Mac-Morpho Revisited: Towards Robust Part-of-Speech Tagging

Erick Rocha Fonseca[1] **, João Luís G. Rosa**[1]

[1]Instituto de Ciências Matemáticas e de Computação
Universidade de São Paulo (USP) – São Carlos, SP – Brazil

{erickrf,joaoluis}@icmc.usp.br

Abstract. *We present a revision of Mac-Morpho, the biggest corpus of Portuguese text containing manually annotated POS tags. Many errors were corrected, yielding a much more reliable resource. We also trained a neural network based classifier for the POS tagging task, following an architecture that achieves state-of-the-art results in English. Our tagger maps each word to a real valued vector and uses it as input, thus dealing with abstract features. These vectors are induced by distributional semantics techniques, and provide the tagger with information for achieving 96.48% accuracy.*

1. Introduction

Part-of-Speech (POS) tagging is an important Natural Language Processing (NLP) task, serving as a first step for many applications. While a rule based approach for an automatic tagger is possible [Bick 2000], most of the work addressing this task is based on machine learning techniques, as usual in NLP, since they require significantly less labor. In order to do so, one must consider two points: the training data and the learning algorithm for the tagger.

The training data, in the form of an annotated corpus, should be large enough to allow the tagger to generalize what it learns to unseen sentences. The two most widespread corpora with annotated POS tags in Portuguese are Mac-Morpho [Aluísio et al. 2003], with around one million words, and Bosque [Afonso et al. 2002], with around 185 thousand.

Both corpora cannot be combined to provide a larger resource, since each one defines a different tagset. For example, while Bosque has different tags for verbs in the infinitive, gerund, participle and inflected forms, Mac-Morpho only distinguishes participles from the other three. On the other hand, Mac-Morpho has different tags for auxiliary and main verbs, which Bosque does not.

The quality of the data is also important: too much noise (such as wrongly assigned tags) may affect the learning process. As the annotation is done by humans, mistakes are often introduced, and thus a rigourous checking procedure must be carried out.

As for learning algorithms, there are many that have been proposed and succesfully applied in this task, usually capable of being employed in different languages. In Portuguese, experiments reported in the literature include Transformation Based Learning [dos Santos et al. 2008], Hidden Markov Models (HMM) and Variable Length Markov Chain (VLMC) [Kepler and Finger 2006], HMM with a character language model [Maia and Xexéo 2011], among others.

Proceedings of the 9th Brazilian Symposium in Information and Human Language Technology, pages 98–107,
Fortaleza, CE, Brazil, October 21–23, 2013.

The work reported in this paper aimed at both these points. First, we performed a thorough error verification and cleaning process on the Mac-Morpho corpus, which we used as training data for our models. We report the problems found and make our revised version publicly available. Also, we joined contracted forms (such as *do* for *de* + *o*), which appear splitted in the corpus, in order to reflect a real world scenario. Most works reported in the literature don't mention this step.

Second, we trained a POS tagger similar to the work found in [Collobert et al. 2011], based on multilayer perceptron neural networks and vector space models, and that achieves state-of-the-art performance in English. Our resulting tagger is also available online. Besides our model, we performed experiments with the OpenNLP POS tagger[1] for comparison.

2. Corpus Treatment

We chose to use the Mac-Morpho corpus because it is the biggest one available with POS tags in Portuguese. Mac-Morpho is composed of 109 files with texts from the Brazilian newspaper Folha de São Paulo, and is divided in 10 sections, each having a given topic (such as agriculture, politics, sports, etc.).

We identified some common errors in the corpus, such as missing words and repeated sentences. One example of sentence with a missing word is *A degradação das terras pelo mau uso dos solos avança no.* ("Land degradation due to inappropriate soil usage advances in the."). These are prejudicial for machine learning, as a classifier will get examples of impossible sequences (caused by missing words) or may be biased towards repeated examples.

