Portuguese Named Entity Recognition Using LSTM-CRF

Pedro Vitor Quinta de Castro(✉), Nádia Félix Felipe da Silva, and Anderson da Silva Soares

Universidade Federal de Goiás, Goiânia, GO 74690-900, Brazil
{I.pedrovitorquinta,II.nadia,III.anderson}@inf.ufg.br

Abstract. Named Entity Recognition is a challenging Natural Language Processing task for a language as rich as Portuguese. For this task, a Deep Learning architecture based on bidirectional Long Short-Term Memory with Conditional Random Fields has shown state-of-the-art performance for English, Spanish, Dutch and German languages. In this work, we evaluate this architecture and perform the tuning of hyper-parameters for Portuguese corpora. The results achieve state-of-the-art performance using the optimal values for them, improving the results obtained for Portuguese language to up to 5 points in the F1 score.

Keywords: Natural Language Processing
Named Entity Recognition · Deep learning · Neural networks
Portuguese language

1 Introduction

Hundreds of millions of unstructured textual information are exchanged every minute [1]. Named Entity Recognition (NER) is an important Natural Language Processing (NLP) task which focus on extracting and classifying named entities from this unstructured textual information, making them interpretable and accessible to different communication channels. The NER task can be approached either by using a rule/pattern based system, or by a machine learning method [2].

As far as we know, few works have focused on neural network architectures with evaluations performed in Portuguese language [3], while several studies have been done for English Language [4–8]. In this paper, we study the LSTM-CRF neural architecture proposed by [4] in the Portuguese language context. The architecture combines a character-based word representation model with word embeddings. This combination is fed into a bidirectional Long Short-Term

Thanks to Data-H Data Science and Artificial Intelligence (www.datah.com.br) and Aviso Urgente (https://avisourgente.com.br) for the financial support, and to Cicero Nogueira dos Santos for kindly sharing insights regarding the HAREM corpora.

A. Villavicencio et al. (Eds.): PROPOR 2018, LNAI 11122, pp. 83–92, 2018.
https://doi.org/10.1007/978-3-319-99722-3_9

Memory (LSTM) network, which is finally connected to a Conditional Random Fields (CRF) layer to perform sequential classification.

As main contributions of this paper, we point out: (i) Since Portuguese is such a morphologically rich language, we intend not only to evaluate how LSTM-CRF performs in Portuguese corpora, but also to perform the hyperparameters tuning in order to achieve the best results for the language in study. (ii) We present the first comparative study about the word embeddings with LSTM based methods for Portuguese NER. We experimented with four different pre-trained word embeddings from [9]—FastText [10,11], Glove [12], Wang2Vec [13] and Word2Vec [14].

2 Related Work

Classical approaches for NER are dependent on handcrafted features, which have language-specific values and are cumbersome to maintain. For instance, [15] created a model based on CRF and used 17 different features for each word in the training corpus, such as part-of-speech (POS) tags, capitalization and the word itself, considering a context window of size 2. Deep learning approaches provide an alternative to these classical approaches.

One of the main advantages of using a deep learning approach that uses word and character level embeddings as input for model training is the independence of language specific features, since the features used are the ones that are automatically learned by using these two types of embeddings. Hence it is possible to use the same network to train models for different languages, as long as it is provided an annotated corpus for each language, as well as the pre-trained word embeddings. [3] used the same network to train models for Portuguese and Spanish, and [4] trained models for English, Dutch, German and Spanish.

Word-level embeddings are multidimensional vectors that represent features automatically learned by unsupervised training. These features represent morphological, syntactic and semantic information about the words. The unsupervised learning of these features is tipically performed on massive corpora, such as Wikipedia[1] and news archives. Using such a large amount of text allows the understanding of contexts on which certain types of words tend to occur [14]. The use of character-level embeddings is important because they allow to capture orthographic features, such as prefixes, suffixes and letter case, the latter being essential for identifying proper names in a text. These orthographic features also represent the importance of the characters used in the language in study. In Portuguese, for example, characters such as "ç" and accented vowels are quite usual. In addition, character-level embeddings are specially important for morphologically rich languages, such as Portuguese, because they provide additional intra-word and shape information to the features learned.

