Giménez, M.; Palanca Cámara, J.; Botti Navarro, VJ. (2020). Semantic-based padding in convolutional neural networks for improving the performance in natural language processing. A case of study in sentiment analysis. Neurocomputing. 378:315-323. https://doi.org/10.1016/j.neucom.2019.08.096

Semantic-based padding in Convolutional Neural Networks for improving the performance in Natural Language Processing. A case of study in Sentiment Analysis.

Maite Gimènez, Javier Palanca, Vicent Botti

Departamento de Sistemas Informàticos y Computaciòn,
Universitat Politècnica de València

Abstract

In this work, we proposed a methodology for applying semantic-based padding in Convolutional Neural Networks for Natural Language Processing tasks. Semantic-based padding takes advantage of the unused space required for having a fixed-size input matrix in a Convolutional Network effectively, using words present in the sentence. We have evaluated the performance of the methodology proposed intensively in Sentiment Analysis tasks using a variety of word embeddings. In all the experimentation carried out the proposed semantic-based padding improved the results achieved when no padding strategy is applied. Moreover, when we used a pre-trained word embeddings, we have managed to surpass the state of the art.

Keywords: Natural Language Processing, Convolutional Neural Networks, Padding

1. Introduction

Recently, deep learning models had revolutionised the performance of different classic machine learning tasks. In the literature, one can found how techniques developed in Computer Vision (1; 2; 3; 4) had been applied to Natural language Processing (NLP) successfully (5; 6; 7). Convolutional Neural Networks (CNN), a deep neural network, proposed by **(author?)** (8), had proven that can improve or achieve competitive performance in several NLP tasks such as Semantic Matching tasks, Text Classification, Sequence Ordering or Context Dependency (9).

In order to be able to apply CNNs effectively to Natural Language Processing problems, the text of a sentence should be treated by the network similarly as it would treat an image. Embeddings, vectors of a fixed size, representing either words or n-grams of characters are used for creating a matrix describing a sentence. Thus, a convolutional neural network can be trained in solving a natural language processing task utilising this matrix of embeddings. However, there are caveats when transferring techniques from one area of study to another.

In this work, we will study one of the most prominent problems that we face when handling text as an image: the padding. Unlike images that have a fixed size, or can be transformed to a fixed size, a sentence has a variable number of words. Hence, practitioners need to establish a single input size in order to be able to apply CNNs. This problem is neglected, and the sentence is padded with a vector of zeros. Our working hypothesis is that this space that we allocate could be used more efficiently. Therefore, we will pad the sentences with meaningful semantic vectors.

Here, we propose a technique to fix the input size, alleviating the drawbacks of the standard zero-padding, and we prove that this methodology improves the performance of the neural network model. Moreover, this proposed methodology will not adversely affect the complexity of the training or penalise the convergence of the neural network. The contributions presented in this paper could be summarised as follow:

- Study of the impact of the padding applied in Convolutional Neural Networks when addressing Natural Language Processing tasks.
- Proposed a general methodology for padding more efficiently.
- Validate that the suggested semantic-based padding improves the performance of a CNN architecture.

The primary goal of this work is improving how techniques developed in Computer Vision or Speech Recognition can be adequately used in Natural Language Understanding, taking into account all the nuances that text presents. We have replicated state-of-the-art architectures and found that applying semantic-based padding improves their performance in a statistically significant way.

The rest of the paper is organised as follows. Next section will explore the literature related to our problem, in particular, we will describe the efforts of applying CNNs to NLP tasks. Section 3 is devoted to define the methodology proposed for semantic-based padding in CNNs. In Section 4, the experimentation that validates our methodology is presented. Finally, in Sections 5 and 6, we discuss our results and future work is proposed.

2. Related Work

This section is devoted to describing Sentiment Analysis, the task used to evaluate the performance of the proposed method-

ology for padding Convolutional Neural Networks Besides, we will review the research done in CNN applied to NLP tasks.

2.1. Sentiment Analysis

Sentiment Analysis (SA) is one of the classic tasks that had attracted the interest of computational linguistics deeply. Therefore, among all the possible NLP tasks we decided to evaluate our proposed methodology on SA because this is one of the most studied tasks within NLP. Consequently, the datasets and the algorithms are well-studied, which allow us to test our methodology in a more stable framework.

Sentiment Analysis is the task of, giving a text, identifying the polarity conveyed in it. Traditionally, three classes of polarity were considered: positive, negative or neutral. Nevertheless, there are tasks where fine-grained polarity levels are considered as well.

One of the early works on SA was developed by **(author?)** (10). In this work, the authors evaluated the performance of different machine learning classifiers using hand-crafted features on movie reviews. Besides the supervised approach of this work, there were unsupervised approaches like the one proposed by **(author?)** (11). This task has been blooming since then, and in 2013 the first shared evaluation of SA using Twitter was held. **(author?)** (12) proposed a SA evaluation campaign using data gathered from Twitter. This campaign is held annually, and many more similar tasks capture the interest of NLP practitioners (13; 14; 15; 16; 17).

In the work of **(author?)** (18) we can find a comprehensive study of the different techniques used to identify the polarity of a text. At this point, most of the approaches used expert knowledge for hand-crafting features to be extracted from the text, building lexicons (19; 20; 21) and applied techniques such as SVM, Maximum Entropy, Naive Bayes, etc., (22; 23; 24; 25).

Noteworthy, the massive growth in the usage of social networks facilitated the creation of large datasets. (26; 27) with enough examples for training deep learning networks which were already used in Computer Vision. Simultaneously, **(author?)** (28) proposed a new representation methodology of text, word embeddings. Text representation entails a challenge for Machine Learning, since previously used techniques were not able to capture efficiently the semantic similarity of words. However, word embeddings are able to learn from big data (in an unsupervised fashion) a representation in a multidimensional space of the words that capture the semantic relationships between them. Recently, new techniques appeared for learning word embeddings, and new datasets have been collected (29; 30).