We developed simple heuristic rules to identify sentences with missing words, listed below. If any of the rules triggered for a given sentence, it was removed from the corpus, as there was no way to determine the missing word.

1. The sentence ends with a conjunction, preposition or article (including contractions).
2. An article appears before a verb, except for the case of *ao* (contraction between *a* and *o*), which is commonly used before infinitives (e.g., *ao ver*, "upon seeing").
3. An article appears before a preposition, conjunction or punctuation sign.
4. A comma appears before a period, colon, semicolon, exclamation or question mark.
5. The same punctuation sign appears twice in a row, except for exclamation and question marks.

We are aware that these rules might trigger for false positives, but they are very rare in comparison with the actual mistakes. Using them, we removed 1,232 sentences. We also checked repeated sentences, discarding 2,947. Adding up, we removed 4,179 sentences from a total of 54,169 in the corpus (7.7%).

Another problem was that half of the files in the corpus don't indicate sentence boundaries; instead, these files have one token per line. Examining sentences separately is important since most taggers require that they be provided one at a time, so we used the Punkt sentence tokenizer from NLTK[2] [Bird et al. 2009] to split the text into sentences.

[1] Available at `http://opennlp.apache.org/`

[2] Available at `http://www.nltk.org`

This tool works by classifying periods as sentence delimiters or not (as in the case of abbreviations), and also tries to correctly handle quotation marks and parentheses in the end of a sentence.

The other half of the files contained XML-like tags to indicate paragraph and sentence boundaries. Due to tag mismatches we couldn't feasibly correct, we had to put off six of these files. With the remaining, we could easily split the sentences.

After searching for characters not used in Portuguese, we found a few typographical errors such as the use of a diaresis instead of the proper accent, e.g., *conträrio* instead of *contrário*. We also found mistyped tags (identifiable as tags not in the defined tagset) and a spurious $ symbol before some punctuation signs. All of these could be manually corrected.

As for tokenization, important decisions in Portuguese concern how to treat preposition contractions and clitic pronouns. Mac-Morpho presents all components of these structures separately, but indicates that they were originally joined. In our experiments, we chose to redo all contractions appearing in the text, aiming at simulating a real world scenario. The tags for the contractions were obtained as a concatenation of the component tags and a plus sign, e.g. `PREP+ART` for the contraction of a preposition (tagged as `PREP`) and an article (`ART`).

However, we chose to keep clitic pronouns separated from verbs. This decision was motivated by two factors: first, it is trivial to identify clitics by simple pattern matching. Second, if we consider other NLP tasks such as semantic role labeling, separating pronouns from verbs is much more important than splitting preposition contractions.

We set aside every tenth sentence in the corpus for testing, leaving the rest for training. This resulted in 4,999 and 44,991 sentences, respectively. Table 1 shows the number of occurrences for each of the 30 tags[3]. We can see that some tags, especially those involving contractions, are very rare.

Tag	Train	Test	Total	Tag	Train	Test	Total
ADJ	39,009	4,264	43,273	ADV	22,306	2,509	24,815
ADV-KS	289	31	320	ADV-KS-REL	648	69	717
ART	61,905	6,804	68,709	CUR	2,235	239	2,474
IN	267	17	284	KC	21,034	2,333	23,367
KS	10,816	1,275	12,091	N	180,835	20,181	201,016
NPROP	82,541	9,237	91,778	NUM	14,506	1,692	16,198
PCP	17,623	1,927	19,550	PDEN	5,120	546	5,666
PREP	82,103	9,296	91,399	PREP+ADV	72	13	85
PREP+ART	52,579	5,680	58,259	PREP+PROADJ	1,549	166	1,715
PREP+PRO-KS	28	4	32	PREP+PRO-KS-REL	168	19	187
PREP+PROPESS	479	54	533	PREP+PROSUB	638	72	710
PROADJ	13,767	1,647	15,414	PRO-KS	1,594	165	1,759
PRO-KS-REL	8,298	863	9,161	PROPESS	10,308	1,228	11,536
PROSUB	5,710	672	6,382	PU	124,881	14,025	138,906
V	75,686	8,415	84,101	VAUX	13,969	1,552	15,521

