occur in sequence (l_1, l_2, l_3) in the corpus divided by the number of times only the two previous tags occur in sequence (l_1, l_2). That is, $\hat{P}(l_3|l_1, l_2) = \frac{C(l_1,l_2,l_3)}{C(l_1,l_2)}$.

In the VLMC tagger the sequences of tags obtained from the training corpus are used to feed the context tree, which after pruned defines the context functions. This way, for a given sequence of tags $l_{i-k,i-1}$, the probability that the next tag is l_i is given by $P(l_i | c(l_{i-k,i-1}))$.

We decided to use the value of K given by an equation defined empirically:

$$K = \frac{\log(n)}{\log(|\mathcal{L}|)} \cdot \frac{n}{|\mathcal{S}|}, \tag{3}$$

where $|\mathcal{L}|$ is the size of the tagset, $|\mathcal{S}|$ the number of sentences, and n the number of words of the training corpus. For the whole training corpus ($775,602$ words), K was equal to 54.6615. The results showed in Section 4 were obtained with this equation.

3.1 Treating Unknown Words

When a text is tagged, words new to the training corpus are found. Thus, in order to tag them more correctly as possible, a special treatment is needed.

We used two complementary standard methods. The first consists in restricting the possible tags for an unknown word, eliminating closed tags. *Closed tags*, like articles and conjunctions, are those tags that are only assigned to a limited number of different words. Yet other tags like verbs and nouns can be assigned to a great number of distinct words, and therefore are called *open tags*. Though this distinction can be easily made by hand it is not passed to the taggers. Instead, they are given a threshold that is used to build these two sets of tags after training: if a tag occurs with too many different words, it is an open tag; otherwise, it is a closed tag.

In the second method the word's morphology is analyzed. We use a simplificated suffix analysis[1]: a tree is built with the suffixes of all words from the training corpus that have an open tag; next, this tree is normalized, creating probability distributions; then, for an unknown word, a search for its greatest suffix existing in the tree is made, returning the probabilities of the possible tags. We consider as the suffix of a word used to build the tree the last half of the word. Moreover, we check to see if the first letter of a word is uppercase and if so, if this word is not sentence starting, we assign to it a higher probability of being a proper noun. Note that no external dictionary is used.

4 Tests and Results

Both taggers were implemented using `C++` and `STL` (*Standard Template Library*), and were compiled with `g++` version 3.3.4. The tests were made on a machine equipped with one Intel Pentium 4 processor of 3 GHz, and 1 GB of RAM.

The taggers were trained and tested with the *Tycho Brahe* corpus [5], which uses a set of 383 tags and contains various texts from historical Portuguese manually tagged, in a total of $1,035,593$ words. These words were splitted into a training corpus containing $775,602$ words, and a testing corpus containing $259,991$ words.

[1] The term suffix we use means "final sequence of characters of a word", what is not necessarily a linguistically significant suffix.

We executed sets of tests varying the size of the training corpus, choosing $5\%, 10\%, \ldots, 95\%, 100\%$ of its sentences and executing 10 times with each one of these sizes (randomizing the sentences each time), but always using the testing corpus without modifications.

