Learning Character-level Representations for Part-of-Speech Tagging

Cícero Nogueira dos Santos CICERONS@BR.IBM.COM
IBM Research - Brazil, Av. Pasteur, 138/146 – Rio de Janeiro, 22296-903, Brazil

Bianca Zadrozny BIANCAZ@BR.IBM.COM
IBM Research - Brazil, Av. Pasteur, 138/146 – Rio de Janeiro, 22296-903, Brazil

Abstract

Distributed word representations have recently been proven to be an invaluable resource for NLP. These representations are normally learned using neural networks and capture syntactic and semantic information about words. Information about word morphology and shape is normally ignored when learning word representations. However, for tasks like part-of-speech tagging, intra-word information is extremely useful, specially when dealing with morphologically rich languages. In this paper, we propose a deep neural network that learns character-level representation of words and associate them with usual word representations to perform POS tagging. Using the proposed approach, while avoiding the use of any handcrafted feature, we produce state-of-the-art POS taggers for two languages: English, with 97.32% accuracy on the Penn Treebank WSJ corpus; and Portuguese, with 97.47% accuracy on the Mac-Morpho corpus, where the latter represents an error reduction of 12.2% on the best previous known result.

1. Introduction

Distributed word representations, also known as word embeddings, have recently been proven to be an invaluable resource for natural language processing (NLP). One of the key advantages of using word embeddings is minimizing the need for handcrafted features. These representations are normally learned using neural networks and capture syntactic and semantic information about words. With a few exceptions (Luong et al., 2013), work on learning of representations for NLP has focused exclusively on the word level. However, information about word morphology and shape is crucial for various NLP tasks. Usually, when a task needs morphological or word shape information, the knowledge is included as handcrafted features (Collobert, 2011). One task for which character-level information is very important is part-of-speech (POS) tagging, which consists in labeling each word in a text with a unique POS tag, e.g. noun, verb, pronoun and preposition.

In this paper we propose a new deep neural network (DNN) architecture that joins word-level and character-level representations to perform POS tagging. The proposed DNN, which we call CharWNN, uses a convolutional layer that allows effective feature extraction from words of any size. At tagging time, the convolutional layer generates character-level embeddings for each word, even for the ones that are outside the vocabulary. We present experimental results that demonstrate the effectiveness of our approach to extract character-level features relevant to POS tagging. Using CharWNN, we create state-of-the-art POS taggers for two languages: English and Portuguese. Both POS taggers are created from scratch, without including any handcrafted feature. Additionally, we demonstrate that a similar DNN that does not use character-level embeddings only achieves state-of-the-art results when using handcrafted features.

Although here we focus exclusively on POS tagging, the proposed architecture can be used, without modifications, for other NLP tasks such as text chunking and named entity recognition.

This work is organized as follows. In Section 2, we describe the proposed the CharWNN architecture. In Section 3, we discuss some related work. Section 4 details our experimental setup and results. Finally, in Section 5 we present our final remarks.

2. Neural Network Architecture

The deep neural network we propose extends Collobert et al.'s (2011) neural network architecture, which is a variant of the architecture first proposed by Bengio et al. (2003).

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

Given a sentence, the network gives for each word a score for each tag $\tau \in T$. In order to score a word, the network takes as input a fixed-sized window of words centralized in the target word. The input is passed through a sequence of layers where features with increasing levels of complexity are extracted. Although the network scores each word separately, we can take the output for the whole sentence and use the Viterbi algorithm to perform structured prediction. The novelty in our network architecture is the inclusion of a convolutional layer to extract character-level representations. In the following subsections we detail the proposed architecture.

2.1. Initial Representation Levels

The first layer of the network transforms words into real-valued feature vectors (embeddings) that capture morphological, syntactic and semantic information about the words. We use a fixed-sized word vocabulary V^{wrd}, and we consider that words are composed of characters from a fixed-sized character vocabulary V^{chr}. Given a sentence consisting of N words $\{w_1, w_2, ..., w_N\}$, every word w_n is converted into a vector $u_n = [r^{wrd}; r^{wch}]$, which is composed of two sub-vectors: the *word-level embedding* $r^{wrd} \in \mathbb{R}^{d^{wrd}}$ and the *character-level embedding* $r^{wch} \in \mathbb{R}^{cl_u}$ of w_n. While word-level embeddings are meant to capture syntactic and semantic information, character-level embeddings capture morphological and shape information.

2.1.1. Word-Level Embeddings

Word-level embeddings are encoded by column vectors in an embedding matrix $W^{wrd} \in \mathbb{R}^{d^{wrd} \times |V^{wrd}|}$. Each column $W_i^{wrd} \in \mathbb{R}^{d^{wrd}}$ corresponds to the word-level embedding of the i-th word in the vocabulary. We transform a word w into its word-level embedding r^{wrd} by using the matrix-vector product:

$$r^{wrd} = W^{wrd} v^w \tag{1}$$

where v^w is a vector of size $\left|V^{wrd}\right|$ which has value 1 at index w and zero in all other positions. The matrix W^{wrd} is a parameter to be learned, and the size of the word-level embedding d^{wrd} is a hyper-parameter to be chosen by the user.