Table 1. Distribution of tags in the corpus

[3]The description of the tags used in Mac-Morpho can be found at `http://www.nilc.icmc.usp.br/lacioweb/english/manuais.htm`

3. The Tagger

We implemented the model presented in [Collobert et al. 2011] for training a POS tagger. It receives a window of tokens as input and maps them to feature vectors, which are then concatenated and fed to a multilayer perceptron neural network. Figure 1 shows a window of three tokens being converted into vectors. There must be one neuron in the network input layer for each of these values.

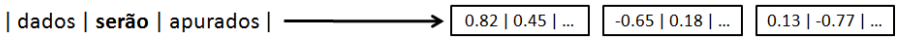

Figure 1. Example of a window of size 3 being converted into feature vectors

3.1. Word Representations

In this architecture, each word has a corresponding real valued vector[4]. The representations used by [Collobert et al. 2011] were obtained in semi-supervised fashion through a neural language model, which was trained to distinguish positive examples of word sequences (extracted from a corpus) from negative ones (random perturbations of the positive example).

In their training process, the authors sampled word sequences from a huge corpus and corrupted them by randomly replacing the middle word. Then, the neural model was fed both sequences and had to output a score for the original one higher by a given margin than for the corrupted one. The corrections in the network parameters were backpropagated to the word representations. As a result, words with similar meaning and usage had vectors with a small euclidian distance.

Using such representations brings a couple of advantages: the automatic classifier can easily detect words that should be treated similarly, and words not seen in the training data for a tagging task are not completely unknown, as long as they have a feature vector. Thus, out-of-vocabulary (OOV) impact is expected to be lesser.

The unsupervised training for generating word representations, however, is extremely slow: the authors report weeks of training time. Motivated by the observations from [Turian et al. 2010] that representations obtained in different ways may be used by a classifier to obtain good results in NLP tasks, we turned our attention to methods based on distributional semantics [Turney and Pantel 2010], which are much faster. In [Huang and Yates 2009], word representations generated with such methods are also employed for POS tagging in English; however, the system architecture in that work is very different from the one explored here.

We used the software package Semantic Vectors[5] [Widdows and Ferraro 2008] to induce representations from a collection of texts composed of the Portuguese Wikipedia[6] and the PLN-BR corpus [Bruckschen et al. 2008]. We used the method known as Hyperspace Analogue to Language (HAL) [Lund and Burgess 1996], which consists in creating a table counting the occurrences of each word in the vocabulary next to each other word.

[4]Actually, not only words, but rather all types, including punctuation, numbers, etc. can have a feature vector. We use the term *word* here because it is commonly found in the literature.

[5]Available at `https://code.google.com/p/semanticvectors/`

[6]Available at `http://dumps.wikimedia.org/ptwiki`

We induced vectors for all 89,075 word types that occurred at least 40 times in the corpus. Other words are mapped to a special vector generated randomly. Two other vectors were also generated randomly for the padding before after the limits of a sentence. We experimented with vectors having from 50 to 300 dimensions, and after examining the results, we concluded that the overall quality was about the same for all numbers of dimensions. We chose to keep the 50-dimension vectors for computational efficiency.

Besides encoding word types, feature vectors can also represent discrete attributes such as presence of capitalization. To this end, each possible value of the attribute must have a corresponding vector; in the case of capitalization, values could be: all lowercase letters, initial uppercase letter, other combinations and a N/A value for punctuation and numbers. Thus, when the network is given a token, its type vector is concatenated with all other feature vectors. Figure 2 exemplifies this process.