Figure 1 shows accuracy[2] results for both taggers. It can be seen that the

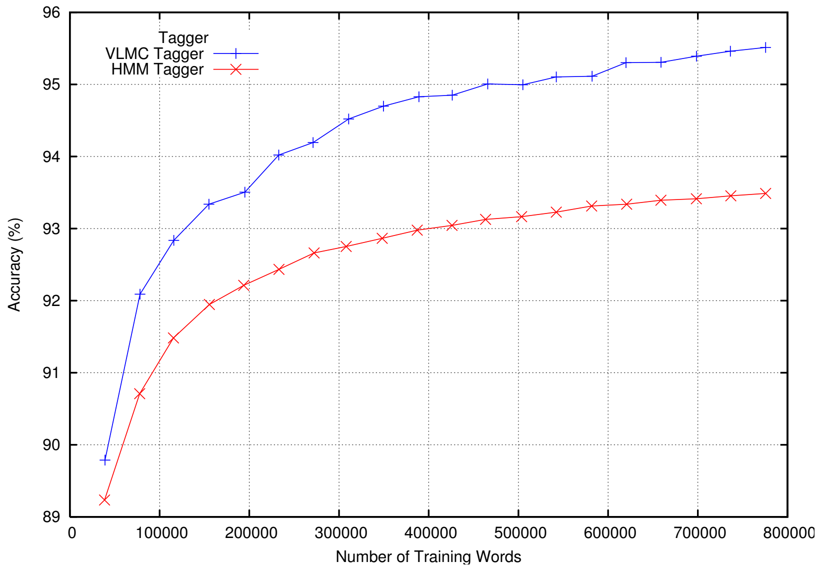

Fig. 1. HMM tagger's and VLMC tagger's accuracy.

VLMC tagger has consistently a greater accuracy than the HMM tagger's one; for the whole corpus, 95.51% over 93.48%. The VLMC tagger's curve grows quicker than the HMM tagger's one. Based on this information we can say that, even increasing the number of words in the training corpus, the HMM tagger will not reach the accuracy of the VLMC tagger.

In the graph of the Figure 2 three curves are shown, representing the total accuracy of the VLMC tagger, its accuracy for known words, and its accuracy for unknown words. When using the entire training corpus, 69.53% of the unknown words are correctly tagged. With only 5% of the training corpus, the average accuracy for unknown words was only 63.97%, a difference of more than 5%. Considering the fact that the number of unknown words is greater when using a smaller training corpus, this difference represents approximately 4% of the testing corpus, which means more than $10,650$ tagging mistakes. With respect to known words, the average accuracy with 5% of the training corpus was 94.20%,

[2] By accuracy we mean the proportion of words from the testing corpus to which the tagger assigns the correct tag.

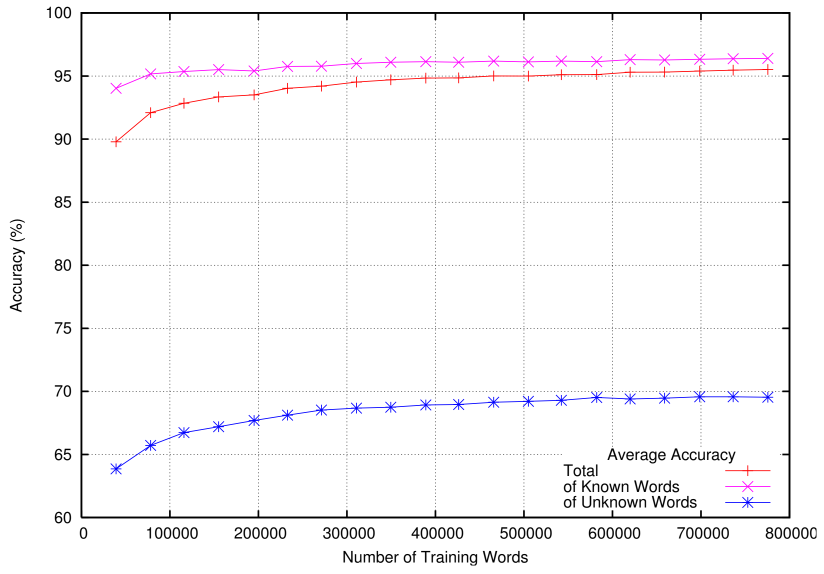

Fig. 2. VLMC tagger's accuracy for known and unknown words.

and with the entire corpus 96.39%. This difference is approximately 3,872 words, about 1.49% of the testing corpus.

In the HMM tagger, the percentage of unknown words correctly tagged varied from 62.37% with 5% of the training corpus, to 68.81% with the entire corpus; a difference of 6.5%, which in the VLMC tagger was of 5%. This can be explained by the fact that the HMM tagger considers very small contexts, and therefore it ends out leaving the choice of the best tag for an unknown word only to the suffix tree. This is why an increase in the number of training words improves the accuracy for the unknown words.

