IberSPEECH 2018
21-23 November 2018, Barcelona, Spain

Wide Residual Networks 1D for Automatic Text Punctuation

Jorge Llombart, Antonio Miguel, Alfonso Ortega, Eduardo Lleida

ViVoLAB, Aragón Institute for Engineering Research (I3A), University of Zaragoza, Spain
jllombg@unizar.es, amiguel@unizar.es, ortega@unizar.es, lleida@unizar.es

Abstract

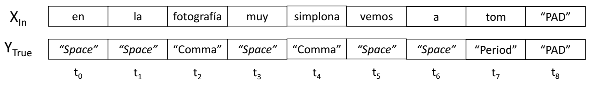
Documentation and analysis of multimedia resources usually requires a large pipeline with many stages. It is common to obtain texts without punctuation at some point, although later steps might need some accurate punctuation, like the ones related to natural language processing. This paper is focused on the task of recovering pause punctuation from a text without prosodic or acoustic information. We propose the use of Wide Residual Networks to predict which words should have a comma or stop from a text with removed punctuation. Wide Residual Networks are a well-known technique in image processing, but they are not commonly used in other areas as speech or natural language processing. We propose the use of Wide residual networks because they show great stability and the ability to work with long and short contextual dependencies in deep structures. Unlike for image processing, we will use 1-Dimensional convolutions because in text processing we only focus on the temporal dimension. Moreover, this architecture allows us to work with past and future context. This paper compares this architecture with Long-Short Term Memory cells which are used in this task and also combine the two architectures to get better results than each of them separately.

Index Terms: Text Punctuation, Wide Residual Network, Recurrent Neural Networks, Natural Language Processing

1. Introduction

Nowadays, there are a great amount of multimedia resources from which we want to extract more and more information. This usually requires a large pipeline with many stages where, at some point, it is needed to have accurate punctuation to accomplish later steps like machine translation, summarization, sentiment analysis, etc. This task is usually performed after some automatic speech recognition system, and it usually takes some prosodic information like pauses time. Some systems are based on a capitalization and punctuation system [1], on Long-Short Term Memory cells (*LSTM*)[2] or on Bidirectional Long-Short Term Memory cells (*BLSTM*) [3], but not always it is possible to access to this features. Other works do not use prosodic features. Some uses Conditional Random Fields (*CRF*) [4], or in a combination with *LSTM* [5]. There are approaches with Convolutional Neural Networks (*CNN*) [6], or distilling from an ensemble of different neural networks [7]. There are also works focused on Character-level Neural Networks [8], where they also use architectures with Convolutional Highway, Residual networks and *BLSTM*.

In image classification, convolutional neural networks have achieved very good results. In those tasks convolutional networks have gain depth, even hundreds of layers. This was possible with Residual Networks [9]. They use very deep and thin structures, but they suppose a great amount of computation. Wide Residual networks (*WRN*) were designed to obtain better results with less depth and, therefore with less computation, increasing the width [10, 11].



X_{In}	en	la	fotografía	muy	simplona	vemos	a	tom	“PAD”
Y_{True}	*“Space”*	*“Space”*	“Comma”	*“Space”*	“Comma”	*“Space”*	*“Space”*	“Period”	“PAD”
	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8

Figure 1: *Input sequence and its tag correspondence for an input sequence of $T = 9$ when the text only has 8 words.*

Wide Residual networks are also used in automatic speech recognition obtaining good results [12, 13]. In those architectures, Wide Residual Networks use two-dimensional convolutions, considering the cepstral input as an image, but in this study we consider that in speech and natural language processing the important dimension is the temporal dimension, and correlation along other dimensions is lower, so we use convolutions 1D.

This paper is divided in four sections. The first is the introduction. In the second section we describe the punctuation task and the neural network models used to perform this task. In section three we explain how the experiments are done. And in the last section we present some conclusions and future work.