2.1.2. Character-Level Embeddings

Robust methods to extract morphological information from words must take into consideration all characters of the word and select which features are more important for the task at hand. For instance, in the POS tagging task, informative features may appear in the beginning (like the prefix "*un*" in "*unfortunate*"), in the middle (like the hyphen in "*self-sufficient*" and the "*h*" in "*10h30*"), or at the end (like suffix "*ly*" in "*constantly*"). In order to tackle this problem we use a convolutional approach, which has been first introduced by Waibel et al. (1989). As depicted in Fig. 1, our convolutional approach produces local features around each character of the word and then combines them using a max operation to create a fixed-sized character-level embedding of the word.

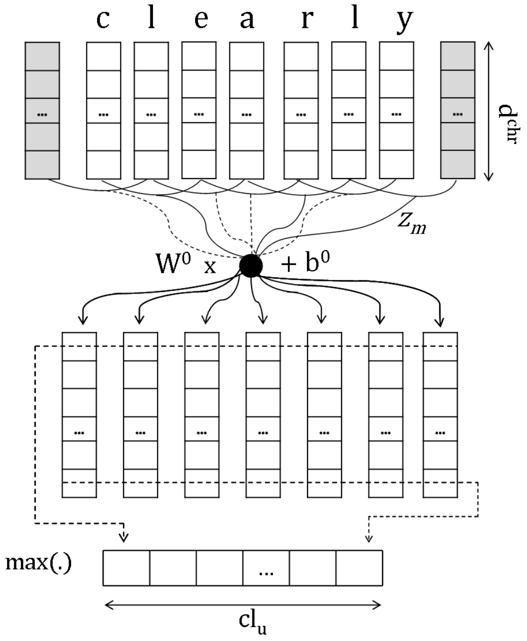

Figure 1. Convolutional approach to character-level feature extraction.

Given a word w composed of M characters $\{c_1, c_2, ..., c_M\}$, we first transform each character c_m into a character embedding r_m^{chr}. Character embeddings are encoded by column vectors in the embedding matrix $W^{chr} \in \mathbb{R}^{d^{chr} \times |V^{chr}|}$. Given a character c, its embedding r^{chr} is obtained by the matrix-vector product:

$$r^{chr} = W^{chr} v^c \tag{2}$$

where v^c is a vector of size $\left|V^{chr}\right|$ which has value 1 at index c and zero in all other positions. The input for the convolutional layer is the sequence of character embeddings $\{r_1^{chr}, r_2^{chr}, ..., r_M^{chr}\}$.

The convolutional layer applies a matrix-vector operation to each window of size k^{chr} of successive windows in the sequence $\{r_1^{chr}, r_2^{chr}, ..., r_M^{chr}\}$. Let us define the vector $z_m \in \mathbb{R}^{d^{chr}k^{chr}}$ as the concatenation of the character embedding m, its $(k^{chr} - 1)/2$ left neighbors, and its $(k^{chr} - 1)/2$ right neighbors:

$$z_m = \left(r_{m-(k^{chr}-1)/2}^{chr}, ..., r_{m+(k^{chr}-1)/2}^{chr}\right)^T$$

The convolutional layer computes the j-th element of the vector r^{wch}, which is the character-level embedding of w, as follows:

$$[r^{wch}]_j = \max_{1<m<M} \left[W^0 z_m + b^0\right]_j \tag{3}$$

where $W^0 \in \mathbb{R}^{cl_u \times d^{chr} k^{chr}}$ is the weight matrix of the convolutional layer. The same matrix is used to extract local features around each character window of the given word. Using the max over all character windows of the word, we extract a “global” fixed-sized feature vector for the word.

Matrices W^{chr} and W^0, and vector b^0 are parameters to be learned. The size of the character vector d^{chr}, the number of convolutional units cl_u (which corresponds to the size of the character-level embedding of a word), and the size of the character context window k^{chr} are hyper-parameters to be chosen by the user.