The fact that the HMM tagger considers very small context also explains the interesting result obtained for the known words. The variation in the accuracy according to the size of the training corpus was small: 93.62% with 5% of it, and 94.32% with the entire corpus. A difference of only 0.7%, against 2.19% from the VLMC tagger. That is, because the HMM tagger considers small contexts, tagging the known words is little influenced by the tagging of the unknown words. This shows that, at the same time that the VLMC tagger treats better unknown words, since it uses bigger contexts as aid, it is also more sensible to eventual tagging mistakes, and may wrongly tag a word by influence of the context.

Figure 3 shows the curves of the average training and tagging times taken by the taggers with respect to the number of words used to train them. Both training and tagging times for the HMM tagger are slightly smaller than the ones for the VLMC tagger. But in fact, considering the total execution time of a tagger as the sum of its training and tagging times (and some constant overhead

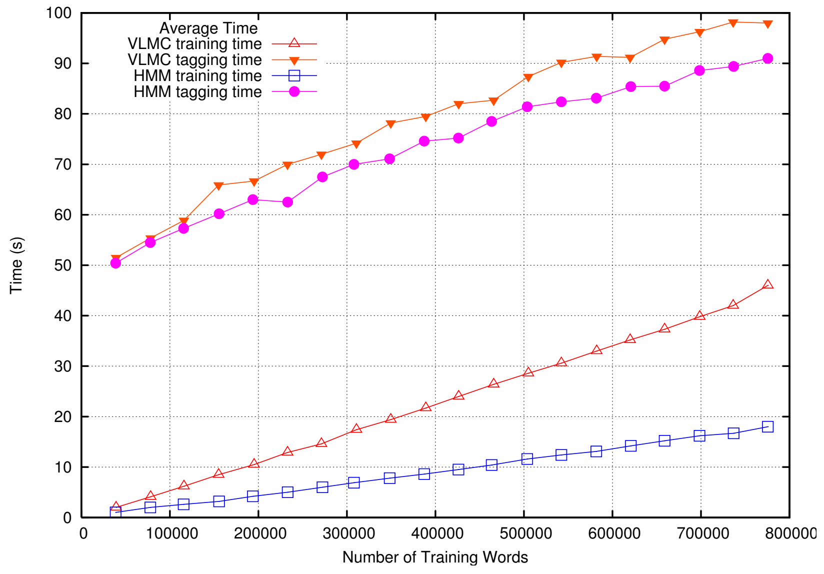

Fig. 3. HMM tagger's and VLMC tagger's average training and tagging times.

time for system operations), the curve of the HMM tagger's execution time has a correlation of 0.9956, and the VLMC tagger's one of 0.9969. This means that both taggers have linear complexity over time, and so both are equally efficient with respect to execution time.

However, note that the HMM tagger's training time is relatively low. The explanation is that we did not make any training iteration for it, as normally is done. On the contrary, we just learned the probabilities through the training corpus. This was done because the time needed for the training process was too great, around 4622 seconds (37 times greater than without training), and the accuracy did not improve as much as expected. Moreover, works like the ones of Church [6] and DeRose [7] also collect statistical information from a tagged corpus instead of using the process of training for HMM.

Figure 4 shows the number of states of two tags built by the HMM tagger during training. Figure 5 shows the growth of the number of nodes in the VLMC tagger's context tree. Comparing both figures, it can be seen that the number of states in the HMM tagger is more than three times larger than the number of nodes in the VLMC tagger, showing that many sequences of two tags from the training corpus are consequence of sparse data, and do not add much knowledge relevant for the tagging.

Other results were also generated, but these are omitted due to space restrictions. The interested reader can consult [reference withold].