2. Text punctuation modelling

In this paper, we will focus on recovering punctuation from Spanish texts where punctuation is missing. Our experiments are designed to recover periods and commas, and we simulate the task by deleting them from a collection of texts. In Spanish there are some different ways to express the end of a sentence: closing exclamation, closing interrogation, full stop, period, etc. In order to simplify the classification we consider that all kind of end of sentence is marked as period. So the set of labels in our experiments consists on $L = \{"Space", "Period", "Comma", "Pad"\}$ where we will label each input word of our experiments as:

- word without punctuation mark: ”Space”
- word followed by any kind of period: ”Period”
- word followed by comma: ”Comma”
- padding: ”PAD”.

This experiment uses a fixed dictionary $O = \{0, ..., D+1\}$, with D the vocabulary size plus two extra symbols. There are two special words, first one is $o_0 = "PAD"$ which is reserved to fill input sequences when it is necessary, and the second is $o_1 = "UNK"$, which is reserved for those words not included in O, usually called out of vocabulary words.

Text punctuation is a classification task in which we want to assign a label from L to each element of an input sequence $X = \{x_0, ..., x_t, ..., x_T\}$ with $t \in T$ the maximum number of elements in the input sequence. Each x_t corresponds with one word $o_d \in O$. Our Neural Network provide as output a sequence of labels $Y = \{y_0, ..., y_t, ..., y_T\}, \forall y_t \in L$. In Figure 1 we represent an input vector X_{in} with the training label vector Y_{True} that corresponds with it.

2.1. Wide Residual Networks 1D

WRNs are usually used in image tasks where two spatial dimensions are the context to work. In automatic speech recognition we can use spectrograms as an image to adopt this two-dimensional schema [12, 13]. However, in text processing, we work only with temporal dimension because word relations are only meaningful along this dimension. *WRN* is a structure based on 2D-convolutional networks. In order to adapt this structure to our environment, we will work with 1D-convolutional layers. In image context, convolutional layers are characterized by the number of output channels, C, but in text context, we consider channels as a temporal sequence of vectors, where each element of this vectors is one of the C filters response for this sequence moment.

A *WRN* consists on several blocks, that in this paper we will call Wide Residual Block (*WRB*). In each of this *WRB*, we increase the number of channels in the output. We will widen the convolutional output. These *WRBs* are also constructed by several Residual Block (*RB*). This Blocks are the basic elements of the *WRNs*, and them have two paths. The first path consists on two convolutional layers with batch normalization [14] and *ReLU* non-linearity [15]. The second path is a residual connection between block input and output. In this work we use a sum of the block input and the convolutional path as residual connection. In the case where the number of channels of the two paths are different, we adjust the number of channels with a convolutional layer in the residual path. In Figure 2a we show the complete structure of the *WRNs* that we use in this work. We will characterize the *WRB* by its widen factor, which is the factor we apply to the number of channels at block input to gain width. In Figure 2a, we describe convolutional layers by its kernel dimension. This kernel dimension is the number of words that the convolutional filter takes to compute each element in the output. In this work the kernel dimension of the first convolution layer is 5. Convolutional layers in *WRB* use a kernel of dimension 3. There is a special case when we use a convolutional layer with kernel dimension of one. This kind of convolutional layer sometimes is called 1 x1 convolution, position independent convolution or position wise fully connected. It is like a fully connected layer with all its weights shared through time. In other words, each element of the input is evaluated with same weights along all sequence. In this layers we also describe input dimension and output dimension of each sequence element as $[input\,dimension \times output\,dimension]$.

Usually in classification we adopt the many-to-one structure where we use an input sequence to classify the central element of the sequence. In convolutional networks for classification, usually we use pooling and reduce operations in order to classify this central element. A problem that has this architecture to classify two contiguous elements, we need to process two sequences that only differ in one element. This produce that we recompute $T-1$ times the same element in the sequence. In order to reduce this massive excess of computation in long context sequences we will use a many-to-many paradigm. In our experiments we select unique very long sequences from the text as the context, and we compute the convolutions to the whole sequence only once. For this our *WRNs* do not use any pooling strategy, the stride is always 1 and use the appropriated padding in order to fix the edge effects in convolutional layers. Therefore, at the output of the *WRBs* X_{WRN} in figure 2a, we have the same sequence length as the input, T. In Figure 2b it is represented the shape of the *WRNs* output sequence. N is the number of sequence examples evaluated at the same time in a forward pass, or also called mini-batch. Each sequence has T elements, where each element is a vector $x_{WRN,n,t}, n \in N, t \in T$ with C_W the convolutional filter responses. In Figure 2b we show that each $x_{WRN,n,t}$ is evaluated through the position independent convolution layer and a Softmax non-linearity to obtain the final classification.

