Neural Networks 66 (2015) 11–21

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Multilingual part-of-speech tagging with weightless neural networks

Hugo C.C. Carneiro [a,*], Felipe M.G. França [a], Priscila M.V. Lima [b]

[a] Systems Engineering and Computer Science Program/COPPE, Universidade Federal do Rio de Janeiro (UFRJ) - Caixa Postal 68511, Cidade Universitária, Rio de Janeiro, Rio de Janeiro 21941-972, Brazil

[b] Instituto Tércio Pacitti de Aplicações e Pesquisas Computacionais (NCE), Universidade Federal do Rio de Janeiro (UFRJ) - Av. Athos da Silveira Ramos, 274 - Edifício do Centro de Ciências Matemáticas e da Natureza, Bloco E, Cidade Universitária, Rio de Janeiro, Rio de Janeiro 21941-916, Brazil

ARTICLE INFO

Article history:
Received 12 April 2014
Received in revised form 17 February 2015
Accepted 22 February 2015
Available online 2 March 2015

Keywords:
Weightless neural networks
Part-of-speech tagging

ABSTRACT

Training part-of-speech taggers (POS-taggers) requires iterative time-consuming convergence-dependable steps, which involve either expectation maximization or weight balancing processes, depending on whether the tagger uses stochastic or neural approaches, respectively. Due to the complexity of these steps, multilingual part-of-speech tagging can be an intractable task, where as the number of languages increases so does the time demanded by these steps. WiSARD (**Wi**lkie, **S**tonham and **A**leksander's **R**ecognition **D**evice), a weightless artificial neural network architecture that proved to be both robust and efficient in classification tasks, has been previously used in order to turn the training phase faster. WiSARD is a RAM-based system that requires only one memory writing operation to train each sentence. Additionally, the mechanism is capable of learning new tagged sentences during the classification phase, on an incremental basis. Nevertheless, parameters such as RAM size, context window, and probability bit mapping, make the multilingual part-of-speech tagging task hard. This article proposes mWANN-Tagger (**m**ultilingual **W**eightless **A**rtificial **N**eural **N**etwork **tagger**), a WiSARD POS-tagger. This tagger is proposed due to its one-pass learning capability. It allows language-specific parameter configurations to be thoroughly searched in quite an agile fashion. Experimental evaluation indicates that mWANN-Tagger either outperforms or matches state-of-art methods in accuracy with very low standard deviation, i.e., lower than 0.25%. Experimental results also suggest that the vast majority of the languages can benefit from this architecture.

1. Introduction

Part-of-speech tagging (POS-tagging) is a common task in natural language processing. It requires high accuracy since its result is commonly used as input (or as part of the input) to other tasks, e.g., syntactic parsing and machine translation. Multilingual POS-tagging presents a further challenge. Not only its accuracy must be high in every language, but also the tagger used must have an agile language-independent architecture. Nowadays, two different techniques are used: (i) several POS-taggers are trained independently, which can create some overhead, or (ii) cross-lingual POS-taggers are employed, which use previously annotated relations between words of different *corpora* (composed of texts in different languages) in order to remove tagging ambiguities (Naseem, Snyder, Eisenstein, & Barzilay, 2009; Snyder, Naseem, Eisenstein, & Barzilay, 2008, 2009). In the first case, once a new tagger is needed for a particular language, there is no technique to speed up the parameter tuning procedure. In both strategies, the architecture of the tagger is not truly language-independent. This article proposes a tagger with both a language-independent architecture and the ability to train taggers for new languages with little time spent on parameter tuning procedures.

Neural network models have proven useful in solving natural language processing tasks (Caridakis, Karpouzis, Drosopoulos, & Kollias, 2012; Hinoshita, Arie, Tani, Okuno, & Ogata, 2011; Klein, Kamp, Palm, & Doya, 2010). Neural-based taggers have been proposed since Schmid (1994), some of which employed the neuro-symbolic paradigm, such as Ma, Murata, Uchimoto, and Isahara (2000) and Marques, Bader, Rocio, and Hölldobler (2007). More recently, a weightless neural-based tagger was proposed in Carneiro, França, and Lima (2010). Despite the variety of techniques and parameters adjustment employed, it is observed that every neural tagger created ever since has only been used for monolingual part-of-speech tagging. This work explores the weightless neural

* Corresponding author.
E-mail addresses: hcesar@cos.ufrj.br (H.C.C. Carneiro), felipe@cos.ufrj.br (F.M.G. França), priscilamvl@gmail.com (P.M.V. Lima).

http://dx.doi.org/10.1016/j.neunet.2015.02.012
0893-6080/

paradigm for multilingual part-of-speech tagging through the proposal of mWANN-Tagger(**m**ultilingual **W**eightless **A**rtificial **N**eural **N**etwork **tagger**), in order to speed up the search of language-specific parameter configurations.

Several other POS-tagger models were proposed, such as rule-based ones (Brill, 1992, 1994) and those that work as a finite-state machine that employs sliding windows (Sánchez-Villamil, Forcada, & Carrasco, 2004, 2005). Of the very widespread probabilistic graphical models for POS-tagging, *hidden Markov models* (HMM) (Jurafsky & Martin, 2008; Manning & Schütze, 1999), *maximum entropy Markov model* (MEMM) (McCallum, Freitag, & Pereira, 2000) and *Conditional random fields* (CRF) (Lafferty, McCallum, & Pereira, 2001) constitute some of the most used techniques. However, HMMs may present some drawbacks to correct classification: (i) if the part of speech of a word cannot be inferred from the words in its vicinity, or (ii) if a word was not presented in the training *corpus*. The former problem was solved with the use of second-order Markov models through the use of trigrams (Brants, 2000; Petrov, Das, & McDonald, 2012). The latter incorporated the use of features to the probabilistic nature of HMM, initially proposed in Ratnaparkhi (1996).

Ratnaparkhi (1996) proposed the use of binary feature functions $f_j(w_i, t_i)$ to represent that word w_i appears with tag t_i in the *corpus*. Examples of feature functions can include information about if the word ends in a particular suffix, if it is capitalized, if it is a number and so on. This way, words could be substituted by a feature vector $\mathbf{f}(w_i, t_i)$, enabling the tagging of unpresented words. These words do not appear in the training *corpus*, however some of its characteristics could appear in the feature vector. A feature $f_j(w_i, t_i)$ could assume the value 1 if w_i possesses the characteristic associated to the feature, and 0 otherwise.

The model makes use of the *maximum entropy* formalism (ME), which states that the probability which best represents a given state of knowledge is the one with highest entropy. This probability is known as *maximum entropy probability distribution* or *Gibbs distribution* (Levine & Tribus, 1978). The model was optimized by maximizing the log-likelihood of this probability distribution.

The feature functions were incorporated to the Markovian architecture of HMM to create a more robust probabilistic graphical model, MEMM (McCallum et al., 2000). This model proved to be very effective by incorporating the global knowledge from probabilistic graphical models and the ability to tag unseen words more precisely through the use of arbitrary features. However, this model presented a drawback called the "label bias problem". This means that if there are states with low-entropy transitions to its following states, they will take little notice of an observation (Bottou, 1991; Lafferty et al., 2001). In order to overcome this limitation of MEMMs, Lafferty et al. (2001) proposed the CRF.

CRFs are defined according to two random variables, **X** over data sequences and **Y** over label sequences. In POS-tagging tasks, **X** range over natural language sentences and **Y** range over the possible strings of tags associated with **X**. Also, CRFs can be used in any possible graph $G = (V, E)$, but for the POS-tagging task it is recommended to use a chain-like graph. This particular case of CRF was called HMM-like CRF by Lafferty et al. (2001). This is the only kind of CRF that is detailed in this paper, since it is the one used in POS-tagging tasks. CRFs substitute the notion of dependency between states of HMMs and MEMMs by the use of features. This way, CRFs work with undirected graphs, differently from HMMs and MEMMs which use directed graphs. HMM-like CRFs employ two distinct types of features: one that is defined for each pair of states (y', y) and another for each pair of state-observations (y, x).

Those maximum entropy models proved to be quite versatile and able to tag texts quite accurately, especially CRF (Lafferty et al., 2001). However, as the number of features grows, their accuracy may diminish and the time spent during the training step increases considerably. This article proposes a POS-tagger that employs a one-pass learning model, whose optimal parameter configuration can be thoroughly searched in feasible time. The choice of a non-overtraining-prone model helps in producing taggers that keep a high accuracy despite the complexity of the feature space. This way, it is possible to create new accurate taggers more quickly. Comparison of language-specific characteristics could benefit from these studies.

A review of weightless neural models, especially WiSARD (**Wi**lkie, **S**tonham and **A**leksander's **R**ecognition **D**evice), is presented in Section 2, so that the reader is capable to understand how the WiSARD model tags sentences (Section 3). The experimental methodology used to test mWANN-Tagger capabilities on tagging texts in languages of distinct natures is discussed in Section 4. Experimental results are described and analyzed in depth in Section 5. Conclusion and future work directions are presented in Section 6.