2.2. Scoring and Structured Inference

We follow Collobert et al.’s (2011) window approach to score all tags T for each word in a sentence. This approach follows the assumption that the tag of a word depends mainly on its neighboring words, which is true for various NLP tasks, including POS tagging. Given a sentence with N words $\{w_1, w_2, ..., w_N\}$, which have been converted to joint word-level and character-level embedding $\{u_1, u_2, ..., u_N\}$, to compute tag scores for the n-th word in the sentence, we first create a vector x_n resulting from the concatenation of a sequence of k^{wrd} embeddings, centralized in the n-th word:

$$x_n = \left(u_{n-(k^{wrd}-1)/2}, ..., u_{n+(k^{wrd}-1)/2}\right)^T$$

We use a special *padding token* for the words with indices outside of the sentence boundaries. Next, the vector x_n is processed by two usual neural network layers, which extract one more level of representation and compute the scores:

$$s(x_n) = W^2 h(W^1 x_n + b^1) + b^2 \tag{4}$$

where matrices $W^1 \in \mathbb{R}^{hl_u \times k^{wrd}(d^{wrd}+cl_u)}$ and $W^2 \in \mathbb{R}^{|T| \times hl_u}$, and vectors $b^1 \in \mathbb{R}^{hl_u}$ and $b^2 \in \mathbb{R}^{|T|}$ are parameters to be learned. The transfer function $h(.)$ is the hyperbolic tangent. The size of the context window k^{wrd} and the number of hidden units hl_u are hyper-parameters to be chosen by the user.

In POS tagging, the tags of neighboring words are strongly dependent. Some tags are arranged in chunks (e.g, proper names with two or more words), and some tags are very unlikely to be followed by other tags (e.g. verbs are very unlikely to follow determiners). Therefore, a sentence-wise tag inference that captures structural information from the sentence can deal better with tag dependencies. Like in (Collobert et al., 2011), we use a prediction scheme that takes into account the sentence structure. The method uses a transition score $A_{t,u}$ for jumping from tag $t \in T$ to $u \in T$ in successive words, and a score $A_{0,t}$ for starting from the t-th tag. Given the sentence $[w]_1^N = \{w_1, w_2, ..., w_N\}$, the score for tag path $[t]_1^N = \{t_1, t_2, ..., t_N\}$ is computed as follows:

$$S\left([w]_1^N, [t]_1^N, \theta\right) = \sum_{n=1}^{N} \Big(A_{t_{n\_1},t_n} + s(x_n)_{t_n}\Big) \tag{5}$$

where $s(x_n)_{t_n}$ is the score given for tag t_n at word w_n and θ is the set of all trainable network parameters $\big(W^{wrd}, W^{chr}, W^0, b^0, W^1, b^1, W^2, b^2, A\big)$.

Using the Viterbi (1967) algorithm, we can infer the predicted sentence tags $[t^*]_1^N$, which are the ones composing the path that leads to the maximal score:

$$[t^*]_1^N = \underset{[t]_1^N \in T^N}{\arg\max}\, S\left([w]_1^N, [t]_1^N, \theta\right) \tag{6}$$

2.3. Network Training

Our network is trained by minimizing a negative likelihood over the training set D. In the same way as in (Collobert et al., 2011), we interpret the sentence score (5) as a conditional probability over a path. For this purpose, we exponentiate the score (5) and normalize it with respect to all possible paths. Taking the log, we arrive at the following conditional log-probability:

$$\begin{aligned}\log p\left([t]_1^N | [w]_1^N, \theta\right) = S\left([w]_1^N, [t]_1^N, \theta\right) - \\ \log\left(\sum_{\forall [u]_1^N \in T^N} e^{S\left([w]_1^N, [u]_1^N, \theta\right)}\right)\end{aligned} \tag{7}$$

The log-likelihood in Equation 7 can be computed efficiently using dynamic programming (Collobert, 2011).

We use stochastic gradient descent (SGD) to minimize the negative log-likelihood with respect to θ:

$$\theta \mapsto \sum_{([w]_1^N, [y]_1^N) \in D} -\log p([y]_1^N | [w]_1^N, \theta) \tag{8}$$

where $[w]_1^N$ corresponds to a sentence in the training corpus D and $[y]_1^N$ represents its respective tag labeling.

The backpropagation algorithm is a natural choice to efficiently compute gradients of network architectures such as the one proposed in this work (Lecun et al., 1998; Collobert, 2011). In order to perform our experiments, we implement the proposed CharWNN architecture using the

Theano library (Bergstra et al., 2010). *Theano* is a versatile Python library that allows the efficient definition, optimization, and evaluation of mathematical expressions involving multi-dimensional arrays. We use *Theano*'s automatic differentiation capabilities in order to implement the backpropagation algorithm.