To train the network we use the cross-entropy as cost function $J(x, y)$. In the case of many-to-one structure we compute the mini-batch error as:

$$J_{many\,to\,one} = \frac{1}{N}\sum_{n=0}^{N-1} J(x_n, y_n), \tag{1}$$

where each example sequence of T elements correspond to one example of N in the mini-batch. In our case we implement the many-to-many structure. We also use the cross-entropy error loss function $J(x, y)$, but the mini-batch cost is computed for each of the T elements of each of the N examples of the mini-batch as:

$$J_{many\,to\,many} = \frac{1}{N}\sum_{n=0}^{N-1}\frac{1}{T}\sum_{t=0}^{T-1} J(x_{n,t}, y_{n,t}) \tag{2}$$

2.2. Long-Short Term Memory Cells

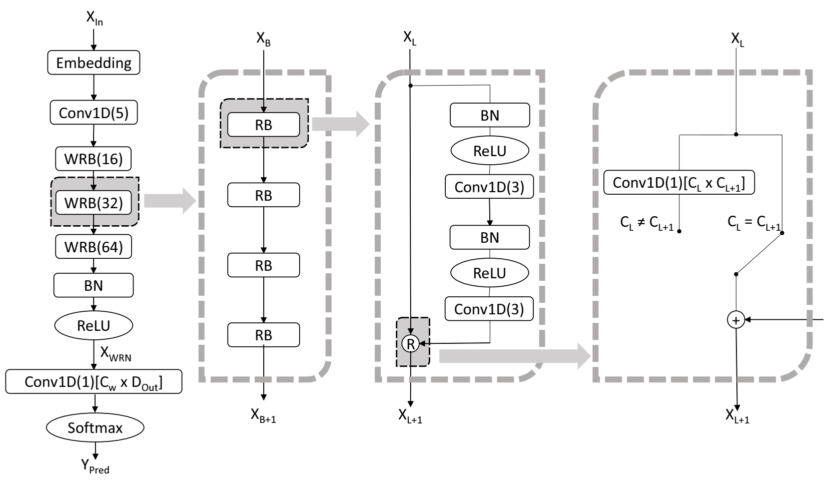
In this study we will compare *WRN* with *LSTM* structures and *BLSTM* structures. In these cases we use the same input X_{in} as described in section 2. In Figure 3 we show that the the input sequences are processed by an *LSTM* or *BLSTM* obtaining a sequence with the same length as the input. In the case of a *LSTM* or a *BLSTM*, the sequence X_{in} passes through the embedding layer as in a *WRN* obtaining a sequence of same length but each element x_t from the sequence is a vector of embedding dimension C, as we represent in Figure 3. This sequence is evaluated by the *LSTM* or the *BLSTM* obtaining a sequence where each element has dimension H or $2{*}H$ respectively with H the hidden dimension of the *LSTM* or the *BLSTM* layers. With this structure we use the same philosophy many-to-many as in *WRN*, and we use the same loss function described in equation 2.

To obtain a combination of *WRN* and *BLSTM* we just substitute the position independent convolution and the Softmax in *WRN* by the *BLSTM* structure showed in Figure 3. Then we will remove or add *WRBs* to the structure to get different results discussed in section 3