2. Weightless artificial neural networks and WiSARD model

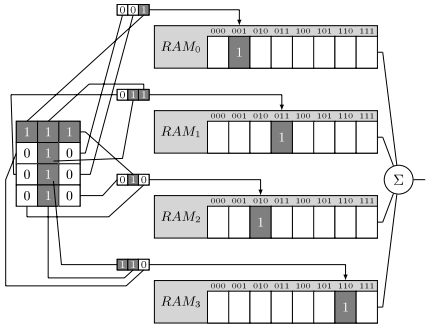
Weightless Artificial Neural Networks (WANNs) are a set of ANN models in which there is no synaptic weight balancing during the training phase. This lack of synaptic weight is compensated by the use of Random Access Memories (RAMs) inside its neural nodes, whereas traditional neural network neurons do not store any information, but only applies a multivariate nonlinear continuous function whose arguments are either the outputs given by the nodes in the previous layer or the network inputs.

There are several weightless artificial neural models, e.g., WiSARD (Aleksander, Thomas, & Bowden, 1984), and its variants, WiSART (a portmanteau of WiSARD and ART—Adaptive Resonance Theory Grossberg, 1987) (Fulcher, 1992), AUTOWISARD (an unsupervised learning extension of WiSARD that allows automatic generation of new discriminators) (Wickert, França, & Prieto, 2001) and others; Probabilistic Logic Nodes (Kan & Aleksander, 1987); Goal Seeking Neuron (Filho, Fairhurst, & Bisset, 1991); General Neural Unit (Aleksander & Morton, 1991); G-RAM (Generalizing Random Access Memory) (Aleksander, 1990a), as well as its most common implementation Virtual G-RAM (VG-RAM) (Mrsic-Flogel, 1991); Sparse Distributed Memory (Kanerva, 1988) and its integer counterpart (Snaider, Franklin, Strain, & George, 2013), and others. A detailed comparison between several weightless neural models can be found in Aleksander, Gregorio, França, Lima, and Morton (2009). This work adopts the WiSARD model with bleaching (Carvalho, Carneiro, França, & Lima, 2013; Grieco, Lima, Gregorio, & França, 2010), because it has the best trade-off between training agility, memory consumption and capability of avoiding saturation. Besides, the neural model proved promising in monolingual POS-tagging (Carneiro et al., 2010).

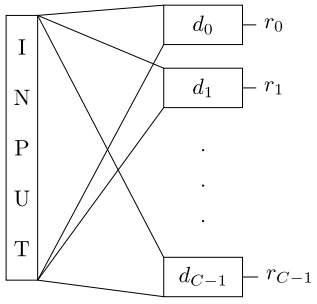
2.1. WiSARD model

The pioneering WiSARD n-tuple classifier constitutes the RAM-based neural network chosen as the basis of mWANN-Tagger architecture. Its main difference from other RAM-based models is the use of a structure called RAM-discriminator, depicted in Fig. 1(a). The discriminator receives inputs from a "retina" (a matrix of **0**s and **1**s, see Fig. 1(a)) mapped to a set of N RAMs via n RAM address bits and a summation device. The summation device Σ outputs the number of RAMs that responded positively to an input pattern. A set of address bits and RAMs constitutes a **RAM node**.

Mapping of retina pixels to the RAM nodes is effected via the address bits, usually in a *pseudorandom* (invariant for a discriminator) and biunivocal fashion (one retina pixel is associated to one and only one address bit of only one RAM).

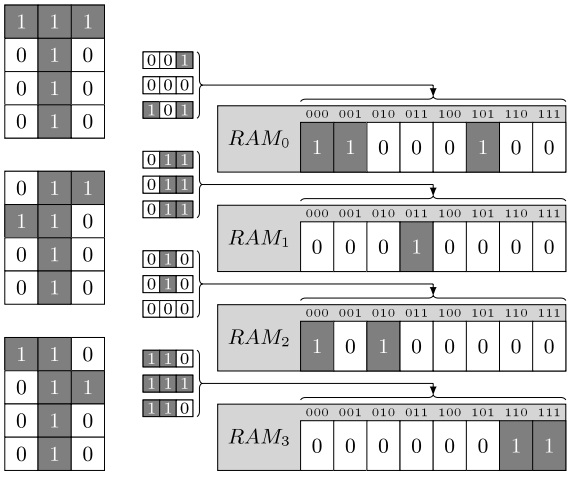


(a) RAM-discriminator structure.



(b) Multidiscriminator structure.

Fig. 1. WiSARD architecture.



(a) Training patterns.

(b) RAM configuration at the end of the training phase.

Fig. 2. WiSARD training procedure.

WiSARD, as any other weighted ANN, is capable of performing pattern recognition tasks, thus it must be capable to generalize in the case of unseen examples. This generalization capability is guaranteed by the use of multiple RAMs and the summation device.

The RAM-discriminator is capable of informing either if a pattern belongs to a given class or how similar it is to the examples presented for that class. WiSARD, on the other hand, can classify a pattern in one or more of several classes. In order to do so, WiSARD makes use of a multi-discriminator architecture (see Fig. 1(b)), in which each discriminator d_i informs the similarity measure of the input in relation to class i. The chosen classes are the ones whose discriminator produced the highest measure, therefore, they are the classes that are most similar to the input pattern. When using WiSARD to tag parts of speech, each class will be a distinct tag from the tagset.

Training phase. The training phase of a WiSARD network consists of two parts: (i) the initialization phase and (ii) the input pattern training itself. During the initialization phase every memory location is set to **0** and the input "retina" generates a pseudo-random mapping, creating groups of k bits from the network input array. These groups are organized into tuples. Each of these n-tuples are associated to the input of a specific RAM. Fig. 1(a) demonstrates how the input "retina" bits can be mapped into the RAMs through some of n-tuples. (In the case of Fig. 1(a), $n = 3$.)

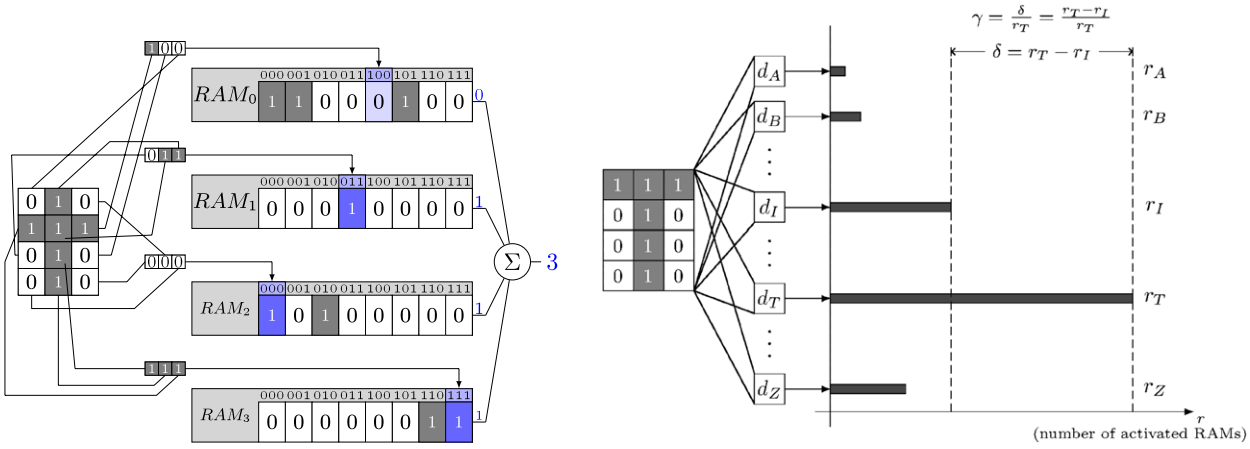
During the training phase itself, for each input pattern to be trained, a **1** is written in the memory positions addressed by the input of the RAMs belonging to the discriminator associated to the class of the input. The training of a pattern of a capital 'T' is shown in Fig. 1(a).

The training phase ends after all training data available have been presented to the network to be trained. Unlike weighted ANNs and some WANNs, in which the training phase requires an iterative process in order to achieve convergence, WiSARD only needs to be presented to the training data once for it to be fully trained. As exemplified in Fig. 2, training consists of presenting all three 'T'-patterns of Fig. 2(a) to the network. They are passed as input to the RAM nodes as illustrated in Fig. 2(b). In Fig. 2(b) each RAM node receives three inputs. Each RAM is fed with tuples representing subpatterns of the training set on the retina. By the end of the training phase, the network achieves a final configuration of its RAM nodes. This configuration is shown in Fig. 2(b). It must be noticed that classification can start at any moment and does not have to wait for the whole set of examples to be presented. Likewise, if a fourth instance is detected, it can be trained after classification had started.