3. Related Work

Our work was mainly inspired by the work of (Collobert et al., 2011) which has the specific goal of avoiding task-specific engineering of features for standard natural language processing tasks, while still achieving good results. A large amount of unlabeled data is used for learning word embeddings, which are obtained as a side effect of training a language model network by stochastic gradient minimization of a ranking criterion. The word embeddings are then used to initialize the word lookup tables of neural networks trained for four specific tasks: POS tagging, chunking (CHUNK), named entity recognition (NER) and semantic role labeling (SRL). For POS and CHUNK the generalization performance obtained without hand-crafted features is quite close to that of the benchmarks used for comparisons and can be further improved using ensemble methods. For NER and SRL there is a larger performance gap that they show is possible to reduce using extra information from gazetteers and by cascading the output from POS and CHUNK. For SRL, further improvement can be obtained by providing parse tree information as extra features. In a footnote, they comment that it would be ideal to learn directly from character sequences rather than words which would allow capturing the morphological relationships between different variants of a word, but leave this out of the scope of their paper. Here, we pursue this direction, which is specially important for languages that have morphologically complex words and for domains such as biological text, with complicated but logical word structure.

The importance of taking into consideration the morphological structure of words for natural language processing appears in other related work. (Alexandrescu & Kirchhoff, 2006) present a factored neural language model where each word is represented as a vector of features such as stems, morphological tags and cases. Then a single embedding matrix is used to look up all of these features. Although this allows handling new words, the word representations do not encode the morphological information, but rather are encoded as network parameters. Thus this information cannot be transferred to other tasks as is the case with the approach proposed in this paper.

In (Luong et al., 2013) they also choose to operate at the morpheme level, assuming access to a dictionary of morphemic analyses of words. Then, they use a recursive neural network (RNN) to explicitly model the morphological structures of words and learn morphologically-aware embeddings. They evaluate the quality of these embeddings on a word similarity task, instead of using them for standard natural language processing tasks such as POS tagging as we do here. They show that the quality of the embeddings they propose is improved when they combine the RNNs with a language model neural network that utilizes surrounding words context, creating what they call a context-sensitive morphological RNN.

Previously, (Lazaridou et al., 2013) had used compositional distributional semantic models, originally designed to learn meanings of phrases, to derive representations for complex words, in which the base unit is the morpheme, similar to (Luong et al., 2013). But, as noted by (Luong et al., 2013), their models can only combine a stem with an affix and do not support recursive morpheme composition. Furthermore, they also require a corpus of stem/derived pairs for training, which we do not require here.

4. Experimental Setup and Results

In this section, we present the experimental setup and results of applying CharWNN to POS tagging of English and Portuguese languages.

4.1. Unsupervised Learning of Word-Level Embeddings

Word-level embeddings play a very important role in the CharWNN architecture. They are meant to capture syntactic and semantic information, which are crucial to POS tagging. Recent work has showed that large improvements in terms of model accuracy can be obtained by performing unsupervised pre-training of word embeddings (Collobert et al., 2011; Luong et al., 2013; Zheng et al., 2013; Socher et al., 2013).

In our experiments, we perform unsupervised learning of word-level embeddings using the *word2vec* tool[1], which implements the *continuous bag-of-words* and *skip-gram* architectures for computing vector representations of words (Mikolov et al., 2013).

We use the December 2013 snapshot of the English Wikipedia corpus as the source of unlabeled English text. The corpus was processed using the following steps: (1) remove paragraphs that are not in English; (2) substitute non-roman characters for a special character; (3) tokenize the text using the tokenizer available with the Stanford POS Tagger (Manning, 2011); (4) and remove sentences that are less than 20 characters long (including white spaces) or

[1] https://code.google.com/p/word2vec/

have less than 5 tokens. Like in (Collobert et al., 2011) and (Luong et al., 2013), we lowercase all words and substitute each numerical digit by a 0 (for instance, *1967* becomes *0000*). The resulting clean corpus contains about 1.75 billion tokens.

For the Portuguese experiments, we use three sources of unlabeled text: the Portuguese Wikipedia; the CETENFolha[2] corpus; and the CETEMPublico[3] corpus. The Portuguese Wikipedia corpus is preprocessed using the same approach applied to the English Wikipedia corpus. However, we implemented our own Portuguese tokenizer. The CETENFolha and CETEMPublico corpora are distributed in a tokenized format. They also contain tags indicating sentences, lists, author of the text, etc. We only include the parts of the corpora tagged as sentences. Additionally, for these two corpora, we execute the processing step (4), which consists in removing short sentences. We also lowercase all words and substitute each numerical digit by a 0. When concatenating the three corpora: Portuguese Wikipedia, CETENFolha and CETEMPublico, the resulting corpus contains around 401 million words.

For both languages we use the same parameters when running the *word2vec* tool, except for the minimum word frequency. For English, a word must occur at least 10 times in order to be included in the vocabulary, which resulted in a vocabulary of 870,214 entries. For Portuguese, we use a minimum frequency of 5, which resulted in a vocabulary of 453,990 entries. To train our word-level embeddings we use *word2vec*'s skip-gram method with a context window of size five. The training time for the English corpus is around 1h30min using 12 threads in a Intel® Xeon® E5-2643 3.30GHz machine.