Dos Santos and Guimarães [3], one of the few works to apply neural networks to Portuguese NER, introduced the CharWNN architecture, which uses Convolutional Neural Networks (CNN) to learn character-level features, combined with

[1] https://www.wikipedia.org/.

pre-trained word-level embeddings to perform sequential classification. Lample et al. [4] used a deep learning approach and outperformed several methods that used handcrafted features and external resources, in four different languages. Because of that, more attention will be paid on their architecture in Sect. 3.

The approaches to Portuguese NER are still in a level way lower than languages such as English or Spanish. While the best result for Enghish corpus present 90.94% for the F1 score, the best reported result for Portuguese has only 71.23% in the same score. It is difficult to make comparisons between Portuguese NER due to the absence of standardized benchmarks [16]. Table 1 shows different settings by the authors to achieve their results.

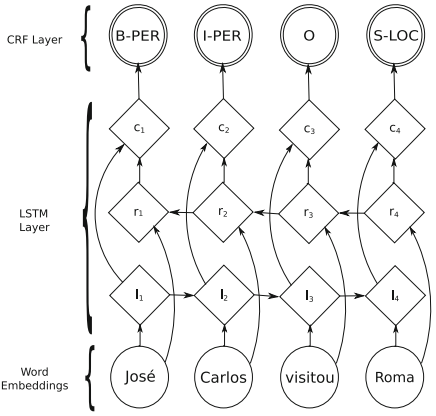
Table 1. Reported results for different languages.

Author	Language	Corpora	Evaluation Script	Precision	Recall	F1
Amaral et al. [15]	Portuguese	train: HAREM I	SAHARA	83.48%*	44.35%*	57.92%*
		test: HAREM II				
Santos et al. [3]	Portuguese	train: HAREM I	CoNLL	73.98%**	68.68%**	71.23%**
		test: miniHAREM		67.16%*	63.74%*	65.41%*
	Spanish	SPA CoNLL-2002 Corpus		82.21%	82.21%	82.21%
Lample et al. [4]	English	CoNLL-2003 Corpus	CoNLL	*not shown*	*not shown*	90.94%
	Spanish	SPA CoNLL-2002 Corpus		*not shown*	*not shown*	85.75%

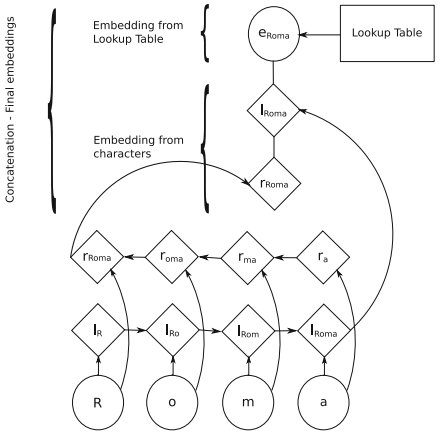
* Indicates the results for predicting all 10 categories from HAREM.
** Indicates the results for predicting 5 selected categories from HAREM.

3 LSTM-CRF Architecture

The LSTM-CRF architecture proposed by [4], as depicted in Fig. 1, is based on two intuitions: (i) Assigning tags for tokens in a text is based on contextual information, i.e., depends on other words and how they are related; (ii) In order to determine if a token is a name, it is important to consider both orthographic and distributional evidences. Orthographic evidences would be related to the shape of the word (the features that determine the appearance of the word), and distributional evidences would be related to the location in which the word



(a) LSTM-CRF architecture. Word embeddings are fed into to a bidirectional LSTM. li represents the word i and its left context, ri represents the word i and its right context. The two vectors are concatenated, yielding a representation of the word i in its context, ci.



(b) The character embeddings of the word "Roma" are fed into a bidirectional LSTM. Their last outputs are concatenated to an embedding from a lookup table to obtain a representation for this word.