Classification phase. By presenting an unseen example to the network, in order to classify it, each discriminator produces a similarity measure r_i. As previously mentioned, this measure is attainable by counting the number of RAMs whose memory positions are addressed by the n-tuple associated thereto. For example, see Fig. 3(a) where a cross-shaped 'T' is presented to the network. The image is mapped into n-tuples through the same procedure depicted in Figs. 1(a) and 2. The memory positions addressed by the tuples are marked in blue. Their content is then output by the RAM nodes. Finally, the sum of the RAM outputs compose the discriminator response: a *similarity measure* between the given pattern and the class of the discriminator. The class whose discriminator produces the highest similarity measure, r_{MAX}, is the one which is associated with the unseen input pattern. The confidence γ of this response is calculated by using the formula $\gamma = \frac{r_{\mathrm{MAX}} - r_{\mathrm{MAX}-1}}{r_{\mathrm{MAX}}}$, where $r_{\mathrm{MAX}-1}$ is the second highest measure. The higher the value of γ, the greater the likelihood of the input pattern belonging to the given class. Fig. 3(b) represents the comparison of discriminator responses. In Fig. 3(b) a 'T' is presented to the network. Each discriminator outputs a response. The discriminator of class 'T' d_T produces the highest response.

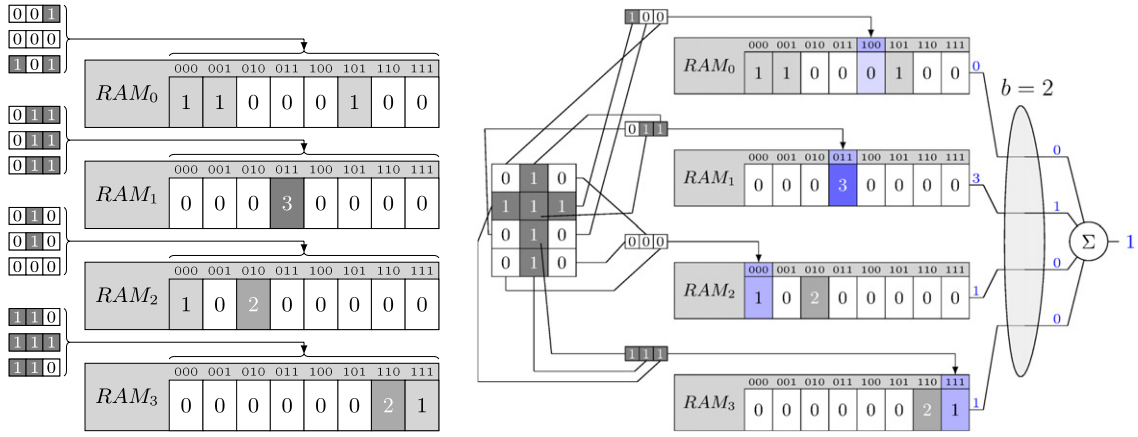
2.2. The bleaching technique and the end of ties

The use of large and usually noisy datasets showed to be harmful to WiSARD learning capability. If a large amount of noisy data is presented to the network during its training, several RAM positions will be set to **1**. This problem is called *saturation*. When the network is saturated, its RAMs tend to output **1** for most inputs. *Bleaching* is a technique proposed by Grieco et al. (2010) which tries to minimize the harms of saturation. It proposes the storage of integer values in the memory positions, instead of Boolean values. Metaphorically, instead of memory positions being either **white** (**0**) or **black** (**1**), they are represented by some shade of gray, black being the highest value to appear in a memory position



(a) Classification of a pattern by a discriminator. (b) Multidiscriminator response.

Fig. 3. WiSARD classification procedure. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) RAM configuration at the end of the training phase. (b) Classification of a pattern. Bleaching threshold $b = 2$.

Fig. 4. WiSARD training and classification procedures with bleaching.

of a discriminator and white **0**. Fig. 4(a) illustrates how the final configuration of the RAM nodes of Fig. 2 would look in a WiSARD with bleaching. According to the grayscale metaphor, black is represented by the value **3** and white by **0**.

Bleaching uses a same threshold b associated to every RAM node, turning any memory position value lower than b into **0**, and **1** otherwise. The bleaching threshold b starts as 1 and gets incremented by one while there are still ties in the discriminator responses. When there are no ties left, the class whose discriminator produces the highest response is the one which will be associated with the input pattern. An example of a classification using the bleaching technique (with $b = 2$) is presented in Fig. 4(b). In this example, the RAM nodes output integer values from 0 to 3. Only the outputs higher than 2 are passed as 1 to the summing device. The other outputs are passed as 0. Different values of b for different RAM nodes could be used. Nevertheless, in this article, the same value of b is used for every RAM node in a discriminator.

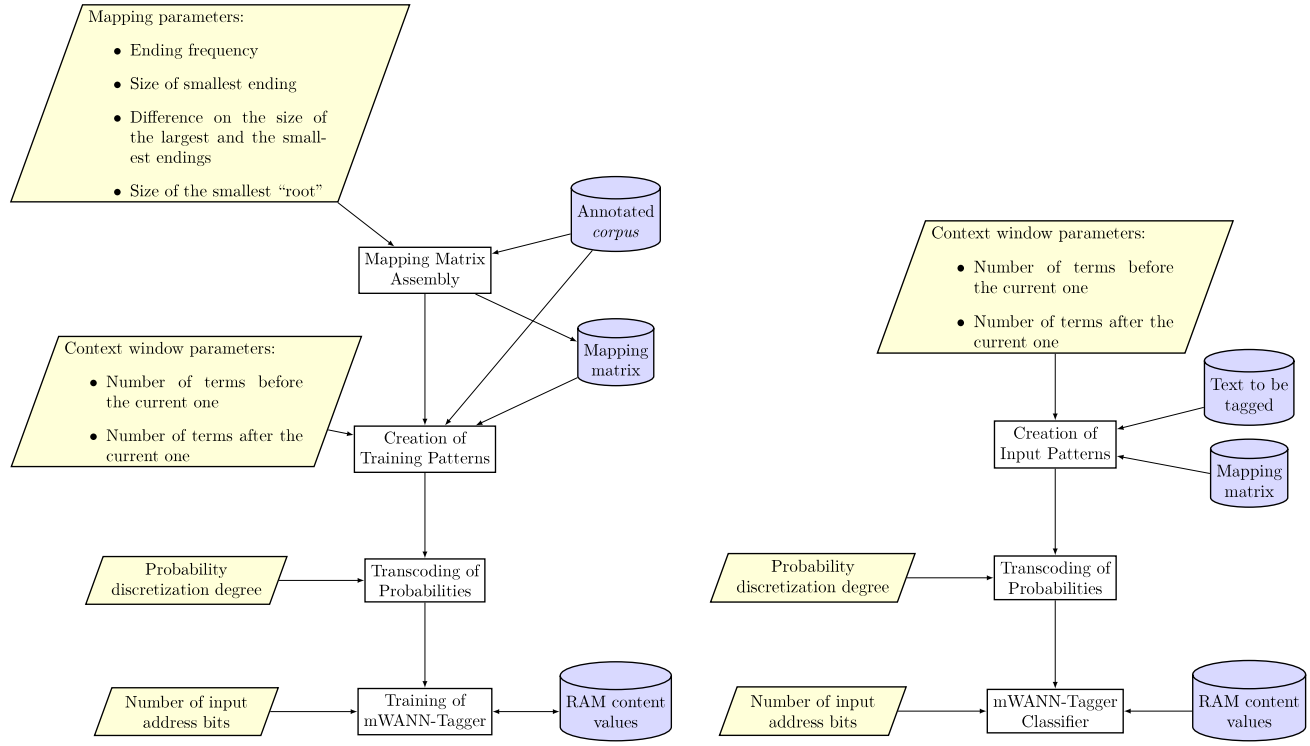
Despite the improvement achieved with plain bleaching, when a WiSARD is trained with a large amount of data, ties might occur even for a very high value of b. So, iteratively increasing b by 1 proves to be an inefficient procedure in these cases. This procedure can become faster by storing every pre-bleaching integer output from the RAMs, ordering them and then increase b according to those values. This algorithm guarantees that at least one discriminator response is lowered when b changes. This way, the ties are eliminated faster than in the original bleaching algorithm.

3. mWANN-Tagger

mWANN-Tagger constitutes a multilingual language-independent part-of-speech tagger. It is a solution to the training performance problem that arises when the number of languages used by a POS-tagger increases. It is intended to profit from the efficiency of weightless artificial neural networks in classification tasks and a method to straightforwardly predict the optimal values for some classification parameters. It is an evolution of WANN-Tagger (Carneiro et al., 2010), a part-of-speech tagger whose architecture is based on the WiSARD model. mWANN-Tagger can be used in two distinct modes, the training and the tagging mode. A schematic diagram of the these modes are depicted in Fig. 5(a) and (b), respectively. A more detailed explanation of the arguments of mWANN-Tagger is given in Section 3.2. The training procedure is better described in Section 3.3.