2.2. Convolutional Neural Networks in NLP

Following the advances described in the previous section, Convolutional Neural Network began to be applied in Natural Language Tasks. This section is devoted to explaining the state of the art in CNNs.

(author?) (31) defined Convolutional Networks as neural networks that use convolutions instead of matrix multiplication in at least one of its layers. A convolution is a linear operation that takes two multidimensional arrays: an input $\vec{x} \in \mathbb{R}^{m,n}$ and a kernel $\vec{w} \in \mathbb{R}^{h,k}$ where $h < m \wedge k < n$. After applying the convolution to those arrays, it will produce multidimensional arrays called feature maps. Each element of a feature map is obtained after applying the convolution across a window of words $\{\vec{x}_{0:h}, \vec{x}_{1:h-1}, \dots \vec{x}_{n-h+1:n}\}$, and it is defined as:

$$c_i = f(\vec{w} \cdot \vec{x}_{i:i+h-1} + b_i) \quad (1)$$

where $\vec{c} \in \mathbb{R}^{n-h+1}$, $b_i \in \mathbb{R}$ is a bias term, and f a non linear function.

CNNs were proposed by **(author?)** (32), but it was not until recently than CNNs began to be massively used after the success that these networks showed in computer vision (3; 4). Following the momentum of CNNs in Computer Vision, it began to be used in NLP as well. Firstly, **(author?)** (33) proposed an architecture using CNNs for Natural Language Processing. Recently, CNNs architectures have been combined in an ensemble of classifiers with recurrent neural networks to exploit the benefits of both architectures (34; 35).

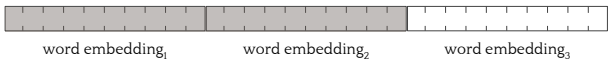
Besides, CNNs have been applied to several NLP tasks such as Sentence Classification (36; 37), Document Ranking (38), or Causal Relation Extraction (5).

3. A Semantic-based Padding

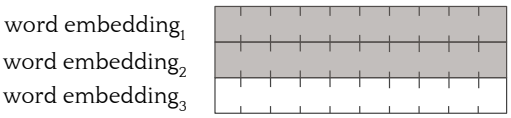
As we introduced previously, there are several methodologies for representing text. Nevertheless, in the last years, the most extensively used is text embeddings. Embeddings are a distributed representation of either words or characters in a multidimensional space as we defined in Section 2. We will focus our work on word embeddings. Consequently, from now on we will use the term embeddings as an equivalent term for word embeddings.

When applying CNNs to Computer Vision, it has been proved by **(author?)** (39) and **(author?)** (40) that the network presents local invariance, i.e. can find relevant feature in a different position of the image, and is able to learn how to compose features from shallow to deeper levels. Transferring these properties to NLP tasks, a fundamental assumption will be that the local invariance allows a CNN to learn which words are the more relevant to the classification task, and the compositional property allows the network to learn how to combine words embeddings to understand the meaning of a sentence. However, since the size of the filter is frequently limited between two and seven n-grams, CNNs are not suited to learn phrases that change the meaning of a sentence entirely if the distance between words is larger than the size of the kernel (9). Figurative language, like the language used in sentences that contain humour, irony or metaphors present these linguistic constructions, where part of the sentence conveys a sentiment and the other part the opposite. These long-distance relationships are hard to capture by CNNs.

In the literature, there are two approaches to build a CNN for text classification. On the one hand, word embeddings can be concatenated and then apply a one-dimension convolution with



(a) Convolution 1D.



(b) Convolution 2D.

Figure 1: Diagram depicting how to prepare text to train a CNN. The shadowed area corresponds to a filter that learns bi-grams. In 1a word embeddings are stacked horizontally. The height of a filter is fixed, and the width will be a multiple of the size of the embedding vectors; hence, a convolution in one dimension is applied. In 1b word embeddings are stacked vertically. The width of a filter is fixed to the size of the embedding vectors, and the height will vary depending on the size of the n-gram to learn; hence, a convolution in two dimensions is applied.

a stride of the size of the embeddings. On the other hand, embeddings can be stacked vertically and compute a convolution were the width of the convolution has been fixed to the size of the embeddings. In the figure 1 a diagram of both approaches is presented.

However, in both cases, the input size of the convolution must be constant through the whole dataset. The maximum size of the matrix is determined by the sentence of the maximum length in the training dataset. Sentences shorter than this value will be padded and sentences longer, in the test dataset, will be trimmed. Therefore, the size of the embeddings matrix will be:

$$\mathbb{R}^{m \times e} \tag{2}$$

being m the size of the longest sentence, and e the size of the embeddings used.

This restriction will lead to the inclusion of padding. Padding guarantees that all the input sentences will have the same dimensions and the convolution can be computed efficiently. This padding is filled with a vector of a particular token, usually a vector of zeros.

Padding with a vector of zeros will introduce noise in the input sentence that will influence the training of CNN model. To address the problems presented, we propose to pad the sentence with embeddings of words present in the text. Therefore, when the kernel is sliding through the matrix of words towards the end of the sentence, it will learn relationships between the end of the sentence and the beginning, forcing the network to learn long-distance relationships between words. In addition, this proposed semantic-based padding neither increase the size of the input of the CNN nor represent a bottleneck preprocessing the input data set because this padding process was already being carried out and our proposed method doesn't constitute a computational overload.