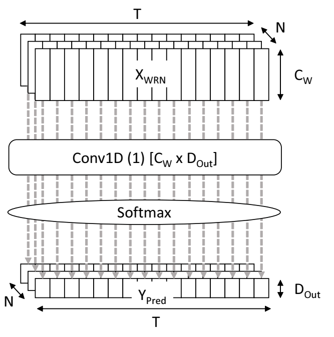
3. Experiment and Results

For this experiment we have collected 1,181,413 articles from electronic edition of 32 diverse Spanish magazines and newspapers, form January 2017 up to February 2018. Each article has 506 words on average. For our purposes all texts were case lowered; numbers, dates, hours and Roman numbers were automatically transcribed and normalized; some units and abbreviations like Km, m, Kg, min, were changed by their transcription; all symbols were deleted; and final question and exclamation marks were substituted by dots. This data set was segmented in train set (980,000 articles), development set (101,000 articles) and test set (100,413 articles), randomly chosen. The dictionary for this work is composed by all words in train text which appear at least 50 times in order to avoid misspelled and rare words. This supposes that all our experiments use a dictionary with 116,737 words including especial delimiters.

In all of the architectures tested in this work we use as first layer an embedding layer with input dimension the number of



(a) *Wide Residual Network.*



(b) *Linear classification block in a* WRN.

Figure 2: *Diagram of the Wide Residual Network architecture. The left figure dis-aggregate the different blocks in* WRN*, and the right figure shows dimensions and schema of the linear classification block in the* WRN.

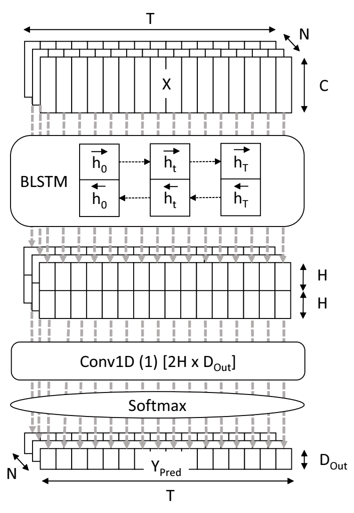


Figure 3: *BLSTM Classification Architecture.*

words in the dictionary, and 1024 as output dimension. The first convolutional layer in the structures has 512 output channels, stride of one and padding of two in order to cover edge effects. In wide residual blocks we use 8 as widen factor. The dimension of each *WRB* is the widen factor times 16 for the first block, the widen factor times 32 in the second block, the widen factor times 64 in the third block and the widen factor times 128 in forth block. The *LSTM* and *BLSTM* architectures are composed by two layers of 1024 cells. All the architectures were trained during 500 iterations where, for this experiment, we consider an iteration of training to process 3,000 random articles from the training set. All trainings were accomplished by back-propagation procedure with Adam optimizer [16] configured with default hyper-parameters in Pytorch software [17].

In order to understand the behaviour of each architecture, we have studied the response of the output when we modify the input. We want to get an idea of the contribution of each word from the input to the classification of a particular output time. For this we have cancelled sequentially each word of the input, by forcing to zero the output of the embedding layer for this word, and representing the value of the network output after a forward pass of this modified input. We assume that those words that do more contribution to a correct classification would be those words that when are cancelled disturb more the classification. In Figure 4 we can see the output values for "Period" class before the Softmax non-linearity of the *LSTM*, *BLSTM* and *WRN* for an output time where a "Period" should be predicted. The dot probability output is shown as each word of the input is sequentially cancelled. One of the first things we can look at is the temporal distance of the first and last cancelled word that does a significant perturbation in "Period" classification. *LSTM* only has perturbations with words before the moment in which we want the classification. We can see that this architecture only uses past information for classification. This behavior may lead to worse performance because future words should be useful in punctuation task. We can not say that a sentence is ended if we do not know when the text change of sense, or subject, or action. In *BLSTM* architecture we can see that it uses words from past and from future. Even those words that do a major perturbation are from a future context. The case of *WRNs* is the same as *BLSTM*, but it uses less number of words for the classification because the perturbations are closer to the classification time.

For evaluation purposes we use three measures, precision, recall and F_1-measure, for each class of interest.

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{3}$$

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{4}$$

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{5}$$

For this measures only relevant cases are taken into account as we see in equations 3, 4 and 5, where only consider the True Positive, False Positive, and False Negative for each class of interest.