3.1. Tagset

In order to create a multilingual POS-tagger, a universal tagset is proved necessary as the tagger requires a fixed amount of tags. This happens because there must be both a correlation among the grammatical categories of every language and a normalization of the tagsets. Thus, any distinction between the datasets is reduced. Only the parameters of the tagger need calibration. This is important because one of the goals of this work is to investigate a correlation between the indices of synthesis and the values of those



(a) Training mode. (b) Tagging mode.

Fig. 5. The modes of mWANN-Tagger.

Table 1
Tagset of mWANN-Tagger.

N	Noun	ADJ	Adjective
ADV	Adverb	V	Verb
PRON	Pronoun	DET	Determiner
ADP	Adposition	NUM	Cardinal Number
CJ	Conjunction	MW	Measure Word
PART	Particle	INTJ	Interjection
PUNC	Punctuation	MISC	Miscellaneous

arguments. Furthermore, Petrov et al. (2012) results corroborate the use of a universal tagset instead of specific ones for each language.

The tagset used in this article (Table 1) is similar to the one used in Petrov et al. (2012), except that it has a separate tag for measure words, as it is an important grammatical category of far eastern languages, and another one for interjections. Another similar common tagset was adopted in Naseem et al. (2009).

3.2. mWANN-Tagger arguments

mWANN-Tagger modules use eight arguments calibrated in different ways, according to:

- **Ending frequency**, the number of distinct words that end with a particular sequence of characters. As there are many sequences of letters that do not add relevant information about the part of speech of a given word, then only the sequences that occur in the end of at least X words will count for the creation of the tagger. X is the *ending frequency* value;
- **Size of the smallest ending**, the smallest number of characters that a word ending must have in order to be considered an ending candidate. This argument is used to avoid short sequences of letters, because there are languages whose endings usually do not have less than X letters. For instance, in English *-y* is the only one-letter ending, informing that the word is probably an adjective (*ugly*, *fairy* and others). Examples of two-letter endings include-*ed* (verb ending associated with both the past and the participle form, e.g., *liked*, *formed*, *cried* and *landed*) and-*nt* (adjective ending related to participle forms borrowed from Romance languages, e.g., *constant*, *consistent*, *reluctant* and *sentient*);
- **Size of the largest ending**. This argument is used to avoid creating enormous dictionaries and thus try to mitigate the effects of overtraining;
- **Size of the smallest root** is an argument analogous to the "size of the smallest ending". This value is important because it avoids considering word endings that are almost the size of the word itself. One trivial example is the word *ugly* and the ending-*ly* (denoting adverbs like *properly*, *hardly* and others). As *ug* is a small character sequence, mWANN-Tagger would choose-*y* (adjective) as *ugly*'s ending, instead of-*ly* (adverb). Another example is the word *bed* that carries the ending-*ed*, which usually represents verbs in the past and participle forms. Nevertheless, *bed* is a noun and not a verb;
- **Amount of words before the current one** is one of the two arguments that represent the context window. Context windows that are too small may leave important information unconsidered when tagging a certain word. On the other hand, large ones can also be hazardous to the tagging process, because of the introduction of too much specificity, therefore producing an effect called *overtraining*;
- **Amount of words after the current one** is the other argument related to the context window size;
- **Probability discretization degree**. As described above, mWANN-Tagger uses a WiSARD architecture, so it must receive only Boolean entries. The grammatical category probabilities must be discretized into bit arrays;
- **Number of RAM address bits** is an argument that determines how generalizing the tagger will be. If the size of the input retina is equally divided into the RAMs address bits, then one may consider that n, the size of the RAM input address, determines the

	ADJ	ADP	ADV	CJ	DET	INTJ	MISC	MW	N	NUM	PART	PRON	PUNC	V
brave	1.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
relief	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
twin	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.67	0.0	0.0	0.0	0.0	0.0
much	0.04	0.0	0.58	0.0	0.38	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
time	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.01
flies	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
like	0.03	0.0	0.0	0.78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.19
an	0.0	0.0	0.0	0.003	0.997	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
⋮														
·	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0
⋮														
-rrow	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.67	0.0	0.0	0.0	0.0	0.0
-ness	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.98	0.0	0.0	0.0	0.0	0.02
-ate	0.32	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.36	0.0	0.0	0.0	0.0	0.32
⋮														
-	0.07	0.1	0.06	0.05	0.12	0.0004	0.006	0.0	0.23	0.01	0.01	0.05	0.13	0.16

(a) Mapping matrix example lines.

Time flies like an arrow.
[time|N] [flies|V] [like|PRP] [an|DET] [arrow|N] [.|PUNC]

(b) Sentence and its corresponding annotated version.

T

	ADJ	ADP	ADV	CJ	DET	INTJ	MISC	MW	N	NUM	PART	PRON	PUNC	V
$B\rightarrow$ -	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
time	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.99	0.0	0.0	0.0	0.0	0.01
$A\rightarrow$ flies	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0	0.0

(c) Training pattern example with $B = 1, A = 1$ and $T = 14$ tags.

	ADJ	ADP	ADV	CJ	DET	INTJ	MISC	MW	N	NUM	PART	PRON	PUNC	V
an	0.0	0.0	0.0	0.01	0.99	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
-rrow	0.33	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.67	0.0	0.0	0.0	0.0	0.01
·	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0

(d) Training pattern example of the ending -rrow.

Fig. 6. mWANN-Tagger procedure of transcoding sentences into probabilities.

number of input bits to be correlated. N, the number of RAMs, is a consequence of the choice of n:

$$N = \frac{\text{bits in the input retina}}{n}. \quad (1)$$

It has been shown (Aleksander, 1990b) that the generalization capacity of the mechanism diminishes with the number of RAMs.

It is worth discussing a concept introduced in this work, the *ending relevance*. The relevance of an ending is important to determine the possible endings that can substitute words that appear in the tagging mode but that did not show in the training mode. It is a function of four arguments of mWANNTagger: ending frequency; size of the smallest ending; difference between the largest and the smallest ending, and the size of the smallest root. These arguments were used as a language-independent alternative to the manual selection of endings that prove to be relevant to the POS-tagging task. The manual selection of the endings requires a previous knowledge of morphological and semantical characteristics of a language. Thus, the ending relevance alternative is required by the tagger, as one of mWANN-Tagger's main objectives is to be language-independent.

3.3. Input pattern assembling

mWANN-Tagger employs an n-tuple classifier in order to tag sentences and thus, its input must be an array of bits. However, the tagger is trained on an annotated *corpus*, i.e., a dataset of annotated sentences composed by pairs $(x, \text{tag}(x))$ (see Fig. 6(b)). Hence, mWANN-Tagger requires an input assembling procedure, so that each sentence of size l in the annotated *corpus* is transcoded into l bit arrays, one for each term in the sentence.

At first, mWANN-Tagger assembles an $m \times n$ mapping matrix. m is the number of distinct terms of the *corpus* (words, punctuation, ciphers, foreign words and so on) plus the number of relevant endings obtained from the *corpus*, as described in Section 3.2, plus 1, to include a *general case*. The lines of the mapping matrix associated with the relevant endings are used in the tagging mode if there is no line in the mapping matrix associated to a new term that is presented to the tagger. If there is no line in the mapping

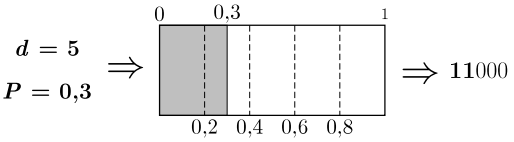


Fig. 7. Probability discretization procedure.

matrix associated neither to this new term nor to any of its possible endings, the general case is used.

Each line of the mapping matrix contains $n = T + 1$ cells. One of these cells stores the term (or ending) associated with the line. The other T cells store the empirical probabilities p of term (or ending) of the line w being tagged as any one of the T existing grammatical categories (in this work, $T = 14$, see Table 1). The empirical probability p of a term w being tagged as a category t_i is calculated through the formula

$$p(w, t_i) = \frac{C(w, t_i)}{\sum_{j=1}^{|T|} C(w, t_j)} \quad (2)$$

where $C(x, y)$ represents a counting function that returns the number of times the term x is tagged as y.

Examples of lines of the mapping matrix are shown in Fig. 6(a), where examples of endings include-*rrow* (that stands for words like *barrow*, *arrow* and *sorrow*),-*ness* (for *boldness* and *randomness*) and-*ate* (for *rotate* and *adequate*). The default case appears at the bottom of the figure, represented by a single hyphen (-). The process of assembling the mapping matrix is similar to the procedures used in dictionary-based models (Gnecco, Kůrková, & Sanguineti, 2011a, 2011b).