We do not perform unsupervised learning of character-level embeddings. For both English and Portuguese, the character vocabulary is quite small if compared to the word vocabulary. Therefore, we assume that the amount of data in the labeled POS tagging training corpora is enough to effectively train character-level embeddings. It is important to note that character-level embeddings are trained using the raw (not lowercased) words. Therefore, the network is allowed to capture relevant information about capitalization.

4.2. POS Tagging Datasets

We use the *Wall Street Journal (WSJ)* portion of the Penn Treebank[4] (Marcus et al., 1993) for English POS tagging. In order to fairly compare our results with previous work, we adopt the same partitions used by other researchers (Manning, 2011; Søgaard, 2011; Collobert et al., 2011). This dataset contains 45 different POS tags. In Table 1, we present additional details about the WSJ corpus. In this table, the last two columns respectively inform the number of out-of-the-supervised-vocabulary words (OOSV), and the number of out-of-the-unsupervised-vocabulary words (OOUV). A word is considered OOSV if it does not appear in the training set, while OOUV words are the ones that do not appear in the vocabulary created using the unlabeled data (as described in Sec. 4.1), i.e, words for which we do not have word embeddings.

Table 1. WSJ Corpus for English POS Tagging.

SET	SENT.	TOKENS	OOSV	OOUV
TRAINING	38,219	912,344	0	6317
DEVELOP.	5,527	131,768	4,467	958
TEST	5,462	129,654	3,649	923

We use the Mac-Morpho corpus to perform POS tagging of Portuguese language text. The Mac-Morpho corpus (Aluísio et al., 2003) contains around 1.2 million manually tagged words. Its tagset contains 22 POS tags (41 if punctuation marks are included) and 10 more tags that represent additional semantic aspects. We carry out tests without using the 10 additional tags. We use the same training/test partitions used by (dos Santos et al., 2008). Additionally, we created a development set by randomly selecting 5% of the training set sentences. In Table 2, we present detailed information about the Mac-Morpho corpus.

Table 2. Mac-Morpho Corpus for Portuguese POS Tagging.

SET	SENT.	TOKENS	OOSV	OOUV
TRAINING	42,021	959,413	0	4155
DEVELOP.	2,212	48,258	1360	202
TEST	9,141	213,794	9523	1004

4.3. Model Setup

We use the development sets to tune the neural network hyper-parameters. Many different combinations of hyper-parameters can give similarly good results. As usually occurs in SGD training, the learning rate is the hyper-parameter that has the largest impact in the prediction. Therefore, we spent more time tuning the learning rate than tuning other parameters. Nevertheless, learning rates in the range of 0.01 and 0.005 give very similar results, even when using the same number of training epochs. Additionally, we use a learning rate schedule that decreases the learning rate λ according to the training epoch t. The learning rate for epoch t, λ_t, is computed using the equation:

[2] http://www.linguateca.pt/cetenfolha/

[3] http://www.linguateca.pt/cetempublico/

[4] We use the LDC99T42 Treebank release 3 version.

$\lambda_t = \dfrac{\lambda}{t}$. A generalization of this learning rate schedule is presented in (Bengio, 2012).

It is important to note that we use the same set of hyper-parameters for both English and Portuguese. This provides some indication on the robustness of our approach to multiple languages. The number of training epochs is the only difference in terms of training setup for the two languages. The WSJ corpus is trained for five training epochs, while the training with the Mac-Morpho corpus lasts eight training epochs. In Table 3, we show the selected hyper-parameter values.

Table 3. Neural Network Hyper-Parameters

PARAMETER	PARAMETER NAME	VALUE
d^{wrd}	WORD-LEVEL DIM.	100
k^{wrd}	WORD CONT. WINDOW	5
d^{chr}	CHAR. EMBED. DIM.	10
k^{chr}	CHAR. CONT. WINDOW	5
cl_u	CONVOL. UNITS	50
hl_u	HIDDEN UNITS	300
λ	LEARNING RATE	0.0075

In order to assess the effectiveness of the proposed character-level representation of words, we compare the proposed architecture CharWNN with an architecture that uses only word embeddings and additional features instead of character-level embeddings of words. In our experiments, WNN represents a network which is fed with word representations only, i.e, for each word w_n its embedding is $u_n = r^{wrd}$. WNN is essentially Collobert et al.'s (2011) NN architecture. Where indicated, it also includes two additional handcrafted features: capitalization and suffix. The capitalization feature has five possible values: all lowercased, first uppercased, all uppercased, contains an uppercased letter, and all other cases. We use suffix of size two for English and of size three for Portuguese. In our experiments, both capitalization and suffix embeddings have dimension five. We use the same NN hyper-parameters values (when applicable) shown in Table 3.