Fig. 1. Word and character level embeddings in the LSTM-CRF architecture. Adapted from [4].

tends to occur (the features that are related to neighboring words in sentences and in the corpus).

Recurrent neural networks (RNNs) represent a class of deep neural networks which are more suitable to handle sequential data, such as texts. Plain feed-forward networks, such as multi-layer perceptrons (MLP), and even CNNs, are limited in the sense that they take a fixed-size vector as input and produce a fixed-sized vector as output. RNNs, on the other hand, support sequences of vectors, being able to take inputs of variable sizes, also producing outputs of variable sizes. In theory, RNNs were conceived to capture long-term dependencies in large sequences, but, in practice, this was not possible due to the occurrence of vanishing and exploding gradient issues [17]. In order to overcome this limitation, [18] proposed the LSTM network, a type of RNN network in which the hidden units are enhanced with three multiplicative gates that control how the information is forgotten and propagated while flowing through each time step. These three gates are: update gate, forget gate and output gate. Equations (1) to (6) show the formulas used to update an LSTM unit at time t.

$$\mathbf{i}_t = \sigma(\mathbf{W}_i \mathbf{h}_{t-1} + \mathbf{U}_i \mathbf{x}_t + \mathbf{b}_i) \tag{1}$$

$$\mathbf{f}_t = \sigma(\mathbf{W}_f \mathbf{h}_{t-1} + \mathbf{U}_f \mathbf{x}_t + \mathbf{b}_f) \tag{2}$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{W}_c \mathbf{h}_{t-1} + \mathbf{U}_c \mathbf{x}_t + \mathbf{b}_c) \tag{3}$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t \tag{4}$$

$$\mathbf{o}_t = \sigma(\mathbf{W}_o \mathbf{h}_{t-1} + \mathbf{U}_o \mathbf{x}_t + \mathbf{b}_o) \tag{5}$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \tag{6}$$

where $\mathbf{i}_t$ represents the update gate, $\mathbf{f}_t$ represents the forget gate and $\mathbf{o}_t$ represents the output gate, all three in a given time t. $\mathbf{c}_t$ and $\tilde{\mathbf{c}}_t$ represent the cell state and the candidate cell state of the LSTM unit, in a given time t. $\mathbf{W}$ stands for the weight matrices of the hidden state $\mathbf{h}$, $\mathbf{U}$ stands for the weight matrices of the input $\mathbf{x}$ and $\mathbf{b}$ stands for the bias vectors. σ represents element-wise sigmoid function and $\odot$ represents element-wise product.

Considering an input sentence represented by $\{x_1, x_2, x_3, ..., x_n\}$, with n words encoded as a d-dimensional vector, the bidirectional LSTM unit would calculate a hidden state $\overrightarrow{\mathbf{h}_t}$ and a hidden state $\overleftarrow{\mathbf{h}_t}$ for the left and right contexts of the sentence, at every word i, as depicted by Fig. 1a. The bidirectional aspect of the LSTM can be implemented by using a second LSTM unit that computes the right context by reading the same sentence in reverse. When the two hidden states are computed, they are concatenated into a single representation, $h_t = [\overrightarrow{\mathbf{h}_t}; \overleftarrow{\mathbf{h}_t}]$.

Figure 1b shows how word embeddings are generated in the LSTM-CRF architecture. The character lookup table is initialized using a random uniform distribution, providing an embedding for every character found in the corpus. For each word in each input sentence, every character from the word is processed in direct and reverse order, using the embedding of the character from the lookup table and feeding it into a bidirectional LSTM unit. For each word, the character level embedding is resulting from the concatenation of the forward and backward representations from the LSTM unit, and this final character embedding is then concatenated with the word level embedding, obtained from the word embeddings used for training.

As for the sequential classification of the named entities, CRF is the algorithm used to predict the sequence of labels. It is a type of statistical modeling method which is often applied in pattern recognition and machine learning. When labeling sequences of words with CRF, the model provides a correlation understanding between words and labels which occur close to each other, i.e., it uses the label from surrounding words in order to determine the label of a given target word. As an example of NER labeling using the IOB2 tagging scheme [19], a word labeled with I-PESSOA could not follow a word labeled with O[2].