For this paper we start considering as baseline a *LSTM* and a *BLSTM*. In Table 1 we show that *BLSTM* provides better results

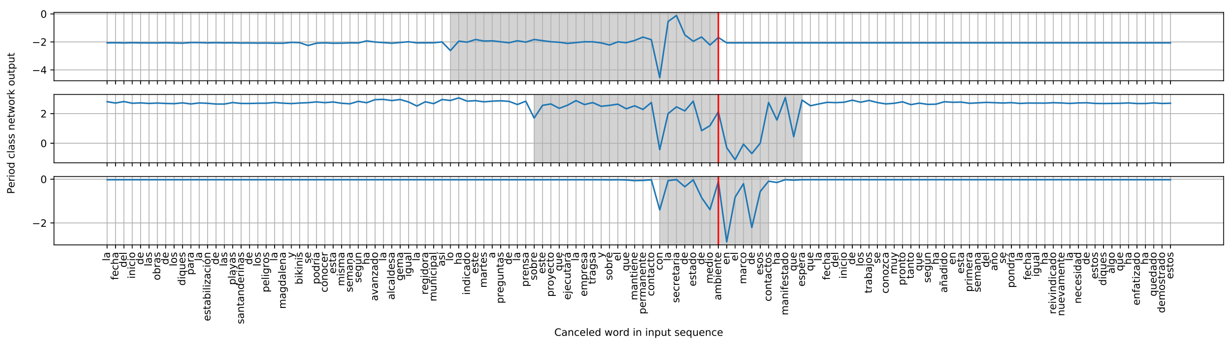


Figure 4: *Variations caused by the cancellation of an input word to "Period" class from the perspective of a True period label after the word "ambiente" pointed by the vertical line. In grey shadow we show the context that affects the classification. From top to bottom we represent the same output from* LSTM, BLSTM *and* WRN *architectures.*

Table 1: *Precision, Recall and F_1-Measure percentage for "Period" and "Comma" classes along the different architectures.*

Architecture	Period			Comma		
	Precision	Recall	F_1	Precision	Recall	F_1
LSTM	73.78	82.18	77.75	87.74	81.36	84.43
BLSTM	87.12	88.96	88.03	91.28	89.78	90.52
Convolutional 1WB	84.05	87.47	85.73	89.76	86.88	88.30
Convolutional 2WB	86.22	89.11	87.64	91.04	88.60	89.80
Convolutional 3WB	89.66	83.75	86.60	88.01	92.51	90.20
Convolutional 4WB	73.79	96.22	83.53	95.92	72.22	82.40
WRN 1WRB	83.10	88.66	85.79	90.67	85.94	88.24
WRN 2WRB	86.20	88.91	87.53	91.13	88.89	90.00
WRN 3WRB	87.41	89.02	88.21	91.17	89.84	90.50
WRN 4WRB	76.30	87.79	81.64	89.23	78.76	83.67
WRN 1WRB + BLSTM	89.33	92.36	**90.82**	93.86	91.38	**92.60**
WRN 2WRB + BLSTM	89.28	90.74	90.00	92.62	91.43	92.02
WRN 3WRB + BLSTM	86.93	89.92	88.40	81.81	89.32	90.55

than *LSTM* achieving a 88.03 of F_1 with "Period" class, and a 90.53 of F_1 with "Comma". These improvements are obtained thanks to use past and future context in *BLSTM*.

Residual connections usually work better when the network is very deep, but they also help in shallower networks. In order to see how residual connections help the classification, we trained convolutional networks which are exactly the same architecture as *WRNs* but without residual connections. In Table 1 we show that when we increase the number of blocks in convolutional networks without residual connections, they start to get worse results than the same configuration in *WRNs*. With only one block they perform equal, but with three blocks, the network without residual connections performs worse. We achieve the best result with a *WRN* using three blocks. We obtain an F_1 of 88.21 for "Period" and 90.50 for "Comma", which is the same performance as the base case of a *BLSTM*.