After mWANN-Tagger assembles the mapping matrix, the tagger creates training patterns for every term of each sentence in the training *corpus* (see Fig. 5(a)). Each pattern is composed by the empirical probabilities of every term (or ending) in the context window of its corresponding term. The pattern is represented by $B + A + 1$ lines, each one representing one term of the context window, i.e., B terms before, A terms after and the current term. Each of these lines contain the T empirical probabilities associated to its respective term (or ending) according to the mapping matrix. If there is a gap with no term in the context window, the empirical probabilities must be filled with 0.0 for every corresponding tag. Examples of training patterns are depicted in Fig. 6(c) and (d). Both figures use $B = A = 1$. It is important to note the lack of a term in the context window in Fig. 6(c) and the use of an ending in Fig. 6(d).

Next, the tagger must transcode the probabilities of the training patterns into bit arrays in order to train these patterns, as shown in Fig. 5(a). This transcoding procedure depends on a discretization process, since probabilities lie in the continuous range [0, 1]. A *probability discretization degree d* is employed in order to transcode a probability into a bit array. This degree represents the number of bits in which the probabilities must be discretized. It transcodes a probability p into a bit array with the first $\lceil p \times d \rceil$ positions marked with 1, and with 0 elsewhere. This is equivalent to say that if the interval [0, 1] were divided into d slices, the discretization of probability p into d bits depended on how many slices this probability (fully or partially) fills, as represented in Fig. 7.

When the training patterns are transcoded into bit arrays, the tagger can execute the training procedure itself. mWANN-Tagger trains the pattern as a traditional n-tuple classifier with bleaching (see Section 2.2). The tagger associates one discriminator d_i to each tag t_i. Thus, when a pattern is trained and it is associated with the tag t_i, the content of the RAM nodes of d_i must be updated.

The tagging mode is quite similar to the training mode, as shown in Fig. 5(b). In this mode, non-annotated texts are presented to the tagger. Afterwards, it converts the texts into patterns containing probabilities, according to the mapping matrix. This conversion occurs in the same way training patterns are created. Therefore, mWANN-Tagger creates one pattern for each term w of a non-annotated text. The tagger, then, transcodes these probabilities into bit arrays, which are used as input to the n-tuple classifier. Finally, the discriminator d_i with the highest response is the one chosen by mWANN-Tagger. The tag t_i associated to this discriminator is the one output by the tagger as the most probable grammatical category of a term w.

The tagger uses a context window in its POS-tagging task. This means that the network only needs local knowledge in order to be trained accurately, differently from probabilistic graphical models. These models use belief propagation algorithms (Baum, Petrie, Soules, & Weiss, 1970; Pearl, 1982), so that the knowledge about the problem is propagated to all of its nodes. Peres and Pedreira (2010) corroborate the mWANN-Tagger approach by suggesting that the use of local knowledge in certain domains may keep the method as powerful as if its knowledge were totally global.

4. Experimental methodology

Distinct languages were used in the experiments to test mWANN-Tagger capability for tagging texts in languages of distinct characteristics. For each language, several mWANN-Tagger parameter configurations were tested, until the optimal one was found. The languages chosen to be used in the experiments can be seen in Section 4.1.

4.1. Datasets

Eight languages were chosen for the experiments in order to obtain a sufficient amount of data for the evaluation task. These languages are listed below along in this subsection with the reasons for their choice.

- **Mandarin Chinese** (ISO-639-1 code: ZH - **Zh**ōngwén) was chosen because it is a very isolating language. Derivational and inflectional morphemes are almost absent. The majority of words is composed purely by roots (usually one or two) (Li & Thompson, 1981; Norman, 1988);
- **English** (ISO-639-1 code: EN) belongs to the Germanic branch of the Indo-European languages, which possesses Indo-European compositional constructions and an inflectional system. However, English inflection system was drastically simplified through the ages (verb conjugation no longer depends on person or number, case system ceased to exist and others). In addition, English borrowed several words from French, for instance *table* and *independence*. The latinization of English vocabulary contributed to increase its derivational index of synthesis. Because of its simplified inflectional system, a simple compositional structure and its indirect influence of Latin derivation rules, English was chosen as one of the languages to be used in the experiments (Baugh & Cable, 2002; Blake, 1992; Hogg, 1992);
- **Japanese** (ISO-639-1 code: JA) belongs to an isolate language family, the Japonic one. Thus, Japanese has some unique morphological structures. But, due to Chinese influences, Japanese borrowed several words from Early Middle Chinese, which nowadays constitute its Sino-Japanese vocabulary. These words are composed only by roots (usually one or two). This caused Japanese compositional index of synthesis to be somewhat high, even Japanese not being an isolating language like the Chinese ones (Loveday, 1996);
- **Portuguese** (ISO-639-1 code: PT), as a Romance language, makes almost no use of word compounds. Latin, through abolishing the compositional structures, required new forms of word creation and, consequently, new derivational suffixes and prefixes were introduced (Chase, 1900). This causes all Romance languages to have a low compositional index of

Table 2
The *corpora* used in the experiments.

ISO-639-1 code	*Corpus*	Number of terms
ZH	Penn Chinese Treebank 6.0 (Xia et al., 2000)	1 099 570
EN	Brown Corpus (Francis & Kučera, 1964)	1 166 194
JA	TüBa-J/S—Tübinger Baumbank des Japanischen Spontansprache (Hinrichs et al., 2000)	156 871
PT	Bosque (Floresta Sintá(c)tica) (Afonso et al., 2002)	213 999
IT	TUT—Turin University Treebank (Bosco et al., 2000)	76 938
DE	NEGRA Corpus (Skut et al., 1997)	337 959
RU	Set of shuffled sentences from Russian National Corpus (Boguslavsky et al., 2002)	166 308
TR	METU-SabancıTurkish Treebank (Oflazer et al., 2003)	53 981

synthesis and a somewhat high derivational one. Furthermore, Portuguese employs contractions, which prevents its compositional index of synthesis from being very low. Portuguese was chosen to be part of the experiments because of these criteria, and also because it simplified Latin noun declension and turned its plural marking into a simple agglutinative structure (Williams, 1938);

- **Italian** (ISO-639-1 code: IT), just as Portuguese, has a low compositional index of synthesis and a somewhat high derivational one. But, differently from Portuguese, its plural form keeps the fusional structure of Latin. Italian also employs a larger amount of contractions than Portuguese does, thus making its compositional index of synthesis a little bit higher than Portuguese one. Italian is used in this article because it is a Romance language that keeps Latin fusional inflection system not only in verbs but also in nouns (Maiden, 1995);
- **German** (ISO-639-1 code: DE – **De**utsch), as a member of the Germanic branch of the Indo-European languages, employs compounding as one of the word formation techniques. It also makes heavy use of derivation and its inflection system was not simplified as the English one was. German, as a language that keeps most word building techniques from the Germanic branch, and that has not suffered a massive latinization of its vocabulary, is a good candidate language to be used in the experiments (Carr, 1939; Neef, 1998);
- **Russian** (ISO-639-1 code: RU), of all Indo-European languages listed in this work, is the one that kept most of the Indo-European word building techniques. Word compounds are quite common in Russian, derivational affixes are extensively used and its inflectional system is huge and complex. This language is among the ones chosen for the experiments as it represents the very synthetic fusional languages (Levin, 1978; Townsend, 1975);
- **Turkish** (ISO-639-1 code: TR) is commonly known as a highly agglutinative language. Besides, as a counterweight to Russian fusional structure, Turkish is used in the experiments representing the group of languages that are very synthetic and purely agglutinative (Lewis, 2001).

One annotated *corpus* was used for each language (their detailed information are in Table 2). During the selection procedure the tagsets of the original corpora were converted into the universal one, as mentioned in Section 3.1. Thus, the datasets used in the experiments are somewhat normalized as their tagsets are the same.

5. Results and discussion

The robustness of mWANN-Tagger can be verified according to two criteria: (i) if the tagger tags sentences both accurately and precisely, and (ii) if it is quite fast in any of its phases, empirically showing the advantages of information-storage training algorithms. This way, two distinct experiments were carried out: (i) a comparison between the best results presented by mWANN-Tagger with state-of-art methods (see Table 5), and (ii) a measurement of mWANN-Tagger time costs in training and tagging phases. The accuracies of mWANN-Tagger are compared with the accuracies of a CRF trained with the same endings produced by the preprocessing step of mWANN-Tagger. A comparison with the accuracies obtained by Petrov et al. (2012) is also made. Both models are described in more detail in Section 5.1. The same CRF model was used for a time cost comparison with mWANN-Tagger. The time spent by both models are discussed in Section 5.2.