[2] This is because ***I*** indicates an internal token in the named entity, and ***O*** indicates a non-entity token, which means that anything after it would be the starting token of an entity or another non-entity token. Since the first token of a named entity starts with ***B***, according to the IOB scheme, it is not possible that an internal entity token follows a non-entity token.

4 Experimental Evaluation

4.1 Datasets

HAREM [20] is considered to be the main reference of corpora for the Portuguese NER task. It is a joint evaluation in the area of NER in Portuguese, intended to regulate and evaluate the success in the identification and classification of proper names in the Portuguese language. HAREM had two editions: HAREM I and HAREM II. MiniHAREM was an intermediate event that repeated the same evaluation as HAREM I. Each of these events produced a gold standard collection for the evaluation of NER systems. The corpora are annotated with ten named entity categories: Person (PESSOA), Organization (ORGANIZACAO), Location (LOCAL), Value (VALOR), Time (TEMPO), Abstraction (ABSTRACCAO), Title (OBRA), Event (ACONTECIMENTO), Thing (COISA) and Other (OUTRO). [3] experimented in two scenarios: total and selective. For the *total* one, all ten categories of HAREM are considered, while only five are considered in the *selective* scenario: Person, Organization, Location, Time and Value. We compare our results with the ones obtained by [3], for both total and selective scenarios. As for model evaluation, we use the same CoNLL script from [3,4]. We use the same HAREM corpora for training and testing datasets that was used by [3]. The gold standard collection from HAREM I was used as the training set, and the gold standard collection from MiniHAREM was used as the test set.

4.2 Parameterization, Training and Experimental Setup

For tagging schemes, we experimented with two different IOB tagging schemes: IOB2 [19] and IOBES. IOBES differs from IOB2 because it labels single-token entities with the ***S*** prefix, and also labels the final tokens from multi-token entities with the ***E*** prefix. Regarding word embeddings, we experimented with four different pre-trained word embeddings from [9]: FastText [10,11], Glove [12], Wang2Vec [13] and Word2Vec [14], all of them with dimension 100. As mentioned in [3,4], we picked the FastText, Wang2Vec and Word2Vec embeddings that were trained with the skip-gram model. The training process of these word-embeddings are described in [21].

Besides the character and word level embeddings, we also experimented with a capitalization feature. This feature is a representation of the word capitalization: 0 if all characters are lowercase, 1 if all characters are uppercase, 2 if the first letter is uppercase and 3 if a letter besides the first is uppercase. In addition to the capitalization feature, we also experimented normalizing the words before producing the dictionaries that are used to perform the word-embedding lookup. This normalization is nothing more than converting the word to its lowercase form, and does not affect the data structures used to learn character-level features.

This architecture uses two bidirectional LSTMs, one for learning the features from the character-level embeddings, and one for the word-level representations. Despite [4] observing that the increase of the number of hidden units for each

LSTM did not have a significant impact on the model's performance, we have experimented with two different dimensions for each. [4] used 25 hidden units for each character LSTM, the forward and the backward, and we have experimented with 25 and 50 hidden units. For the word LSTMs, [4] used 100 hidden units, and we have experimented with 100 and 200 units.

The model is trained using the backpropagation algorithm, and optimization is done using stochastic gradient descent (SGD), with a learning rate of 0.01 and a gradient clipping of 5.0. [4] experienced with other optimization methods, but none performed better than SGD with gradient clipping. In order to determine the best set of parameter values to be used in our experiment, from the ones mentioned in this section, first we picked 6 parameters: tagging scheme, word embedding, capitalization feature, word normalization and dimension of character and word LSTM units. From all the possible values for each of these parameters, there are 128 different combinations to be evaluated. We ran each of the combinations 10 times, in the selective scenario, with only 5 epochs, which we considered as sufficient for determining the best set of parameters. Once we determined the best set of parameters, we trained the model for 100 epochs, using the parameter values obtained from the previous step. We also trained with these parameters 10 times, in order to estimate an average of the model's performance.