Type	**Vector**		
não	0,97	-0,34	0,16
sei	-0,81	0,09	-0,21
.	0,49	0,82	0,63
⋮		⋮	

Value	**Vector**	
All lowercase	0,04	0,72
Uppercase initial	-0,59	0,18
Other case combinations	-0,12	-0,65
N/A	0,94	0,51

Token	**Resulting Vector**				
Não	0,97	-0,34	0,16	-0,59	0,18
sei	-0,81	0,09	-0,21	0,04	0,72
.	0,49	0,82	0,63	0,94	0,51

Figure 2. Representations including a discrete attribute

3.2. Simple Word Window Approach

In the most basic setup, the simple word window approach, the network has one hidden layer and performs usual operations (weighted sum followed by a sigmoid function). It outputs a score f_j for the token in the middle of the input window having each tag j; so, in order to tag all tokens in a given sentence, the network must examine each window at a time. In the case of tokens near the beginning or the end of a sentence, the input window is complemented with pseudo-tokens serving as padding. These pseudo-tokens also have their own corresponding feature vectors. Figure 3 shows an example of all possible windows obtained from a sentence.

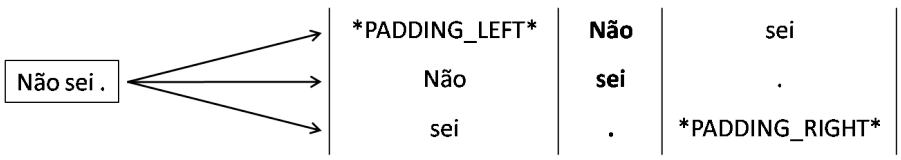

Figure 3. Windows of size 3 produced from a sentence.

The neural network is trained via backpropation, doing a gradient ascent aimed at maximizing the log likelihood over training data. Due to the paucity of space, we refer the reader to [Collobert et al. 2011] for a complete demonstration of the differentiation of the system output. Gradients are backpropagated until the input layer, so word representations

can be adjusted in the same way as the network connections. The corrections take place after running the network for each window.

3.3. Sentence Approach

The network score may also be combined with tag transition scores, encoding knowledge such as "an article is very likely to be followed by a noun", and thus working similarly to a Hidden Markov Model. These scores are encoded in a transition matrix A, which contains in each cell $A_{i,j}$ the score for a token tagged with i being followed by another one tagged with j. Thus, denoting the neural network parameters with θ and the network score for tag j at the t-th token with $f_{t,j}$, the score for a given sentence x of size T having the tag sequence y is:

$$s(x, y, \theta) = \sum_{t=1}^{T} f_{t,y_t} + A_{y_{t-1},y_t} \tag{1}$$

Note that we need a score $A_{0,j}$ for starting the sentence with tag j. This is actually a generalization of the previous setup: we can think of the simple word window approach as having a transition matrix where all cell values are set to zero.

Instead of assigning the tag with the highest score for each token right after examining its window, this setup stores all tag scores for each token and then searches for the tag sequence with the highest score using the Viterbi algorithm. Because of it, when training a network with this approach, the gradients are only calculated after tagging a whole sentence.

The transition matrix is also updated via gradient ascent, increasing values for transitions found in the sentence at the expense of the scores of unseen transitions. The computation of the gradients in the sentence approach involves both the network output and the transition values, and is much more complex than for the simple window approach.

4. Experimental Results

4.1. Tagging Accuracy

We evaluated our models on our revised version of the Mac-Morpho corpus. As additional attributes, we employed the presence of capitalization (as shown in Section 3.1) and word endings. Feature vectors for both attributes have 5 dimensions.

In order to determine which endings should have their own vectors, we examined all word types occurring in the training set and picked all endings of size 3 that occurred in at least 20 types. This yielded a total of 389 endings, and a vector was randomly generated for each of them. All words with size 3 or less are mapped to another vector, and those not ending in any of the listed patterns are mapped to another one. For simplicity, we call any word ending a *suffix*, even if it is not linguistically accurate.