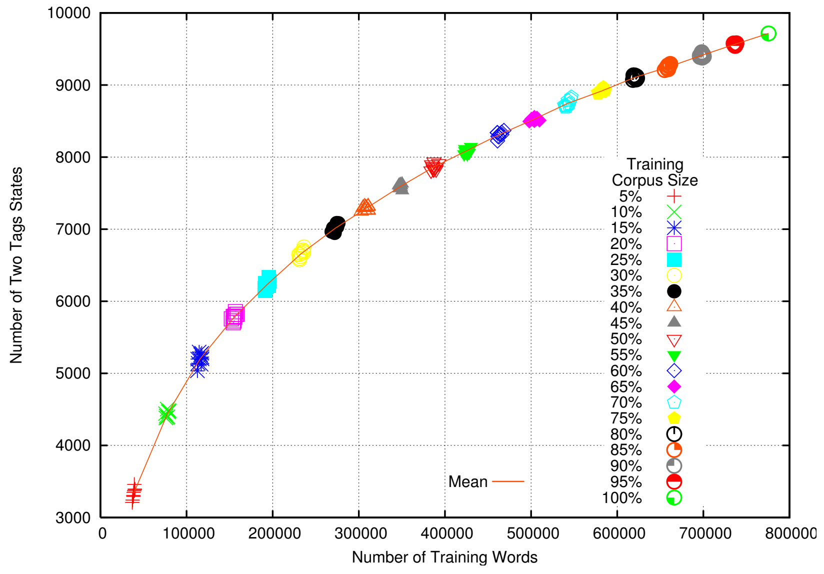

Fig. 4. Number of states of two tags obtained by the HMM tagger with respect to the number of words in training.

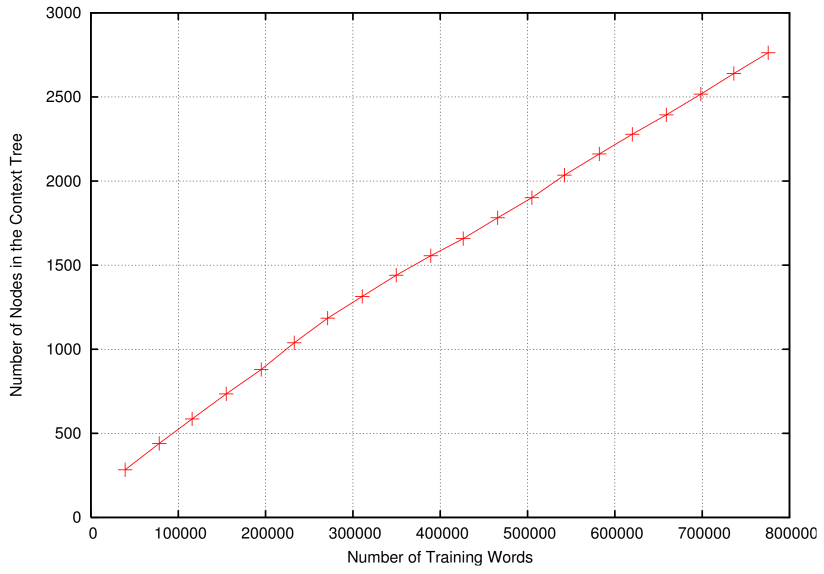

Fig. 5. Growth of the number of nodes in the VLMC tagger's context tree.

5 Related Works

A currently well known tagger is the one of Brill [8], which is based on transformation rules and achieves around 95.4% of accuracy in the Wall Street Journal (WSJ) [9] English corpus. For the Portuguese, Chacur and Finger [10] proposed and implemented a variant of the Brill's method, and obtained reasonable results. After that Finger [11] used some optimization techniques and obtained better results, around 95.43% of accuracy on the Tycho Brahe [5] corpus.

Among the taggers that use statistical models Ratnaparkhi [12] implements one based on Maximum Entropy, which obtains 96.6% of accuracy on the WSJ corpus. Brants [13] implements a tagger based on HMM that achieves an accuracy of 96.7%. Recently, Toutanova [14] showed a tagger based on Ciclic Dependendy Network, which obtains 97.24% of accuracy, claiming this is the state of the art for English. For Portuguese, Aires [15] adapts various English taggers and shows their results for Portuguese. The best one achieves 90.25% of accuracy, and is obtained by the adaptation of the Ratnaparkhi's Maximum Entropy tagger.