In the last experiments we concatenate the output of a *WRN* with the input, without embedding layer, of the *BLSTM*. These two architectures are compatibles because both are designed to work with sequences, processing element by element, so we do not need to do any transformation to the inputs or outputs. In Table 1 we present the result of concatenation of a *WRN* of one, two or three blocks and a *BLSTM* with the same architecture as the baseline. We can see that all those configurations perform better than any of the two architectures alone. The case of a *WRN* of one block presents the better result with an F_1 of $\mathbf{90.82}$ in "Period" and $\mathbf{92.60}$ in "Comma".

4. Conclusions and future work

In this work, we have shown the use of Wide Residual Networks for punctuation recovering. We show that architectures that use past and future context obtain better results. *WRNs* performs similarly to *BLSTM* in our experiments so other aspects of the architecture should be taken into account, like training time or resources consumed. The best results are obtained when *WRNs* and *BLSTMs* are concatenated. This work suggests the idea of the use of *WRNs* as feature extractor. Then we need powerful structures than a linear classifier to perform the final classification. In problems that requires sequential processing, *BLSTM* structures that uses past and future context, are the best option.

For future work, it would be interesting to differentiate between question marks and "periods". In Spanish, this is a challenging task since we need to predict the closing question mark, but also the opening question mark, which is a symbol not used in other languages like English. Future work also will pay attention to architecture improvements like hyper-parameter configuration or to include attention mechanisms.

5. Acknowledgements

This work is supported by the Spanish Government and the European Union under project TIN2017-85854-C4-1-R, and by Gobierno de Aragón / FEDER (research group T36_17R). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

6. References

[1] F. Batista, H. Moniz, I. Trancoso, and N. Mamede, “Bilingual experiments on automatic recovery of capitalization and punctuation of automatic speech transcripts,” *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 20, no. 2, pp. 474–485, 2012.

[2] O. Tilk and T. Alumäe, “Lstm for punctuation restoration in speech transcripts,” in *Sixteenth annual conference of the international speech communication association*, 2015.

[3] ——, “Bidirectional recurrent neural network with attention mechanism for punctuation restoration.” in *Interspeech*, 2016, pp. 3047–3051.

[4] W. Lu and H. T. Ng, “Better punctuation prediction with dynamic conditional random fields,” in *Proceedings of the 2010 conference on empirical methods in natural language processing*. Association for Computational Linguistics, 2010, pp. 177–186.

[5] K. Xu, L. Xie, and K. Yao, “Investigating lstm for punctuation prediction,” in *Chinese Spoken Language Processing (ISCSLP), 2016 10th International Symposium on*. IEEE, 2016, pp. 1–5.

[6] X. Che, C. Wang, H. Yang, and C. Meinel, “Punctuation prediction for unsegmented transcript based on word vector.” in *LREC*, 2016.

[7] J. Yi, J. Tao, Z. Wen, and Y. Li, “Distilling knowledge from an ensemble of models for punctuation prediction,” *Proc. Interspeech 2017*, pp. 2779–2783, 2017.

[8] W. Gale and S. Parthasarathy, “Experiments in character-level neural network models for punctuation,” in *Proceedings Interspeech*, 2017, pp. 2794–2798.

[9] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[10] S. Zagoruyko and N. Komodakis, “Wide residual networks,” *arXiv preprint arXiv:1605.07146*, 2016.

[11] H. Li, Z. Xu, G. Taylor, and T. Goldstein, “Visualizing the loss landscape of neural nets,” *arXiv preprint arXiv:1712.09913*, 2017.

[12] H. K. Vydana and A. K. Vuppala, “Residual neural networks for speech recognition,” in *Signal Processing Conference (EUSIPCO), 2017 25th European*. IEEE, 2017, pp. 543–547.

[13] L. D. Jahn Heymann and R. Haeb-Umbach, “Wide residual blstm network with discriminative speaker adaptation for robust speech recognition,” in *Proceedings of the 4th International Workshop on Speech Processing in Everyday Environments (CHiME16)*, 2016, pp. 12–17.

[14] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[15] X. Glorot, A. Bordes, and Y. Bengio, “Deep sparse rectifier neural networks,” in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[16] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[17] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in pytorch,” in *NIPS-W*, 2017.