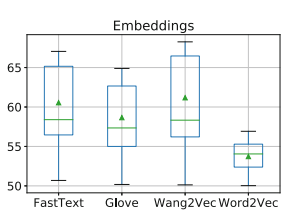
4.3 Results

Figure 2 contains boxplots with the comparisons between each set of parameter values assessed in our trainings. We realized that the ones that had the greatest impact in our training were embedding type and word normalization, while the different values assessed for tagging scheme, capitalization and hidden units dimensions did not have a considerable impact in the results. Figure 2a indicates that Wang2Vec embeddings outperformed Word2Vec, Glove and FastText, with a mean F1 score of 61.17. FastText, which is ranked second, had a mean F1 score of 60.54, while Glove scored 58.65 and Word2Vec scored 53.72.

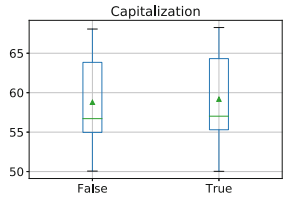
The normalization of words was the most significant parameter that we experimented with. Keeping the words as they are, without normalization, provided a mean score of 55.78, while normalizing the words to lower case form gave a mean score of 64.47. We realized that the normalization had such a great impact because of the pre-trained word embeddings used for NER training. All words contained in the embeddings were only presented in their lowercase form, so whenever a lookup was performed in the embeddings table, if the word started with an uppercase letter, it would not be found, and a random vector would be initialized to it. So, performing the normalization prior to the lookup enforces the use of the proper word vectors for NER training.

From the results obtained after running all combinations of parameters values, we verified that the optimal combination for training a final model would be: Wang2Vec as pre-trained word embeddings, IOBES tagging scheme, normalization of words, use of capitalization features, 25 hidden units for the character

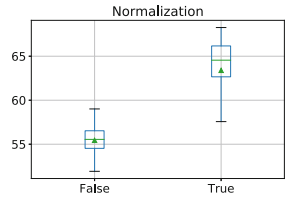
LSTM and 100 hidden units for word LSTM. Despite not having a considerable difference between the results obtained for the different values of tagging schemes, capitalization features and hidden units dimension for both character and word LSTMs, the choice of values for these parameters were due to their results being slightly higher than the other evaluated options. Table 2 displays the comparison between the results from [3] and the ones obtained in our final training, using the tuned hyperparameters. LSTM-CRF outperformed CharWNN in both total and selective scenarios, improving the F1 score in both total and selective scenarios by 5 points.



(a) Results obtained for each type of pre-trained word embeddings evaluated.

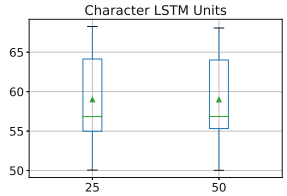


(b) Results for capitalization features, if the capitalization values were used as features in the training or not.

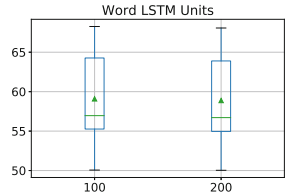


(c) Results for words normalization, if the words were lowercased before looking up their embeddings or not.

(d) Results obtained for IOB2 and IOBES tagging schemes.



(e) Results obtained for 25 and 50 units for character LSTM units.



(f) Results obtained for 100 and 200 units for word LSTM units.

Fig. 2. Results obtained for each set of parameters values evaluated. Each boxplot depicts the data related to the F1 score obtained for each of the 1280 executions, grouped by the set of parameter values displayed in each of them. The green triangles represent the arithmetic means of the F1 scores obtained. (Color figure online)

Table 2. Comparison with the state-of-the-art for the HAREM I corpus

Architecture	Total scenario			Selective scenario		
	Precision	Recall	F1	Precision	Recall	F1
CharWNN	67.16%	63.74%	65.41%	73.98%	68.68%	71.23%
LSTM-CRF	**72.78%**	**68.03%**	**70.33%**	**78.26%**	**74.39%**	**76.27%**

5 Conclusions

In this paper, we experimented different scenarios of Named Entity Recognition with Portuguese corpora, using a deep neural network architecture based on bidirectional LSTM and Conditional Random Fields. We evaluated different combinations of hyperparameters for training, and verified the optimal values for the parameters that had a greatest impact in the performance of the model: word embeddings model and word normalization. We achieve state-of-the-art performance for Portuguese NER task using the optimal values for these parameters.