Three padding methodologies are proposed.

- **Random**: This is the most basic way to fill the embeddings matrix. Each one of the empty rows will be filled with embeddings from random words present in the sentence. This padding doesn't keep any semantic meaning, but we include it as a baseline.

- **Loop**: This is our first proposed semantic-based padding. After including each word of a sentence, the same sentence is included repeatedly, similarly as in the memory tape of a Turing machine, until the end of the matrix is filled. Hence, we are forcing the CNN architecture to learn long-distance relationships between phrases.

- **Roll**: This last padding is a variant of the previous one. In this case, we will repeat each word of the sentence once in the embeddings matrix. Thus, at maximum, the sentence will appear two times. If the length of the sentence is two times shorter than m —from Equation 2—, the size of the embeddings matrix, zero-padding will be still added at the end.

  We include this variant, under the assumption that including zero-padding might be helpful for capturing the feature of how long the sentence was before the padding, while still caputuring long distance relationships between words. In systems with hand-crafted features, the length of the sentence is a ubiquitous feature that most of the models include (41; 42).

Besides, we will include a fourth padding methodology in all our experiments, a classic zero-padding, that we will denominate in our experiments **None** padding.

A diagram illustrating the different paddings proposed can be found in Figure 2.

Considering this example sentence that could be found in a dataset *'I love this movie'*; and assuming that the maximum length of sentences seen in training is ten. Each one of the paddings used will transform the sentence as follows:

- **None:** $< I >< love >< this >< movie >< END >< END >< END >< END >$

- **Random:** $< I >< love >< this >< movie >< I >< movie >< love >< love >< movie >< this >$

- **Loop:** $< I >< love >< this >< movie >< I >< love >< this >< movie >< I >< love >$

- **Roll:** $< I >< love >< this >< movie >< I >< love >< this >< movie >< END >< END >$

Brackets separate tokens. The token $< END >$ is used for padding the end of a sentence and it will be replaced by a vector of zeros.

All the experiments will be executed with the four types of paddings. Our objective is to prove that semantic-based paddings can improve the performance of the CNN architecture, regardless of the variations that training deep learning methods may present. Accordingly, we have trained and evaluated CNN models with the padding used in the state-of-the-art

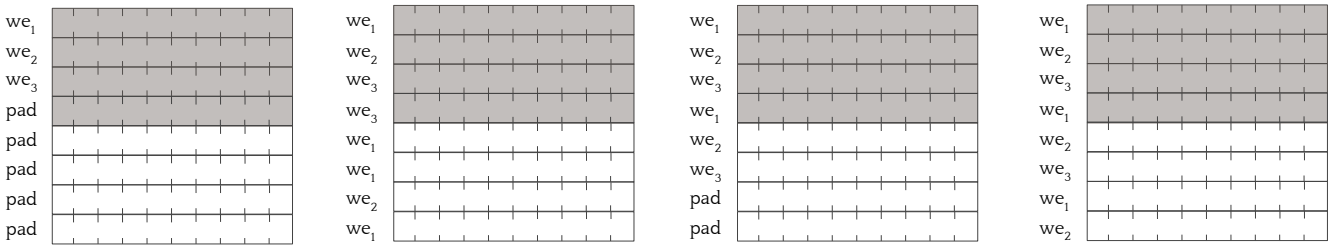


Figure 2: Diagrams with the different paddings compared in paper. Here we define we_i as the word embedding of the i-th word in the sentence. The first matrix corresponds to the standard *None* padding, following the *Random*, *Roll*, and *Loop* padding. The shadowed area is the area of interest of one of the four-grams learned kernels. Each kernel will slide through the sentence learning the feature maps.

systems and the paddings we proposed. This will allow us evaluating fairly the improvements of the methodology.

No other preprocessing of the sentences will be carried out. Firstly, because deep learning networks will learn an internal representation which best suits the task at hand. Secondly, because we want to isolate the effect of the padding from other conditioning factors.

4. Experiments and Results

In this Section, we will describe the experiments conducted for validating that the semantic-based padding actually improves the performance of the systems where it has been applied. We have selected a classic NLP task: sentence-level sentiment analysis on various well-known datasets.

Following the description of the experimental setup is described. Also, we will discuss the results achieved.

4.1. Experimental setup

In order to test the performance of the proposed paddings, we decided to employ a state-of-the-art architecture. The architecture proposed by **(author?)** (37) will be used. In the Figure 3 a diagram of this architecture is depicted. This CNN architecture is a manageable deep learning model which allow us to evaluate the performance of our proposed semantic-based padding with different embeddings and datasets extensively. Besides, this architecture has been validated in the literature ((**author?)** (43, 44, 9)).

We have reimplemented the single-channel architecture proposed using the toolkit Tensorflow developed by **(author?)** (45) and selected the same hyperparameters described in the literature. This is, we use three types of CNN kernels size $\mathbb{R}^{3\times e}$, $\mathbb{R}^{4\times e}$, and $\mathbb{R}^{5\times e}$ respectively, being e the size of the embeddings studied; each type of convolution will learn 100 feature maps; we use rectified linear units after applying the convolution operation; the dropout rate is 0.5, and the fully connected layer consists on 150 neurons. We have trained the model using mini-batches of size 100 through Adam gradient descent. We have used a Titan XP GPU for carrying out all the experimental phase gracefully donated by NVIDIA. We opted for reimplementing the system because this allows to validate the performance of the padding without the variances that might appear due to the particularities of the toolkit or even the hardware used to train the reported state-of-the-art systems.