A good performance of mWANN-Tagger in the experiments of Sections 5.1 and 5.2 would empirically show that mWANN-Tagger can be used as an agile and precise language-independent POS-tagger.

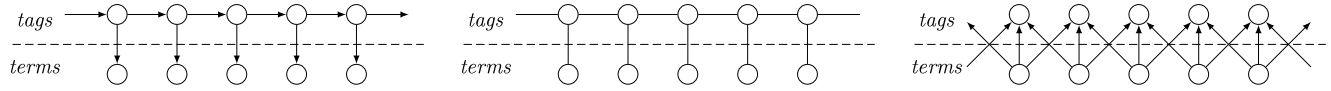
5.1. Comparison of accuracy results with the state of the art

Table 3 presents the best parameter configuration of mWANN-Tagger for each language of the dataset. As mentioned in the beginning of Section 5, those results were compared with the CRF model and with the results obtained in Petrov et al. (2012). The CRF model was used to experimentally show that mWANN-Tagger was competitive or even better than stochastic models that take advantage of employing features, e.g. word endings, to tag words. A traditional chain-structured CRF was employed. The model used the same endings produced by mWANN-Tagger preprocessing step. Those endings compose the feature vector of the CRF model. The amount of endings used in each language can be found in Table 4. The CRF implementation was made using MALLET toolkit (McCallum, 2002). This toolkit was chosen because it is written in the same programming language of mWANN-Tagger and because it was developed by the academic group of Prof. Andrew McCallum, one of the creators of CRF (Lafferty et al., 2001; McCallum, 2003). The experiments of CRF were run in the same machines the ones of mWANN-Tagger ran.

The work of Petrov et al. (2012) was also compared to mWANN-Tagger because it represents the state of the art in multilingual POS-tagging. Petrov et al. (2012)'s work consisted of creating a universal part-of-speech tagset and implementing a POS-tagger to show how a universal tagset could produce a better result in part-of-speech tasks than a set of customized tagsets. Petrov et al. (2012) used an instance of a well-known stochastic POS-tagger whose main advantage is its robustness and its fast training. They used an implementation of the Trigrams'n'Tags POS-Tagger (TnT-Tagger) (Brants, 2000), which is a statistical POS-tagger that uses second order Markov models.

Both CRF and TnT-Tagger make use of the advantages of belief propagation algorithms (Baum et al., 1970; Pearl, 1982) to acquire a global knowledge of the text. On the other hand, mWANN-Tagger uses only the words inside its context window, which is usually small (2 to 4 words). The contrast between the knowledge acquired to tag sentences can be noticed in the graphical representations of the models depicted in Fig. 8. Finally, both the tagger used in this paper and the one used in Petrov et al. (2012) share some properties, such as acting on distinct independently trained languages and using a single standardized tagset.

According to Table 5, mWANN-Tagger accuracy values are consistently higher than the ones obtained by CRF. mWANN-Tagger has a higher accuracy than those reported by Petrov et al. (2012) in 6 out of the 8 languages compared. 3 of them being consistently higher. In Portuguese mWANN-Tagger was competitive with the work of Petrov et al. (2012). The standard deviation of mWANN-Tagger accuracy is lower than 0.25% whereas the average one of the TnT-Tagger is 2.25%. The standard deviation of CRF ranges from as low as 1.0% in English to as high as 3.6% in Italian.



(a) HMM model. (b) Chain-CRF model. (c) mWANN-Tagger model.

Fig. 8. Relation structures between tag nodes and term nodes.

Table 3
Best mWANN-Tagger parameter configuration for each language.

ISO-639-1 code	ZH	EN	JA	PT	IT	DE	RU	TR
Ending frequency	5	3	1	2	2	2	1	1
Size of smallest ending	3	2	5	2	2	2	2	3
Size of largest ending	4	4	5	4	3	5	4	6
Size of smallest root	3	2	1	2	2	2	1	0
Context window	[2, 1]	[1, 2]	[2, 1]	[2, 2]	[2, 1]	[1, 1]	[1, 1]	[2, 2]
Prob. discret. degree	142	107	45	82	70	100	61	113
# RAM address bits	95	66	27	60	39	61	52	32
Mean accuracy	93.53%	97.65%	98.61%	96.80%	96.15%	96.93%	97.01%	92.26%
Accuracy std. deviation	0.10%	0.03%	0.12%	0.13%	0.19%	0.16%	0.13%	0.23%

Table 4
The average amount of endings used per fold for each language.

ISO-639-1 code	Number of endings per fold
ZH	329.7
EN	4350.0
JA	1150.3
PT	3896.9
IT	714.5
DE	8225.3
RU	6691.7
TR	16286.9

Table 5
Comparison between accuracy results obtained by mWANN-Tagger, CRF and the TnT-Tagger used in Petrov et al. (2012).

ISO-639-1 language code	mWANN-Tagger	CRF	Petrov et al. (2012)	Uses same *corpus*?
ZH	**93.53**%	86.48%	93.4%	Yes
EN	**97.65**%	96.35%	96.8%	No
JA	**98.61**%	96.54%	98.0%	Yes
PT	**96.80**%	94.29%	**96.8**%	Yes
IT	**96.15**%	92.02%	95.8%	No
DE	96.93%	94.48%	**97.9**%	Yes
RU	**97.01**%	93.10%	96.8%	Yes[a]
TR	**92.26**%	80.31%	89.1%	Yes

[a] By the time this article was written, only a set of shuffled sentences of the Russian National Corpus were freely available off-line. Due to some technical and/or copyright problems, the rest of the *corpus* was only accessible on-line.

5.2. Time performance

After mWANN-Tagger best parameter values were found, the tagger was tested 100 times with this configuration in order to measure its performance. The same evaluation methodology was made with CRF. Only one 10-fold cross-validation procedure was executed because CRF takes a long time to converge. Table 6 contains the time spent by mWANN-Tagger and CRF during the training and the tagging steps, for every sentence of each language *corpus*.

The time spent by mWANN-Tagger in the training step is 2 orders of magnitude lower than CRF. On the other hand, CRF tags sentences up to 80 times faster than mWANN-Tagger. The time spent in bleaching technique is potentially the main cause of mWANN-Tagger being slower than CRF. This can be mitigated by using multiple threads during the tagging step, since WiSARD architecture is highly parallelizable (Gregorio & Giordano, 2014). The time spent by CRF during the training step can be explained by its convergence-dependable nature and, possibly, to the large number of features produced (see Table 4). Preliminary experiments were made with the MEMM architecture. They showed that MEMM accuracy is lower than that of CRF, but the former trains up to 10 times faster than the latter. Nonetheless, its training time performance is still considerably lower than that of mWANN-Tagger.

Table 6
Average time spent by mWANN-Tagger during each step for each term of every language.

ISO-639-1 language code	Training step		Tagging step	
	mWANN-Tagger (ms)	CRF (ms)	mWANN-Tagger (ms)	CRF (ms)
ZH	0.20	19.36	2.35	0.03
EN	0.87	95.89	5.59	0.22
JA	0.05	7.36	0.44	0.02
PT	0.10	6.71	0.52	0.02
IT	0.05	4.16	0.33	0.02
DE	0.08	11.22	0.40	0.03
RU	0.04	7.72	0.23	0.02
TR	0.25	9.86	2.21	0.03

The high amount of features may have also implied in the low accuracy of CRF, shown in Table 5. The high amount of features possibly caused overfitting of the model. This hypothesis could be corroborated by checking how the number of features used decreases from simple models like ME (Ratnaparkhi, 1996) to the more sophisticated MEMM (McCallum et al., 2000). In CRF (Lafferty et al., 2001) the number of features employed is very small. For example, the number of possible endings/suffixes decreases from every ending X, such that $|X| \leq 4$ in Ratnaparkhi (1996) to only 9 English-specific suffixes in Lafferty et al. (2001). This indicates that there are some impairments to the usage of CRF for multilingual POS-tagging. This problem can possibly be solved by a less naive process of ending choice. However, the strategy can increase the overhead of the preprocessing step.

The time spent at each step depends on the value of the tagger parameters. During the preprocessing step mWANN-Tagger creates both the mapping and the training files, so, any parameter that affects the size of these files makes the time elapsed increase or decrease accordingly. The parameters that are directly proportional to the increase of the size of both files, and thus of the time spent in the preprocessing phase, are the both the *size of the smallest ending* and the *size of the largest ending* (affecting the size of the mapping file), and the two parameters A and B that change the context window size (affecting the size of the training file). The ones that are inversely proportional are the *ending frequency* and the *size of the smallest root*, responsible for the size of the mapping file. However, the preprocessing step took less than 1 s in almost every language.

As the CRF model uses almost the same preprocessing step, it was not included in the comparison made in this section.

The time spent in the training and tagging steps highly depends on the size of the input retina. It is directly proportional to the *probability discretization degree* and to the context window parameters, as the former increases the amount of bits used for each probability representation and the latter the amount of words, and, thus, the probabilities, in the context window.