6 Conclusions

We built two PoS taggers for Portuguese, one based on the traditionally used HMM, and one based on VLMC, a recent theoretic statistical model.

With an accuracy of 95.51% with the VLMC tagger we obtained a result very close to the best ones reported for Portuguese [10,15,11]. Also, the time spent in training with more than $775,000$ words and in tagging almost $260,000$ $(1,035,593$ in total) is very fast, though we cannot compare it to other taggers in literature since this result is normally not presented (even so, we report that the VLMC tagger tooks only 157 seconds to train and tag with the million words stated above, running on cheap machine).

When instructing the VLMC tagger to consider longer contexts, it was not able to detect many long distance dependencies. Moreover, instructing it to consider not so long contexts have improved the performance (in terms of accuracy). So we conclude that, when having less long contexts available, the tagger chooses short and recent contexts, what improves the performance and shows that there are long contexts that decay it.

Though Variable Length Markov Chains do not capture very long contexts, they perform consistently better for part-of-speech analysis than the classicaly applied theory of fixed order Markov Chains. We give results that allow us to observe limitations and advantages of the application of statistical models based on VLMC: they learn various short and average distance fixed contexts ($d \leq 6$), but they do not have generalizating capacity to learn linguistic phenomena occuring in variable contexts and of longer distance. Future research in statistical linguistics regarding long range dependencies should concentrate in other ways of solving this limitation.

References

1. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. In: Proceedings of the IEEE. Volume 77 of no. 2. (1989) 257–285
2. Bühlmann, P., Wyner, A.J.: Variable length markov chains. Annals of Statistics **27**(2) (1999) 480–513
3. Mächler, M., Bühlmann, P.: Variable length markov chains: Methodology, computing and software. Research Report 104, Eidgenossische Technische Hochschule (ETH), CH-8091 Zürich, Switzerland (2002) Seminar fur Statistik.
4. Rissanen, J.: A universal data compression system. IEEE Trans. Inform. Theory **IT-29** (1983) 656 – 664
5. IEL-UNICAMP and IME-USP: Corpus Anotado do Português Histórico Tycho Brahe. (2005) Acessado em 2005.
6. Church, K.W.: A stochastic parts program and noun phrase parser for unrestricted text. In: Proceedings of the second conference on Applied natural language processing, Austin, Texas, Association for Computational Linguistics (1988) 136–143
7. DeRose, S.J.: Grammatical category disambiguation by statistical optimization. Computational Linguistics **14** (1988) 31–39
8. Brill, E.: Unsupervised learning of disambiguation rules for part of speech tagging. In Yarovsky, D., Church, K., eds.: Proceedings of the Third Workshop on Very Large Corpora, Somerset, New Jersey, Association for Computational Linguistics (1995) 1–13
9. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics **19**(2) (1994) 313–330
10. Alves, C.D.C., Finger, M.: Etiquetagem do português clássico baseada em córpora. In: Proceedings of IV Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR99), Évora, Portugal (1999) 21–22
11. Finger, M.: Técnicas de otimização da precisão empregadas no etiquetador Tycho Brahe. In: Proceedings of V Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR2000), Atibaia, Brazil (2000) 19–22
12. Ratnaparkhi, A.: A maximum entropy model for part-of-speech tagging. In: Proceedings of the Empirical Methods in Natural Language Processing Conference, University of Pennsylvania (1996)
13. Brants, T.: Tnt – a statistical part-of-speech tagger. In: Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP-2000), Seattle, WA (2000)
14. Toutanova, K., Klein, D., Manning, C.D., Singer, Y.: Feature-rich part-of-speech tagging with a cyclic dependency network. In: Proceedings of HLT-NAACL 2003. (2003) 252–259
15. Aires, R.V.X.: Implementação, adaptação, combinação e avaliação de etiquetadores para o português do brasil. Dissertação de mestrado, Instituto de Ciências Matemáticas e Computação, Universidade de São Paulo - Campus São Carlos (2000)