4.1.1. Sentiment Analysis

Sentiment Analysis, as we discussed in Section 2, is one of the most studied tasks in NLP. This is the reason that led us to select this task for this work. It allows us to compare our proposed method to a well-known task and datasets. The datasets used are:

- **SST-5**: Stanford Sentiment Treebank ((**author?)** (26)), also found in the literature as *SST-fine* is a dataset consisting of movie reviews that were annotated for five levels of sentiment: strong negative, negative, neutral, positive and strong positive. This dataset is annotated both phrase-level and sentence-level. Only the label for the whole sentence was considered as the golden truth.

- **SST-2**: Stanford Sentiment Treebank binary ((**author?)** (26)). The sentences of this dataset are the same sentences that previous dataset. However, in this case, only two classes are considered: positive and negative sentences. Sentences with strong positive and positive labels are merged in the positive class. Similarly, sentences with strong negative and negative labels are joined as negative sentences. Finally, neutral sentences are discarded.

Table 1 reflects the statistics of the datasets studied. Noteworthy, the input matrix has size $\mathbb{R}^{52\times e}$ being e the size of the embeddings studied. Thus, on average 32 positions of the input matrix are filled with padding ($\overline{Pad}$). Therefore, it is reasonable to argument that how we pad the input sentence will have an impact on the training of our deep learning model. In addition, the number of sentences that require a considerable amount of padding corresponds to the head of a truncated Gaussian in both datasets, as it can be visualised in Figures 4 and 5. However, the tail of these Gaussians, i.e. the longer sentences, determine the size of the input matrix. Besides, this problem will increase in tasks where the average length of the sentences is smaller than in these datasets and where sentences present more variability in their lengths.

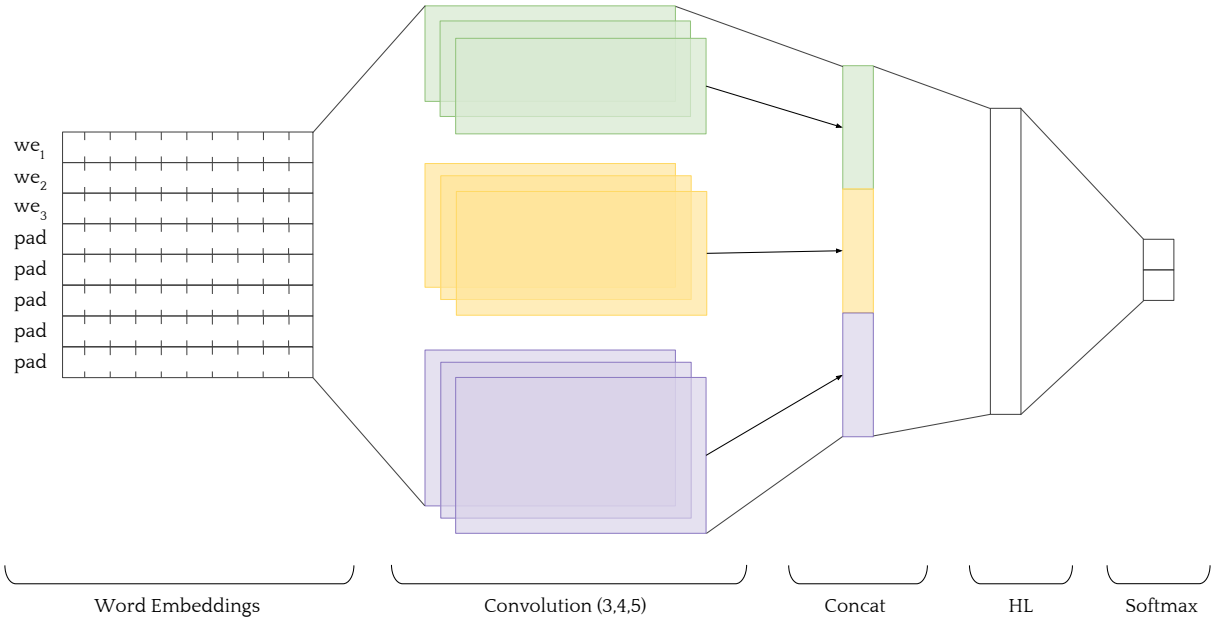


Figure 3: Model architecture without any kind of padding (None padding).

	Train	Dev.	Test	Vocabulary	$\overline{Length}$	Min	Max	$\overline{Pad}$
SST-2	6920	872	1821	17539	19.3	2	52	32.45
SST-5	8544	1101	2210	19500	19.14	2	52	32.85

Table 1: Statistics of the number of words and padding needed in the datasets.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
Random	None	74.83 (0.74)	-	-	-
	Random	72.96 (1.09)	-	-	-
	Roll	75.18 (1.1)	0.34	$P < .001$	-
	Loop	77.31 (0.16)	2.47	$P < .001$	$P < .001$

Table 2: Results obtained experimenting with Random Word Embeddings on the SST-2 dataset.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
Random	None	33.91 (1.50)	-	-	-
	Random	33.51 (0.96)	-	-	-
	Roll	34.42 (1.54)	0.50	$P < .001$	-
	Loop	35.03 (1.51)	1.52	$P < .001$	$P < .001$

Table 3: Results obtained experimenting with Random Word Embeddings on the SST-5 dataset.

4.1.2. Word Embeddings Studied

In this section, we will describe the word embeddings studied. We decided to use different word embeddings to validate the ability of the semantic-based padding to improve the performance of a model regardless of the methodology used for training these embeddings.

- **Random**: In this case, we assigned a vector of size $\mathbb{R}^{100}$ to each word. These vectors are initialised with random noise. Values are drawn from a uniform distribution. These word embeddings are the only one used that were not pre-trained. Hence, they have no previous knowledge. Consequently, we need to train word embeddings while we are learning the classification task.