4.4. Results for POS Tagging of Portuguese Language

In Table 4, we report POS Tagging accuracy (Acc.) results for the Mac-Morpho corpus. The architecture WNN, which does not use character-level embeddings, performs very poorly without the use of the capitalization feature. This happens because we do not have information about capitalization in our word embeddings, since we use lowercased words. When taking into account all words in the test set (column Acc.), CharWNN achieves a slightly better result than the one of WNN with two handcrafted features. Regarding out-of-vocabulary words, we can see in Table 4 that intra-word information is essential to better performance. For the subset of words for which we do not have word embeddings (column Acc. OOUV), the use of character-level information by CharWNN is responsible for an error reduction of 58% when compared to WNN without intra-word information (from 75.40% to 89.74%). These results demonstrate the effectiveness of our convolutional approach to learn valuable character-level information for POS Tagging.

Our *Theano* based implementation of CharWNN takes around 2h30min to complete eight training epochs for the Mac-Morpho corpus. In our experiments, we use 4 threads in a Intel® Xeon® E5-2643 3.30GHz machine.

Table 4. Comparison of different NNs for POS Tagging of the Mac-Morpho Corpus.

SYSTEM	FEATURES	ACC.	ACC. OOSV	ACC. OOUV
CHARWNN	–	**97.47**	92.49	**89.74**
WNN	CAPS+SUF3	97.42	**92.64**	89.64
WNN	CAPS	97.27	90.41	86.35
WNN	SUF3	96.35	85.73	81.67
WNN	–	96.19	83.08	75.40

In Table 5, we compare the result of CharWNN with reported state-of-the-art results for the Mac-Morpho corpus. In (dos Santos et al., 2008), the authors use Entropy Guided Transformation Learning, a learning strategy which combines Decision Trees and Transformation-Based Learning. In (Fernandes, 2012), the authors use Structured Perceptron with Entropy Guided Feature Induction. In both pieces of work, the authors use many handcrafted features, most of them to deal with out-of-vocabulary words. Our system reduces the error of the previously best system by 12.2%. This is a remarkable result, since we train our model from scratch, i.e., without the use of any handcrafted features.

Table 5. Comparison with state-of-the-art systems for the Mac-Morpho corpus.

SYSTEM	ACCURACY
THIS WORK	**97.47**
(FERNANDES, 2012)	97.12
(DOS SANTOS ET AL., 2008)	96.75

4.4.1. PORTUGUESE CHARACTER-LEVEL EMBEDDINGS

We can check the effectiveness of the morphological information encoded in character-level embeddings of words by computing the similarity between the embeddings of rare words and words in the training set. In Table 6, we present four OOSV words and one OOUV word (*"drograsse"*), and their respective most similar words in the training set. The similarity between two words w_i and w_j is computed as the cosine between the two vectors r_i^{wch} and r_j^{wch} (character-level embeddings). We can see in Table 6 that the character-level embeddings are very effective for learning suffixes and prefixes.

In Table 7, we present the same five out-of-vocabulary words and their respective most similar words in the vocabulary extracted from the unlabeled data. In this Table, we use word-level embeddings (r^{wrd}) to compute the similarity. We can see in this case that the words are more semantically related, some of them being synonyms. On the other hand, they are morphologically less similar than the ones in Table 6.

4.5. Results for POS Tagging of English Language

In Table 8, we report POS tagging results of different NN configurations for the WSJ corpus. The behavior of the NNs is quite similar to the one obtained with the Mac-Morpho corpus. Again, the capitalization feature has a much larger impact than the suffix feature. Using the character-level embeddings of words (CharWNN) we again get better results than the ones obtained with the WNN that uses handcrafted features. This demonstrates that our convolutional approach to learning character-level embeddings can be successfully applied to the English language.

Our CharWNN implementation takes around 1 hour to complete five training epochs for the WSJ corpus. Again, we use 4 threads in a Intel® Xeon® E5-2643 3.30GHz machine.

Table 8. Comparison of different NNs for POS Tagging of the WSJ Corpus.

SYSTEM	FEATURES	ACC.	ACC. OOSV	ACC. OOUV
CHARWNN	–	**97.32**	**89.86**	85.48
WNN	CAPS+SUF2	97.21	89.28	**86.89**
WNN	CAPS	97.08	86.08	79.96
WNN	SUF2	96.33	84.16	80.61
WNN	–	96.13	80.68	71.94

Table 9 shows a comparison of our proposed CharWNN results with reported state-of-the-art results for the WSJ corpus. In (Søgaard, 2011), the author uses a semi-supervised version of the condensed nearest neighbor algorithm. During the learning process, his strategy uses SVMTool, which makes use of a rich feature set to solve the POS tagging task. In (Manning, 2011), the author uses a set of additional features and distributional similarity classes to improve Toutanova et al.'s (2003) POS Tagger, which is based on a Cyclic Dependency Network. This POS tagger also has a rich feature set. Collobert et al.'s (2011) POS tagger is essentially the same as our WNN (without character-level embeddings) using capitalization and suffix features. The result of our system for the WSJ corpus is similar to the ones of (Manning, 2011) and (Collobert et al., 2011), and very competitive with the result of (Søgaard, 2011). Again, this is a remarkable result, since differently from the other systems, our approach does not use any handcrafted feature.