We experimented with the following setups:

- Simple window approach with capitalization. (WC)
- Simple window approach with capitalization and suffixes. (WCS)

- Sentence approach with capitalization. (SC)
- Sentence approach with capitalization, suffixes and randomly initialized word representations. (SCSR)
- Sentence approach with capitalization and suffixes. (SCS)

In all setups, we employed a word window of size 5 and 100 neurons in the hidden layer, except for the SCSR, which had 150 neurons. These values were the smaller ones that yielded good performance overall. Training neural networks with such a high number of connections[7] is a very delicate process, as they can easily diverge if the learning rate is too high. On the other hand, if the rate is too low, the learning process can take too much time or even get stuck in a local minimum.

We found a good balance by starting training for two epochs with the rate set to 10^{-3}, then 200 epochs with 10^{-4}, and finally a few more with 10^{-5}. Training further with lower rates did not result in significant improvements. Still, training times were very long, usually around 20 hours. Training is clearly the main drawback of this method.

It is not possible to compare our results directly with others reported in the literature, as we use a different tagset than the default from Mac-Morpho (obtained after collapsing preposition contractions). In order to make comparisons, we trained a Maximum Entropy model from Apache OpenNLP, using its default parameters. Results are reported in Table 2, considering accuracy over all words in the test set and only in those that did not appear in the training set (OOV).

Model	Overall accuracy	OOV accuracy
WC	95.28%	82.84%
WCS	96.01%	87.92%
SCSR	91.38%	85.52%
SC	96.22%	84.31%
SCS	**96.48%**	87.34%
MaxEnt	95.92%	**91.99%**

Table 2. Experimental results

We can see that most variants of our model performed very well on the testing data, except for SCSR, as expected. This confirms the representative power of pre-initialized feature vectors, even using far simpler techniques than [Collobert et al. 2011]. We also see that our configurations using the simple window approach and/or without suffix features achieve good results, while the more complex ones have a small advantage.

When it comes to OOV accuracy, the importance of recognizing suffixes is greater, as expected. Despite achieving the second best results overall, the SC model performed poorly with OOV words, probably due to its lack of suffix knowledge. In fact, even the SCSR setup performed better in this regard than the ones that used HAL representations but no suffix features.

The MaxEnt classifier outperformed all our models in OOV accuracy. It also observes word endings, and the combination of features it uses seems to be more efficient for unknown words. This denies our *a priori* belief that our word representations would be

[7] Five tokens with 55 or 60 features each in the input yield 275 – 300 input neurons, meaning 27,500 to 45,000 connections from the input layer to the hidden one

Reference Word	Similar Words
ele	esta, também, essa, eles, ela
às	jon, joey, visivelmente, jorginho, galina
ontem	anteontem, carybé, colbert, extra-oficialmente, larson
reclamação	engrenagem, simbologia, reação, corrente, experiência
será	detectou, evitou, cumpre, seria, traz

Table 3. Comparison with word representations produced by HAL

Reference Word	Similar Words
ele	eu, nós, eles, ela, você
às	zenon, à, juno, pubs, aos
ontem	amanhã, anteontem, agora, aí, aqui
reclamação	estrutura, semente, simbologia, reação, experiência
será	demonstra, cumpre, exigia, foi, seria

Table 4. Comparison with word representations after training the SCS model

helpful for OOV classifying. Still, it would be worth investigating if word representations produced by a neural language model would lead to better performance.

4.2. Effect on Representations

We also examined the effect that training a model for POS tagging had on the word representations. We picked some common words from Mac-Morpho and searched for the ones most similar to them, according to the cosine of their vectors. Table 3 shows the results with the initial HAL vectors, and Table 4 with the vectors after training the SCS model.

In general, we found that representations for nouns and verbs didn't improve much, as shown for *reclamação* and *será*. HAL vectors could be enhanced by finer semantic knowledge, but the adjustments made were only related to POS tags.

For other word classes, it seems that better representations have been achieved. The vector for *ele* became closer to the ones for other personal pronouns, and the one for *ontem* to those of other time and space adverbs.