- **NNLM-50**: These word embeddings were trained following the Neural Network Language model proposed by **(author?)** (46). It maps each word into a 50-dimensional embedding vector. The Neural Network that learned these embeddings was trained on English Google News 200B corpus. Besides, it has a pre-built out-of-vocabulary (OOV) method that maps words that were not seen in the

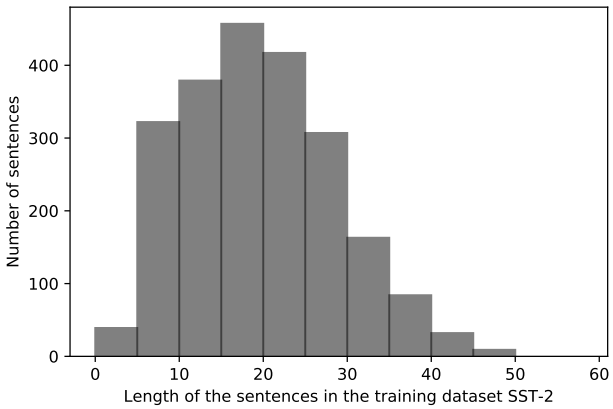


Figure 4: Distribution of the sentences length in the dataset SST-2.

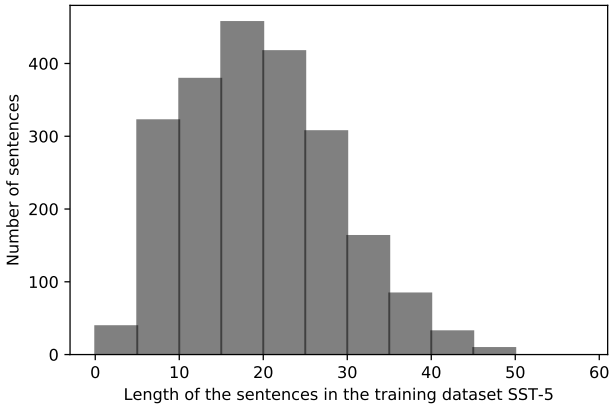


Figure 5: Distribution of the sentences length in the dataset SST-5.

vocabulary of the training dataset with hash buckets. Each hash bucket is initialised using the remaining embedding vectors that hash to the same bucket.

- **NNLM-50-norm**: Similar to the previous one but in this case, word embeddings were normalised.
- **NNLM-128**: These words embeddings are trained in the same fashion as *NNLM-50*. However, each word is mapped into a 128-dimensional embedding vector.
- **NNLM-128-norm**: Analogous, these words embeddings are a normalized version of *NNLM-128*. Each one of the NNLM models were proposed by **(author?)** (46)
- **W2V-250** Word embeddings based on skipgram version of word2vec proposed by **(author?)** (28). They were trained using the English Wikipedia corpus. The algorithm used for training was hierarchical softmax, and sub-sampling was set to 1e-5. All the words OOV were mapped into one bucket which was initialised with zeros.
- **W2V-250-norm**: These embeddings are a normalized version of *W2V-250*. These last two embedings were proposed by **(author?)** (28).

All word embeddings modules studied, except for the *Random* word embeddings, were downloaded from the on-line library Tensorflow Hub [1].

Moreover, since word embeddings can be trained for improving the task at hand, we have included experiments where the word embedding input will be updated while training the sentiment analysis classifier. Experiments carried out with embeddings are labelled in the results as training.

4.2. Results

In this Section, we will expose the results of our experimentation. As we described previously, we have implemented the model presented in Section 4.1 to validate the impact of the semantic-based padding in Sentiment Analysis tasks. Moreover, we have repeated ten times each experiment, and we report the mean accuracy of the experimentation and its standard deviation.

We reported as well the absolute difference of the semantic-based padding and the zero-based padding, i.e. None padding.

In addition, we have calculated the T-test (47) between None-padding and the semantic-based padding, i.e. Roll-padding and Loop-padding, to demonstrate that it exists a significant difference between the state-of-the-art padding and the semantic-based padding. Also, we have computed the T-test between the two proposed semantic-based paddings to find out if there are differences among these two methodologies for padding sentences.

The results of the experimentation with dataset SST-2 can be found in Tables 2, 4, 6, and 8. Similarly, the results of the experimentation with dataset SST-5 can be found in Tables 3, 5, 7, and 9.

In several experiments, we have improved the results presented in the state-of-the-art papers (37; 43). These experiments are highlighted in bold. Noteworthy, if the model with None padding, the one used in the literature, achieves the state-of-the-art performance, our proposed semantic-based padding improves the accuracy. Moreover, in all the cases the semantic-based padding outperforms the zero padding approach.

5. Discussion and Future Work

Following, we will evaluate the results presented in Section 4.2. Firstly, the most obvious conclusion we can draw from the experiments assessed is that semantic-based padding improves the performance of the Sentiment Analysis. And it does it in a statistically significant way, as the p-value calculated between the results achieved without padding and the ones obtained with both semantic-based padding are below $< .05$ in almost every case. However, when we are training the word embeddings simultaneously with the SA task, the performance of the semantic-based padding does not have an impact as relevant

[1] For more information, please visit the following URL `https://www.tensorflow.org/hub/`