The word embedding model that had the best performance in our experiments was Wang2Vec, which is a method derived from modifications in Word2Vec. The purpose of these modifications was to improve the capture of the syntactic behavior of words, taking into consideration the order in which they appear in the texts. We verify that this improves the performance of a sequence labeling task such as NER. We also verify that normalizing words before looking up their embeddings greatly improves the performance of the model, opposed to looking up the embeddings according to the letter case they are in the text.

For future work, we will experiment on the effects of applying this NER model in texts belonging to a specific domain, instead of a general purpose corpora such as the ones based on news and wikipedia articles.

References

1. How Much Data is Created on the Internet Each Day? https://blog.microfocus.com/how-much-data-is-created-on-the-internet-each-day/. Accessed 19 Mar 2018
2. Maynard, D., Bontcheva, K., Augenstein, I.: Natural Language Processing for the Semantic Web, 1st edn. Morgan and Claypool, San Rafael (2017)
3. dos Santos, C., Guimarães, V.: Boosting named entity recognition with neural character embeddings. arXiv preprint arXiv:1505.05008 (2015)
4. Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., Dyer, C.: Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360 (2016)
5. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. arXiv preprint arxiv:1103.0398 (2011)
6. Nothman, J., Ringland, N., Radford, W., Murphy, T., Curran, J.R.: Learning multilingual named entity recognition from Wikipedia. In: Artificial Intelligence, vol. 194, pp. 151–175. Elsevier Science Publishers Ltd., Essex (2013). https://doi.org/10.1016/j.artint.2012.03.006
7. Chiu, J., Nichols, E.: Named entity recognition with bidirectional LSTM-CNNs. arXiv preprint arXiv:1511.08308 (2015)
8. Ma, X., Hovy, E.: End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. arXiv preprint arXiv:1603.01354 (2016)
9. Repositório de Word Embeddings do NILC. http://www.nilc.icmc.usp.br/nilc/index.php/repositorio-de-word-embeddings-do-nilc. Accessed 30 Mar 2018
10. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. arXiv preprint arXiv:1607.04606 (2016)

11. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
12. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP-2014), vol. 12, pp. 1532–1543 (2014)
13. Ling, W., Dyer, C., Black, A., Trancoso, I.: Two/too simple adaptations of word2vec for syntax problems. In: Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics (2015)
14. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arxiv:1301.3781 (2013)
15. Amaral, D., Vieira, R.: NERP-CRF: a tool for the named entity recognition using conditional random fields. In: Linguamática, vol. 6, pp. 41–49 (2014)
16. Marrero, M., Urbano, J., Sánchez-Cuadrado, S., Morato, J., Gómez-Berbís, J.: Named entity recognition: fallacies, challenges and opportunities. Comput. Stand. Interfaces **35**, 482–489 (2013)
17. Bengio, Y., Simard, P., Frasconi, P.: Learning long-term dependencies with gradient descent is difficult. IEEE Trans. Neural Netw. **5**, 157–166 (1994). https://doi.org/10.1109/72.279181
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997). https://doi.org/10.1162/neco.1997.9.8.1735
19. Sang, E., Veenstra, J.: Representing text chunks. arXiv preprint arxiv:cs/9907006 (1999)
20. HAREM: Reconhecimento de entidades mencionadas em português. https://www.linguateca.pt/HAREM/. Accessed 21 Mar 2018
21. Hartmann, N., Fonseca, E., Shulby, C., Treviso, M., Rodrigues, J., Aluisio, S.: Portuguese word embeddings: evaluating on word analogies and natural language tasks. arXiv preprint arXiv:1708.06025 (2017)