Table 9. Comparison with state-of-the-art systems for the WSJ corpus.

SYSTEM	ACCURACY
(SØGAARD, 2011)	**97.50**
THIS WORK	97.32
(COLLOBERT ET AL., 2011)	97.29
(MANNING, 2011)	97.28

4.5.1. ENGLISH CHARACTER-LEVEL EMBEDDINGS

For the WSJ corpus, we also compute the similarity between the character-level embeddings of rare words and words in the training set. In Table 10, we present six OOSV words, and their respective most similar words in the training set. Again, the similarity between two words is computed as the cosine between the two character-level embeddings of the words. From Table 10, we can also see that the character-level embedding approach is effective at learning suffixes, prefixes and even word shapes from the WSJ corpus.

Table 11 shows the same six OOSV words and their respective most similar words in the vocabulary extracted from the unlabeled data. The similarity is computed using the word-level embeddings (r^{wrd}). Similarly to the Portuguese language case, we can easily note that in Table 10 the retrieved words are morphologically more related than the ones in Table 11 Notice that due to our processing of the unlabeled data, we have to replace all numerical digits by 0 in order to look for the word-level embeddings of *83-year-old* and *0.0055*.

5. Conclusions

In this work we present a new deep neural network architecture that jointly uses word-level and character-level representations to perform natural language processing. The main contributions of the paper are: (1) the idea of using convolutional neural networks to extract character-level features and jointly use them with word-level features; (2)

Table 6. Most similar words using character-level embeddings learned with Mac-Morpho Corpus.

GRADAÇÕES	CLANDESTINAMENTE	REVOGAÇÃO	DESLUMBRAMENTO	DROGASSE
TRADIÇÕES	GLOBALMENTE	RENOVAÇÃO	DESPOJAMENTO	DIVULGASSE
TRADUÇÕES	CRONOMETRICAMENTE	REAVALIAÇÃO	DESMANTELAMENTO	DIRIGISSE
ADAPTAÇÕES	CONDICIONALMENTE	REVITALIZAÇÃO	DESREGRAMENTO	JOGASSE
INTRADUÇÕES	APARENTEMENTE	REFIGURAÇÃO	DESENRAIZAMENTO	PROCURASSE
REDAÇÕES	FINANCEIRAMENTE	RECOLOCAÇÃO	DESMATAMENTO	JULGASSE

Table 7. Most similar words using word-level embeddings learned using unlabeled Portuguese texts.

GRADAÇÕES	CLANDESTINAMENTE	REVOGAÇÃO	DESLUMBRAMENTO	DROGASSE
TONALIDADES	ILEGALMENTE	ANULAÇÃO	ASSOMBRO	–
MODULAÇÕES	ALI	PROMULGAÇÃO	EXOTISMO	–
CARACTERIZAÇÕES	ATAMBUA	CADUCIDADE	ENFADO	–
NUANÇAS	BRAZZAVILLE	INCONSTITUCIONALIDADE	ENCANTAMENTO	–
COLORAÇÕES	VOLUNTARIAMENTE	NULIDADE	FASCÍNIO	–

Table 10. Most similar words using character-level embeddings learned with WSJ Corpus.

INCONSIDERABLE	83-YEAR-OLD	SHEEP-LIKE	DOMESTICALLY	UNSTEADINESS	0.0055
INCONCEIVABLE	43-YEAR-OLD	ROCKET-LIKE	FINANCIALLY	UNEASINESS	0.0085
INDISTINGUISHABLE	63-YEAR-OLD	FERN-LIKE	ESSENTIALLY	UNHAPPINESS	0.0075
INNUMERABLE	73-YEAR-OLD	SLIVER-LIKE	GENERALLY	UNPLEASANTNESS	0.0015
INCOMPATIBLE	49-YEAR-OLD	BUSINESS-LIKE	IRONICALLY	BUSINESS	0.0040
INCOMPREHENSIBLE	53-YEAR-OLD	WAR-LIKE	SPECIALLY	UNWILLINGNESS	0.025

Table 11. Most similar words using word-level embeddings learned using unlabeled English texts.