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
NNLM-50	None	75.17 (1.42)	-	-	-
	Random	76.00 (1.06)	-	-	-
	Roll	76.44 (0.91)	1.26	.015	-
	Loop	77.50 (0.71)	1.50	$P < .001$	$P < .001$
NNLM-50 training	None	76.30 (1.06)	-	-	-
	Random	76.58 (2.31)	-	-	-
	Roll	77.20 (0.85)	0.89	$P < .001$	-
	Loop	76.82 (0.69)	0.51	.03	$P < .001$
NNLM-50 norm.	None	76.49 (0.73)	-	-	-
	Random	75.81 (0.90)	-	-	-
	Roll	77.47 (0.54)	0.97	$P < .001$	-
	Loop	77.92 (1.11)	1.42	$P < .001$	.17
NNLM-50 norm. training	None	76.48 (0.64)	-	-	-
	Random	75.37 (0.92)	-	-	-
	Roll	76.86 (0.91)	0.38	$P < .001$	-
	Loop	77.70 (0.61)	1.21	$P < .001$	$P < .001$

Table 4: Results obtained experimenting with NNLM-50 Word Embeddings on the SST-2 dataset.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
NNLM-50	None	35.41 (1.24)	-	-	-
	Random	35.71 (1.28)	-	-	-
	Roll	36.01 (0.64)	0.59	$P < .005$	-
	Loop	36.20 (0.54)	0.78	$P < .001$	.64
NNLM-50 training	None	35.30 (0.83)	-	-	-
	Random	35.05 (0.94)	-	-	-
	Roll	37.40 (0.70)	2.10	$P < .001$	-
	Loop	37.08 (1.45)	1.78	$P < .001$	.62
NNLM-50 norm.	None	34.19 (0.97)	-	-	-
	Random	34.73 (1.41)	-	-	-
	Roll	36.32 (1.64)	2.13	$P < .001$	-
	Loop	36.35 (1.22)	2.16	$P < .001$	.54
NNLM-50 norm. training	None	35.15 (0.87)	-	-	-
	Random	35.30 (1.69)	-	-	-
	Roll	36.79 (0.72)	0.38	$P < .001$	-
	Loop	37.58 (0.65)	1.21	$P < .001$	$P < .001$

Table 5: Results obtained experimenting with NNLM-50 Word Embeddings on the SST-5 dataset.

as when embeddings are not trained. Updating the weights of the embedding input during training helps to increase the performance in the task. Therefore, the margin for improvement with the semantic-padding is narrower. As future work, we could modify the learning rate and the number of training epoch hyperparameters to validate the impact of semantic-padding in these scenarios.

Secondly, Random padding always performs worst than any other padding. This is because the model is learning relationships between words that have no relationship between them in the sentence. Despite the individual impact of some words in determining the sentiment of a sentence. This experiment also proves that CNNs are learning how words interact with each other, and when random connections appear the model performs worst.

In the results presented both semantic-based paddings boost the performance achieved by the zero padding. The improvement found between both systems is very similar, which is validated by the p-value obtained. In most of the experiments, the difference between the two proposed semantic-based paddings is not statistically significant. On the one hand, the end of sentence tokens present on the *Roll* semantic-based padding encodes the length of the sentence, which is a typical feature in models trained with hand-crafted features. On the other hand, the *Loop* semantic-based padding allows CNN to learn the same pattern of words in more positions of the embedding input matrix. Although both proposed semantic-based paddings achieve performance very similar, outperforming the state-of-the-art used zero padding, the features learned by the model differ. It will be interesting to investigate in future work how this internal representation learned affects the performance of the model.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
NNLM-128	None	81.34 (1.3)	-	-	-
	Random	81.31 (0.41)	-	-	-
	Roll	**82.45** (0.45)	1.09	$P < .001$	-
	Loop	**82.36** (0.67)	1.02	$P < .001$	.53
NNLM-128 training	None	75.04 (3.23)	-	-	-
	Random	73.75 (0.81)	-	-	-
	Roll	75.76 (1.17)	0.72	.067	-
	Loop	75.57 (1.50)	0.53	.14	.65
NNLM-128 norm. training	None	**81.63** (0.56)	-	-	-
	Random	80.08 (1.06)	-	-	-
	Roll	**82.25** (0.75)	0.62	$P < .001$	-
	Loop	**82.00** (0.75)	0.37	$P < .001$	.43
NNLM-128 norm. training	None	75.09 (1.81	-	-	-
	Random	74.58 (0.75)	-	-	-
	Roll	77.55 (0.53)	2.46	$P < .001$	-
	Loop	76.57 (0.25)	1.48	$P < .005$	$P < .001$

Table 6: Results obtained experimenting with NNLM-128 Word Embeddings on the SST-2 dataset.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
NNLM-128	None	38.59 (1.82)	-	-	-
	Random	38.60 (0.41)	-	-	-
	Roll	**40.93** (1.45)	2.3	$P < .001$	-
	Loop	40.07 (0.65)	1.47	$P < .001$	.41
NNLM-128 training	None	34.95 (1.20	-	-	-
	Random	33.72 (1.91)	-	-	-
	Roll	39.47 (1.12)	4.52	$P < .001$	-
	Loop	39.57 (0.57)	4.62	$P < .001$	.84
NNLM-128 norm.	None	38.35 (0.95)	-	-	-
	Random	38.32 (0.70)	-	-	-
	Roll	39.56 (1.43)	1.21	$P < .001$	-
	Loop	40.07 (0.98)	1.72	$P < .001$	.76
NNLM-128 norm. training	None	34.12 (2.53)	-	-	-
	Random	33.35 (0.98)	-	-	-
	Roll	35.46 (0.78)	1.34	$P < .005$	-
	Loop	36.62 (0.98)	2.50	$P < .001$	.78

Table 7: Results obtained experimenting with NNLM-128 Word Embeddings on the SST-5 dataset.