The time spent during the tagging step also depends on the number of RAM nodes of the network. As the number of RAM nodes increases, so does the time to choose the best bleaching threshold. Considering that during the tagging step the mapping file must be accessed in such a way that the probabilities of every word can be obtained, parameters that affect the size of this file also affect the time spent at this step. Nevertheless, their effect on the increasing (or decreasing) of the tagger performance during this step is still rather low when compared to that of the other three parameters.

6. Conclusion and future work

This article introduced a novel part-of-speech tagger that uses a weightless artificial neural network architecture, mWANN-Tagger. Its architecture allows for a one-pass training technique: mWANN-Tagger's training phase is faster than any other tagger to this date. Although its tagging phase is not faster than traditional stochastic models, it is still capable of tagging in an agile fashion. Compared with state-of-art methods, mWANN-Tagger's accuracy matched or outperformed them, thus confirming that methods that use local knowledge can perform better than the ones using belief propagation algorithms, e.g., Markov models and conditional random fields.

The main intuition of this tagger was to use the ending and/or, analogously, the beginning of words. Nevertheless, it lacked the ability to successfully tag the parts of speech of languages that possess a nonconcatenative morphology, for instance Arabic, Hebrew, Syriac and others (McCarthy, 1981). This is left for a future investigation.

This work showed that mWANN-Tagger is capable of training and tagging sentences in a quite agile fashion. However, the assembling of the mapping matrix and the use of endings avoid the tagger to use one of benefits of the weightless paradigm: the ability to train new example during runtime and thus improve itself. An improvement in the assembling of the mapping matrix, considering the interleaving between the training and the tagging phases would make mWANN-Tagger be able to train new examples during runtime, becoming a more versatile tagger.

There is evidence indicating a strong correlation between the index of synthesis and the best architecture for each language (Carneiro, 2012). Herein it is conjectured that the optimal parameter configuration of the tagger could be obtained through a function of the index of synthesis of a given language (Greenberg, 1960). So, an immediate direction for extending mWANN-Tagger would be to study how these measures correlate, which constitutes work in progress.

Another possible mWANN-Tagger extension is to use computational linguistics techniques oriented to Semitic languages (Cohen-Sygal & Wintner, 2006; Kiraz, 2000, 2001), so that these languages would have their parts of speech accurately tagged. Finally, mWANN-Tagger could be used as the basis of a weightless neural-based grammar induction system. This system could employ a mutual learning architecture, because mWANN-Tagger has a fast performance during both the training and the tagging steps, and is also capable of learning during runtime. This mutual learning architecture would consist in using mWANN-Tagger to tag sentences with the k most probable sequences of parts of speech, which would then be used to train the grammar induction system. After a couple of iterations, this system would be capable of determining which of these k sequences is the correct one. If that were not the one mWANN-Tagger classified as the most likely sequence, then the tagger would be trained with this new sequence.

Acknowledgments

This work was partially supported by CAPES (grant number 2575/2012), CNPq (grant number 307437/2013-2) and FAPERJ (grant number E-26/103.177/2011) Brazilian research agencies.

References

Afonso, S., Bick, E., Haber, R., & Santos, D. (2002). Floresta Sintá(c)tica: a treebank for Portuguese. In *Proceedings of the 3rd international conference on language resources and evaluation LREC-2002* (pp. 1698–1703).

Aleksander, I. (1990a). Ideal neurons for neural computers. In R. Eckmiller, G. Hartmann, & G. Hauske (Eds.), *Parallel processing in neural systems and computers*. New York, NY, USA: Elsevier Science Inc..

Aleksander, I. (1990b). *An introduction to neural computing*. London: Chapman and Hall.

Aleksander, I., Gregorio, M.D., França, F.M.G., Lima, P.M.V., & Morton, H. (2009). A brief introduction to weightless neural systems. In *Proceedings of the 17th European symposium on artificial neural networks, computational intelligence and machine learning ESANN-2009* (pp. 299–305).

Aleksander, I., & Morton, H. B. (1991). General neural unit: retrieval performance. *Electronics Letters*, *27*, 1776–1778.

Aleksander, I., Thomas, W. V., & Bowden, P. A. (1984). WISARD: a radical step foward in image recognition. *Sensor Review*, *4*, 120–124.

Baugh, A. C., & Cable, T. (2002). *A history of the english language*. Prentice Hall.

Baum, L. E., Petrie, T., Soules, G., & Weiss, N. (1970). A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *The Annals of Mathematical Statistics*, *41*, 164–171.

Blake, N. (1992). *The Cambridge history of the english language, Vol. 2.* (pp. 1066–1476). Cambridge University Press.

Boguslavsky, I., Chardin, I., Grigorieva, S., Grigoriev, N., Iomdin, L., Kreidlin, L., & Frid, N. (2002). Development of a dependency treebank for Russian and its possible applications in NLP. In *Proceedings of the 3rd international conference on language resources and evaluation LREC-2002* (pp. 852–856).

Bosco, C., Lombardo, V., Vassallo, D., & Lesmo, L. (2000). Building a treebank for Italian: a data-driven annotation schema. In *Proceedings of the 2nd international conference on language resources and evaluation LREC-2000* (pp. 99–105).

Bottou, L. (1991). *Une Approche théorique de l'Apprentissage Connexionniste: applications à la Reconnaissance de la Parole*. (Ph.D. thesis), Orsay, France: Université de Paris XI.

Brants, T. (2000). TnT: a statistical part-of-speech tagger. In *Proceedings of the sixth conference on applied natural language processing* (pp. 224–231). Stroudsburg, PA, USA: Association for Computational Linguistics.

Brill, E. (1992). A simple rule-based part-of-speech tagger. In *Proceedings of ANLP-92, 3rd conference on applied natural language processing, Trento, IT* (pp. 152–155).

Brill, E. (1994). Some advances in transformation-based part of speech tagging. In *Proceedings of the twelth national conference on artificial intelligence, AAAI'94* (pp. 722–727).

Caridakis, G., Karpouzis, K., Drosopoulos, A., & Kollias, S. (2012). Non parametric, self organizing, scalable modeling of spatiotemporal inputs: the sign language paradigm. *Neural Networks*, *36*, 157–166.

Carneiro, H. C. C. (2012). *A Função do Índice de Síntese das Linguagens na Classificação Gramatical com Redes Neurais Sem Peso*. (Master's thesis), Rio de Janeiro, Brazil: Universidade Federal do Rio de Janeiro.

Carneiro, H. C. C., França, F. M. G., & Lima, P. M. V. (2010). WANN-Tagger: a weightless artificial neural network tagger for the Portuguese language. In J. Filipe, & J. Kacprzyk (Eds.), *ICFC-ICNC 2010—proceedings of the international conference on fuzzy computation and international conference on neural computation, [parts of the international joint conference on computational intelligence IJCCI 2010], Valencia, Spain, October 24–26, 2010* (pp. 330–335). SciTePress.

Carr, C. T. (1939). Nominal compounds in Germanic. In *St. Andrews University Publications: Vol. 41*. Oxford University Press.

Carvalho, D.S., Carneiro, H.C.C., França, F.M.G., & Lima, P.M.V. (2013). B-bleaching: agile overtraining avoidance in the wisard weightless neural classifier. In *Proceedings of the 21st European symposium on artificial neural networks, computational intelligence and machine learning ESANN-2013* (pp. 515–520).

Chase, G. D. (1900). The form of nominal compounds in latin. *Harvard Studies in Classical Philology*, *11*, 61–72.

Cohen-Sygal, Y., & Wintner, S. (2006). Finite-state registered automata for non-concatenative morphology. *Computational Linguistics*, *32*, 49–82.

Filho, E. C. B. C., Fairhurst, M. C., & Bisset, D. L. (1991). Adaptive pattern recognition using goal seeking neurons. *Pattern Recognition Letters*, *12*, 131–138.

Francis, W. N., & Kučera, H. (1964). *A standard corpus of present-day edited American english, for use with digital computers. Technical report*. Providence, Rhode Island, US: Department of Linguistics, Brown University.

Fulcher, E. P. (1992). WiS-ART: unsupervised clustering with ram discriminators. *International Journal of Neural Systems*, *3*, 57–63.