INCONSIDERABLE	00-YEAR-OLD	SHEEP-LIKE	DOMESTICALLY	UNSTEADINESS	0.0000
INSIGNIFICANT	SEVENTEEN-YEAR-OLD	BURROWER	WORLDWIDE	PARESTHESIA	0.00000
INORDINATE	SIXTEEN-YEAR-OLD	CRUSTACEAN-LIKE	000,000,000	HYPERSALIVATION	0.000
ASSUREDLY	FOURTEEN-YEAR-OLD	TROLL-LIKE	00,000,000	DROWSINESS	0.000000
UNDESERVED	NINETEEN-YEAR-OLD	SCORPION-LIKE	SALES	DIPLOPIA	±
SCRUPLE	FIFTEEN-YEAR-OLD	UROHIDROSIS	RETAILS	BREATHLESSNESS	-0.00

the demonstration that it is feasible to train state-of-the-art POS taggers for different languages using the same model, with the same hyper-parameters, and without any handcrafted features; (3) the definition of a new state-of-the-art result for Portuguese POS tagging on the Mac-Morpho corpus.

A weak point of the proposed approach is the introduction of additional hyper-parameters to be tuned. However, we argue that it is generally preferable to tune hyper-parameters than to handcraft features, since the former can be automated.

As future work, we would like to analyse in more detail the interrelation between the two representations: word-level and character-level. Another possibility for future work consists in applying the proposed strategy to other natural language processing tasks such as text chunking and named entity recognition.

Acknowledgments

The authors would like to thank Janaina Cruz Pereira for drawing the nice figure that illustrates our convolutional approach to character-level feature extraction.

References

Alexandrescu, Andrei and Kirchhoff, Katrin. Factored neural language models. In *Proceedings of the Human Language Technology Conference of the NAACL*, pp. 1–4, New York City, USA, June 2006.

Aluísio, Sandra M., Pelizzoni, Jorge Marques, Marchi, Ana Raquel, de Oliveira, Lucélia, Manenti, Regiana, and Marquiafável, Vanessa. An account of the challenge of tagging a reference corpus for brazilian portuguese. In *PROPOR*, pp. 110–117, 2003.

Bengio, Yoshua. Practical recommendations for gradient-based training of deep architectures. In Montavon, Grégoire, Orr, GenevièveB., and Müller, Klaus-Robert (eds.), *Neural Networks: Tricks of the Trade*, volume 7700 of *Lecture Notes in Computer Science*, pp. 437–478. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-35288-1.

Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Janvin, Christian. A neural probabilistic language model. *Journal of Machine Learning Reseach*, 3:1137–1155, 2003. ISSN 1532-4435.

Bergstra, James, Breuleux, Olivier, Bastien, Frédéric, Lamblin, Pascal, Pascanu, Razvan, Desjardins, Guillaume, Turian, Joseph, Warde-Farley, David, and Bengio, Yoshua. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, 2010.

Collobert, R. Deep learning for efficient discriminative parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS)*, pp. 224–232, 2011.

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

dos Santos, Cícero Nogueira, Milidiú, Ruy Luiz, and Rentería, Raúl P. Portuguese part-of-speech tagging using entropy guided transformation learning. In *Proceedings of PROPOR 2008*, pp. 143–152, Aveiro, Portugal, 2008.

Fernandes, Eraldo Luís Rezende. *Entropy Guided Feature Generation for Structure Learning*. PhD thesis, Pontifícia Universidade Católica do Rio de Janeiro, 2012.

Lazaridou, Angeliki, Marelli, Marco, Zamparelli, Roberto, and Baroni, Marco. Compositional–ly derived representations of morphologically complex words in distributional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1517–1526, 2013.

Lecun, Yann, Bottou, Lon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, pp. 2278–2324, 1998.

Luong, Minh-Thang, Socher, Richard, and Manning, Christopher D. Better word representations with recursive neural networks for morphology. In *Proceedings of the Conference on Computational Natural Language Learning*, Sofia, Bulgaria, 2013.

Manning, Christopher D. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing*, CICLing'11, pp. 171–189, 2011.

Marcus, Mitchell P., Marcinkiewicz, Mary Ann, and Santorini, Beatrice. Building a large annotated corpus of english: The penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations*, 2013.

Socher, Richard, Bauer, John, Manning, Christopher D., and Ng, Andrew Y. Parsing with compositional vector grammars. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. 2013.

Søgaard, Anders. Semisupervised condensed nearest neighbor for part-of-speech tagging. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers*, pp. 48–52, 2011.

Toutanova, Kristina, Klein, Dan, Manning, Christopher D., and Singer, Yoram. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pp. 173–180, 2003.

Viterbi, A. J. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2):260–269, April 1967.

Waibel, A., Hanazawa, T., Hinton, G., Shikano, K., and Lang, K. J. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(3):328–339, 1989.

Zheng, Xiaoqing, Chen, Hanyang, and Xu, Tianyu. Deep learning for chinese word segmentation and pos tagging. In *Proceedings of the Conference on Empirical Methods in NLP*, pp. 647–657, 2013.