Another future work will be to try this technique with new tasks within Natural Language Processing. Also, new datasets of Sentiment Analysis might be worth evaluating, particularly those where the relationship between phrases in a sentence has a significant impact such as corpora where irony or humour is present. Following this line of future work, since there are pre-trained word embeddings and datasets in other languages, one could evaluate if the semantic-based padding also improves the performance of NLP tasks in other languages.

6. Conclusions

In this work, we have proposed a random padding and two semantic-based paddings for Natural Language Processing tasks and proved its performance with two Sentiment Analysis datasets and seven different word embeddings. For each experiment performed the semantic-based padding proved to improve the performance of the system. Moreover, when the model without padding achieved state-of-the-art performance, the semantic-based padding outdo the accuracy reported in state of the art.

Finally, we had proposed future work that can expand the achievements presented in this work. Namely, study the internal representation learned by the Convolutional Neural Network, which will improve the interpretability of the network as well; and expand the experimentation to other domains.

In summary, studying the internals of deep learning networks can lead us to improve the performance of the system. Moreover, it contributes to validate or discard hypothesis of what the model is learning, guiding the development of new architectures.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
W2V-250	None	80.38 (3.24)	-	-	-
	Random	80.49 (0.21)	-	-	-
	Roll	**81.88** (0.81)	1.5	$P < .001$	-
	Loop	**82.14** (0.94)	1.76	$P < .001$	.50
W2V-250 training	None	73.67(5.27)	-	-	-
	Random	73.99 (1.21)	-	-	-
	Roll	76.27 (0.98)	2.60	$P < .001$	-
	Loop	75.83 (1.57)	2.16	$P < .001$	.08
W2V-250 norm.	None	77.18 (0.68)	-	-	-
	Random	75.11 (2.12)	-	-	-
	Roll	77.97 (0.74)	0.79	.03	-
	Loop	78.08 (0.61)	0.90	$P < .005$	.32
W2V-250 norm. training	None	75.37 (1.66)	-	-	-
	Random	75.03 (0.45)	-	-	-
	Roll	76.26 (0.67)	0.89	$P < .005$	-
	Loop	77.25 (1.58)	1.88	$P < .001$	$P < .005$

Table 8: Results obtained experimenting with W2V-250 Word Embeddings on the SST-2 dataset.

Embeddings	Padding	$\overline{Accuracy}$	$\lvert\Delta\rvert$	p-v. None	p-v. Roll
W2V-250	None	33.75 (0.90)	-	-	-
	Random	32.20 (0.79)	-	-	-
	Roll	35.71 (1.28)	1.96	$P < .001$	-
	Loop	35.26 (1.04)	1.50	0	
W2V-250 training	None	32.67 (0.19)	-	-	-
	Random	30.28 (1.80)	-	-	-
	Roll	34.34 (0.87)	1.67	$P < .001$	-
	Loop	33.18 (1.99)	0.51	.058	$P < .001$
W2V-250 norm.	None	34.85 (1.32)	-	-	-
	Random	33.90 (1.21)	-	-	-
	Roll	35.44 (0.33)	0.59	$P < .005$	-
	Loop	35.55 (1.24)	0.70	$P < .005$	.72
W2V-250 norm. training	None	33.03 (2.37)	-	-	-
	Random	32.40 (2.83)	-	-	-
	Roll	35.26 (1.29)	2.23	$P < .001$	-
	Loop	34.11 (2.18)	1.08	$P < .005$	.008

Table 9: Results obtained experimenting with W2V-250 Word Embeddings on the SST-5 dataset.

7. Acknowledgments

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. The work of the first author is financed by Grant PAID-01-2461 2015, from the Universitat Politècnica de València.

References

[1] Q. Ye, D. Doermann, Text detection and recognition in imagery: A survey, IEEE transactions on pattern analysis and machine intelligence 37 (7) (2015) 1480–1500.

[2] W. Zhao, R. Chellappa, P. J. Phillips, A. Rosenfeld, Face recognition: A literature survey, ACM computing surveys (CSUR) 35 (4) (2003) 399–458.

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

[4] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[5] P. Li, K. Mao, Knowledge-oriented convolutional neural network for causal relation extraction from natural language texts, Expert Systems with Applications 115 (2019) 512–523.

[6] S. Yoo, J. Song, O. Jeong, Social media contents based sentiment analysis and prediction system, Expert Systems with Applications 105 (2018) 102–111.

[7] G. Mujtaba, L. Shuib, N. Idris, W. L. Hoo, R. G. Raj, K. Khowaja, K. Shaikh, H. F. Nweke, Clinical text classification research trends: Systematic literature review and open issues, Expert Systems with Applications.

[8] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.

[9] W. Yin, K. Kann, M. Yu, H. Schütze, Comparative study of cnn and rnn for natural language processing, arXiv preprint arXiv:1702.01923.

[10] B. Pang, L. Lee, S. Vaithyanathan, Thumbs up?: sentiment classification using machine learning techniques, in: Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10,