Gnecco, G., Kůrková, V., & Sanguineti, M. (2011a). Can dictionary-based computational models outperform the best linear ones? *Neural Networks*, *24*, 881–887.
Gnecco, G., Kůrková, V., & Sanguineti, M. (2011b). Some comparisons of complexity in dictionary-based and linear computational models. *Neural Networks*, *24*, 171–182.
Greenberg, J. H. (1960). A quantitative approach to the morphological typology of language. *International Journal of American Linguistics*, *26*, 178–194.
Gregorio, M.D., & Giordano, M. (2014). Change detection with weightless neural networks. In *IEEE conference on computer vision and pattern recognition, CVPR workshops 2014, Columbus, OH, USA, June 24–27, 2014* (pp. 403–407).
Grieco, B. P. A., Lima, P. M. V., Gregorio, M. D., & França, F. M. G. (2010). Producing pattern examples from "mental" images. *Neurocomputing*, *73*, 1057–1064.
Grossberg, S. (1987). Competitive learning: from interactive activation to adaptive resonance. *Cognitive Science*, *11*, 23–163.
Hinoshita, W., Arie, H., Tani, J., Okuno, H. G., & Ogata, T. (2011). Emergence of hierarchical structure mirroring linguistic composition in a recurrent neural network. *Neural Networks*, *24*, 311–320.
Hinrichs, E. W., Bartels, J., Kawata, Y., Kordoni, V., & Telljohann, H. (2000). The verbmobil treebanks. In *KONVENS 2000 / Sprachkommunikation, Vorträge der gemeinsamen Veranstaltung 5. Konferenz zur Verarbeitung natürlicher Sprache (KONVENS), 6. ITG-Fachtagung "Sprachkommunikation"* (pp. 107–112). VDE-Verlag GmbH.
Hogg, R. M. (1992). *The Cambridge history of the english language*: *Vol. 1. The beginnings to 1066*. Cambridge University Press.
Jurafsky, D., & Martin, J. H. (2008). *(Prentice Hall series in artificial intelligence), Speech and language processing (2nd edition)* (2nd ed.). Prentice Hall.
Kan, W.K., & Aleksander, I. (1987). A probabilistic logic neuron network for associative learning. In *IEEE first international conference on neural networks* pp. 541–548.
Kanerva, P. (1988). *Sparse distributed memory*. Cambridge, MA, USA: MIT Press.
Kiraz, G. A. (2000). Multitiered nonlinear morphology using multitape finite automata: a case study on Syriac and Arabic. *Computational Linguistics*, *26*, 77–105.
Kiraz, G. A. (2001). *Computational nonlinear morphology: survey of semitic computational morphology*. Cambridge University Press.
Klein, M., Kamp, H., Palm, G., & Doya, K. (2010). A computational neural model of goal-directed utterance selection. *Neural Networks*, *23*, 592–606.
Lafferty, J., McCallum, A., & Pereira, F. (2001). Conditional random fields: probabilistic models for segmenting and labeling sequence data. In *Proc. 18th international conf. on machine learning* (pp. 282–289). San Francisco, CA: Morgan Kaufmann.
Levin, M. I. (1978). *Maximum entropy formalism*. Columbus, OH, USA: Slavica.
Levine, R. D., & Tribus, M. (1978). *Russian declension and conjugation*. Cambridge, MA, USA: MIT Press.
Lewis, G. (2001). *Turkish grammar*. Oxford University Press.
Li, C. N., & Thompson, S. A. (1981). *Mandarin Chinese: a functional reference grammar*. Berkeley: University of California Press.
Loveday, L. J. (1996). *Language contact in Japan: a sociolinguistic history*. Oxford University Press.
Ma, Q., Murata, M., Uchimoto, K., & Isahara, H. (2000). Hybrid neuro and rule-based part of speech taggers. In *Proceedings of the 18th conference on computational linguistics—volume 1* (pp. 509–515). Stroudsburg, PA, USA: Association for Computational Linguistics.
Maiden, M. (1995). *A linguistic history of Italian*. Longman Publishing Group.
Manning, C. D., & Schütze, H. (1999). *Foundations of statistical natural language processing*. Cambridge, Mass.: MIT Press.
Marques, N. C., Bader, S., Rocio, V., & Hölldobler, S. (2007). Neuro-symbolic word tagging. In J. Neves, M. F. Santos, & J. Machado (Eds.), *New trends in artificial intelligence*. APPIA - Associação Portuguesa para a Inteligência Artificial.
McCallum, A.K. (2002). MALLET: A Machine Learning for Language Toolkit. URL: http://www.cs.umass.edu/~mccallum/mallet.
McCallum, A. (2003). Efficiently inducing features of conditional random fields. In *Nineteenth conference on uncertainty in artificial intelligence, UAI03*.
McCallum, A., Freitag, D., & Pereira, F. (2000). Maximum entropy Markov models for information extraction and segmentation. In *Proceeding 17th international conference on machine learning* (pp. 591–598). San Francisco, CA: Morgan Kaufmann.
McCarthy, J. J. (1981). A prosodic theory of nonconcatenative morphology. *Linguistic Inquiry*, *12*, 373–418.
Mrsic-Flogel, J. (1991). Convergence properties of self-organizing maps. In T. Kohonen, K. Mäkisara, O. Simula, & J. Kangas (Eds.), *Proceedings of the international conference on artificial neural networks 1991 (Espoo, Finland)* (pp. 879–886). Amsterdam, New York: North-Holland.
Naseem, T., Snyder, B., Eisenstein, J., & Barzilay, R. (2009). Multilingual part-of-speech tagging: two unsupervised approaches. *Journal of Artificial Intelligence Research*, *36*, 341–385.
Neef, M. (1998). A case study in declarative morphology: German case inflection. In W. Kehrein, & R. Wiese (Eds.), *Phonology and morphology of the Germanic languages*. Tübingen: Max Niemeyer Verlag.
Norman, J. L. (1988). *Chinese*. Cambridge University Press.
Oflazer, K., Say, B., Hakkani-Tür, D. Z., & Tür, G. (2003). Building a Turkish treebank. *Treebanks*, 261–277.
Pearl, J. (1982). Reverend bayes on inference engines: a distributed hierarchical approach. In *Proceedings of the American association of artificial intelligence national conference on AI, Pittsburgh, PA* (pp. 133–136).
Peres, R. T., & Pedreira, C. E. (2010). A new local–global approach for classification. *Neural Networks*, *23*, 887–891.
Petrov, S., Das, D., & McDonald, R. (2012). A universal part-of-speech tagset. In *LREC–language resources evaluation conference*.
Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the conference on empirical methods in natural language processing*.
Sánchez-Villamil, E., Forcada, M. L., & Carrasco, R. C. (2004). Unsupervised training of a finite-state sliding-window part-of-speech tagger. In J. L. V. González, P. Martínez-Barco, R. Muñoz, & M. Saiz-Noeda (Eds.), *EsTAL* (pp. 454–463). Springer.
Sánchez-Villamil, E., Forcada, M.L., & Carrasco, R.C. (2005). Parameter reduction in unsupervisedly trained sliding-window part-of-speech taggers. In *International conference recent advances in natural language processing* (pp. 466–471).
Schmid, H. (1994). Part-of-speech tagging with neural networks. In *Proceedings of the 15th conference on computational linguistics—volume 1* (pp. 172–176). Stroudsburg, PA, USA: Association for Computational Linguistics.
Skut, W., Krenn, B., Brants, T., & Uszkoreit, H. (1997). An annotation scheme for free word order languages. In *Proceedings of the 5th conference on applied natural language processing* (pp. 88–95). Association for Computational Linguistics.
Snaider, J., Franklin, S., Strain, S., & George, E. O. (2013). Integer sparse distributed memory: analysis and results. *Neural Networks*, *46*, 144–153.
Snyder, B., Naseem, T., Eisenstein, J., & Barzilay, R. (2008). Unsupervised multilingual learning for POS tagging. In *Proceedings of the conference on empirical methods in natural language processing* (pp. 1041–1050). Stroudsburg, PA, USA: Association for Computational Linguistics.
Snyder, B., Naseem, T., Eisenstein, J., & Barzilay, R. (2009). Adding more languages improves unsupervised multilingual part-of-speech tagging: a Bayesian non-parametric approach. In *Proceedings of human language technologies: the 2009 annual conference of the north American chapter of the association for computational linguistics* (pp. 83–91). Stroudsburg, PA, USA: Association for Computational Linguistics.
Townsend, C. (1975). *Russian word-formation*. Columbus, OH, USA: Slavica.
Wickert, I., & França, F. M. G. (2001). AUTOWISARD: unsupervised modes for the WISARD. In J. Mira, & A. Prieto (Eds.), *Lecture Notes in Computer Science*: *Vol. 2084. Connectionist models of neurons, learning processes, and artificial intelligence* (pp. 435–441). Berlin, Heidelberg: Springer.
Williams, E. B. (1938). *From latin to Portuguese: historical phonology and morphology of the Portuguese language*. University of Pennsylvania Press.
Xia, F., Palmer, M., Xue, N., Okurowski, M.E., Kovarik, J., Chiou, F.D., Huang, S., Kroch, T., & Marcus, M. (2000). Developing guidelines and ensuring consistency for Chinese text annotation. In *Proceedings of the 2nd international conference on language resources and evaluation LREC-2000*.