5. Conclusions

In this contribution, we presented a revised version of the Mac-Morpho corpus, with many mistakes corrected. We addressed the POS tagging task trying to simulate a real world scenario, and dealt with preposition contractions as single words. This is contrary to most works in Portuguese POS tagging, which do not care for this detail. We believe that this allows for a more robust tagger, capable of working with texts without any preprocessing.

We also experimented with an algorithm that has been shown to provide state-of-the-art performance for English POS tagging, and achieved good results in Portuguese. We found that it works well when given word vectors induced by a distributional semantics technique, instead of a neural language model. The latter remains to be explored, and perhaps could improve performance even further, especially for OOV data. The code for our implementation and the revised version of Mac-Morpho are available online at `https://github.com/erickrf/nlpnet`.

References

[Afonso et al. 2002] Afonso, S., Bick, E., Haber, R., and Santos, D. (2002). Floresta sintá(c)tica: a treebank for Portuguese. In *Proceedings of the Third International Conference on Language Resources and Evaluation (LREC 2002)*, pages 1698–1703.

[Aluísio et al. 2003] Aluísio, S., Pelizzoni, J., Marchi, A. R., de Oliveira, L., Manenti, R., and Marquiafável, V. (2003). An account of the challenge of tagging a reference corpus for brazilian portuguese. In *Proceedings of the 6th international conference on Computational processing of the Portuguese language*, PROPOR'03, pages 110–117, Berlin, Heidelberg. Springer-Verlag.

[Bick 2000] Bick, E. (2000). *The parsing system PALAVRAS: automatic grammatical analysis of Portuguese in a constraint grammar framework*. PhD thesis, Department of Linguistics – Aarhus University.

[Bird et al. 2009] Bird, S., Klein, E., and Loper, E. (2009). *Natural Language Processing with Python*. O'Reilly.

[Bruckschen et al. 2008] Bruckschen, M., Muniz, F., Souza, J. G. C., Fuchs, J. T., Infante, K., Muniz, M., Gonçalves, P. N., Vieira, R., and Aluísio, S. M. (2008). Anotação Lingüística em XML do Corpus PLN-BR. Technical report.

[Collobert et al. 2011] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

[dos Santos et al. 2008] dos Santos, C. N., Milidiú, R. L., and Rentería, R. P. (2008). *Portuguese Part-of-Speech Tagging Using Entropy Guided Transformation Learning*, volume 5190 of *Lecture Notes in Computer Science*, chapter 15, pages 143–152.

[Huang and Yates 2009] Huang, F. and Yates, A. (2009). Distributional Representations for Handling Sparsity in Supervised Sequence-Labeling. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 495–503.

[Kepler and Finger 2006] Kepler, F. N. and Finger, M. (2006). Comparing Two Markov Methods for Part-of-Speech Tagging of Portuguese. In Sichman, J. S., Coelho, H., and Rezende, S. O., editors, *Advances in Artificial Intelligence - IBERAMIA-SBIA 2006*, pages 482–491.

[Lund and Burgess 1996] Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior Research Methods, Instruments, & Computers*, 28(2):203–208.

[Maia and Xexéo 2011] Maia, M. R. d. H. and Xexéo, G. B. (2011). Part-of-Speech Tagging of Portuguese Using Hidden Markov Models with Character Language Model Emissions. In *Proceedings of the 8th Brazilian Symposium in Information and Human Language Technology*, pages 159–163.

[Turian et al. 2010] Turian, J., Ratinov, L., and Bengio, Y. (2010). Word representations: A simple and general method for semi-supervised learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 384–394.

[Turney and Pantel 2010] Turney, P. D. and Pantel, P. (2010). From Frequency to Meaning: Vector Space Models of Semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

[Widdows and Ferraro 2008] Widdows, D. and Ferraro, K. (2008). Semantic Vectors: A Scalable Open Source Package and Online Technology Management Application. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, pages 1183–1190.