Association for Computational Linguistics, 2002, pp. 79–86.
[11] P. D. Turney, Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews, in: Proceedings of the 40th annual meeting on association for computational linguistics, Association for Computational Linguistics, 2002, pp. 417–424.
[12] P. Nakov, S. Rosenthal, Z. Kozareva, V. Stoyanov, A. Ritter, t. Wilson, Semeval-2013 task 2: Sentiment analysis in twitter, in: Proceedings of the 7th international workshop on semantic evaluation, 2013, pp. 312–320.
[13] S. Rosenthal, P. Nakov, S. Kiritchenko, S. Mohammad, A. Ritter, V. Stoyanov, Semeval-2015 task 10: Sentiment analysis in twitter, in: Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015), 2015, pp. 451–463.
[14] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, V. Stoyanov, Semeval-2016 task 4: Sentiment analysis in twitter, in: Proceedings of the 10th international workshop on semantic evaluation (semeval-2016), 2016, pp. 1–18.
[15] S. Rosenthal, N. Farra, P. Nakov, Semeval-2017 task 4: Sentiment analysis in twitter, in: Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017, pp. 502–518.
[16] J. Villena Román, S. Lana Serrano, E. Martínez Cámara, J. C. González Cristóbal, Tass-workshop on sentiment analysis at sepln.
[17] F. Barbieri, V. Basile, D. Croce, M. Nissim, N. Novielli, V. Patti, Overview of the evalita 2016 sentiment polarity classification task, in: Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016), 2016.
[18] B. Pang, L. Lee, et al., Opinion mining and sentiment analysis, Foundations and Trends® in Information Retrieval 2 (1–2) (2008) 1–135.
[19] S. M. Mohammad, P. D. Turney, Crowdsourcing a word–emotion association lexicon, Computational Intelligence 29 (3) (2013) 436–465.
[20] S. Kiritchenko, X. Zhu, S. M. Mohammad, Sentiment analysis of short informal texts, Journal of Artificial Intelligence Research 50 (2014) 723–762.
[21] L. K. Hansen, A. Arvidsson, F. Å. Nielsen, E. Colleoni, M. Etter, Good friends, bad news-affect and virality in twitter, in: Future information technology, Springer, 2011, pp. 34–43.
[22] M. Giménez, F. Pla, L.-F. Hurtado, Elirf: a svm approach for sa tasks in twitter at semeval-2015, in: Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), 2015, pp. 574–581.
[23] L. Barbosa, J. Feng, Robust sentiment detection on twitter from biased and noisy data, in: Proceedings of the 23rd international conference on computational linguistics: posters, Association for Computational Linguistics, 2010, pp. 36–44.
[24] B. O'Connor, R. Balasubramanyan, B. R. Routledge, N. A. Smith, et al., From tweets to polls: Linking text sentiment to public opinion time series., Icwsm 11 (122-129) (2010) 1–2.
[25] X. Zhu, S. Kiritchenko, S. Mohammad, Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets, in: Proceedings of the 8th international workshop on semantic evaluation (SemEval 2014), 2014, pp. 443–447.
[26] R. Socher, A. Perelygin, J. Wu, J. Chuang, C. D. Manning, A. Ng, C. Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in: Proceedings of the 2013 conference on empirical methods in natural language processing, 2013, pp. 1631–1642.
[27] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, C. Potts, Learning word vectors for sentiment analysis, in: Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies-volume 1, Association for Computational Linguistics, 2011, pp. 142–150.
[28] T. Mikolov, K. Chen, G. Corrado, J. Dean, Efficient estimation of word representations in vector space, arXiv preprint arXiv:1301.3781.
[29] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), 2014, pp. 1532–1543.
[30] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, arXiv preprint arXiv:1607.04606.
[31] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, `http://www.deeplearningbook.org`.
[32] Y. LeCun, Generalization and network design strategies, in: R. Pfeifer, Z. Schreter, F. Fogelman, L. Steels (Eds.), Connectionism in Perspective, Elsevier, Zurich, Switzerland, 1989.
[33] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, P. Kuksa, Natural language processing (almost) from scratch, The Journal of Machine Learning Research 12 (2011) 2493–2537.
[34] O. Araque, I. Corcuera-Platas, J. F. Sanchez-Rada, C. A. Iglesias, Enhancing deep learning sentiment analysis with ensemble techniques in social applications, Expert Systems with Applications 77 (2017) 236–246.
[35] T. Chen, R. Xu, Y. He, X. Wang, Improving sentiment analysis via sentence type classification using bilstm-crf and cnn, Expert Systems with Applications 72 (2017) 221–230.
[36] Y. Zhang, B. Wallace, A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification, arXiv preprint arXiv:1510.03820.
[37] Y. Kim, Convolutional neural networks for sentence classification, arXiv preprint arXiv:1408.5882.
[38] Y. Shen, X. He, J. Gao, L. Deng, G. Mesnil, Learning semantic representations using convolutional neural networks for web search, in: Proceedings of the companion publication of the 23rd International Conference on World Wide Web., International World Wide Web Conferences Steering Committee, 2014, pp. 373–374.
[39] W. Zhang, K. Itoh, J. Tanida, Y. Ichioka, Parallel distributed processing model with local space-invariant interconnections and its optical architecture, Applied optics 29 (32) (1990) 4790–4797.
[40] W. Zhang, Shift-invariant pattern recognition neural network and its optical architecture, in: Proceedings of annual conference of the Japan Society of Applied Physics, 1988.
[41] S. M. Mohammad, S. Kiritchenko, X. Zhu, Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets, arXiv preprint arXiv:1308.6242.
[42] M. Giménez, D. I. Hernández, F. Pla, Segmenting target audiences: Automatic author profiling using tweets., in: Proceedings of CLEF, 2015.
[43] J. Barnes, R. Klinger, S. S. i. Walde, Assessing state-of-the-art sentiment models on state-of-the-art sentiment datasets, arXiv preprint arXiv:1709.04219.
[44] W. Medhat, A. Hassan, H. Korashy, Sentiment analysis algorithms and applications: A survey, Ain Shams Engineering Journal 5 (4) (2014) 1093–1113.
[45] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `https://www.tensorflow.org/`
[46] Y. Bengio, R. Ducharme, P. Vincent, C. Jauvin, A neural probabilistic language model, Journal of machine learning research 3 (Feb) (2003) 1137–1155.
[47] J. C. De Winter, Using the student's t-test with extremely small sample sizes., Practical Assessment, Research & Evaluation 18 (10).