On the Use of Dependencies in Relation Classification of Text with Deep Learning

Bernard Espinasse[1], Sébastien Fournier[1], Adrian Chifu[1], Gaël Guibon[1], René Azcurra[1], and Valentin Mace

Aix-Marseille Université, Université de Toulon, CNRS, LIS, Marseille, France
firstname.name@lis-lab.fr

Abstract. Deep Learning is more and more used in NLP tasks, such as in relation classification of texts. This paper assesses the impact of syntactic dependencies in this task at two levels. The first level concerns the generic Word Embedding (WE) as input of the classification model, the second level concerns the corpus whose relations have to be classified. In this paper, two classification models are studied, the first one is based on a CNN using a generic WE and does not take into account the dependencies of the corpus to be treated, and the second one is based on a compositional WE combining a generic WE with syntactical annotations of this corpus to classify. The impact of dependencies in relation classification is estimated using two different WE. The first one is essentially lexical and trained on the Wikipedia corpus in English, while the second one is also syntactical, trained on the same previously annotated corpus with syntactical dependencies. The two classification models are evaluated on the SemEval 2010 reference corpus using these two generic WE. The experiments show the importance of taking dependencies into account at different levels in the relation classification.

Keywords: Dependencies · Relation Classification · Deep Learning · Word Embedding · Compositional Word Embedding.

1 Introduction

Deep Learning is more and more used for various task of Natural Language Processing (NLP), such as relation classification from text. It should be recalled that the Deep Learning has emerged mainly with convolutional neural networks (CNNs), originally proposed in computer vision [5]. These CNNs were later used in language processing to solve problems such as sequence labelling [1], semantic analysis - semantic parsing [11], relation extraction, etc.

CNNs are the most commonly used for deep neural network models in the relation classification task. One of the first contribution is certainly the basic CNN model proposed by Lui *et al.* (2013) [7]. Then we can mention the model proposed by Zeng *et al.* (2014) [13] with max-pooling, and the model proposed by Nguyen and Grishman (2015) [10] with multi-size windows. Performance of these CNN-based relation classification models are low in terms of *Precision* and *Recall*. These low performances can be explained by two reasons.

First, despite their success, CNNs have a major limitation in language processing, due to the fact that they were invented to manipulate pixel arrays in image processing, and therefore only take into account the consecutive sequential n-grams on the surface chain. Thus, in relation classification, CNNs do not consider long-distance syntactic dependencies, these dependencies play a very important role in linguistics, particularly in the treatment of negation, subordination, fundamental in the analysis of feelings, etc. [8].

Second, Deep Learning-based relation classification models generally use as input a representation of words obtained by lexical immersion or Word Embedding (WE) with training on a large corpus. Skip-Gram or Continuous Bag-of-Word WEs models, generally only consider the local context of a word, in a window of a few words before and a few words after, this without consideration of syntactic linguistics characteristics. Consequently syntactic dependencies are not taken into account in relation classification using Deep Learning-based models.

As syntactic dependencies play a very important role in linguistics, it makes sense to take them into account for classification or relation extraction. The consideration of these dependencies in Deep Learning models can be carried out at different levels. At a first level, *(Syntaxical Word Embedding)* dependencies are taken into account upstream, at the basic word representation level in a generic WE trained on a large corpus syntactically annotated and generated with specific tool. At a second level, related to the relation classification corpus *(Compositional Word Embedding)*, there is a combination of a generic WE trained on a large corpus with specific features as dependencies, extracted from the words in the sentences of the corpus to classify. This paper assesses the impact of syntactic dependencies in relation classification at these two different levels.

The paper is organized as follows. In Section 2, we first present a generic *syntactical WE* trained on a large corpus that has been previously annotated with syntactical dependencies and considering for each word dependencies, in which it is involved. In Section 3, two Deep Learning models of relation classification are presented. The first model, that we have developed, is based on a CNN using as input a generic WE trained on a large corpus completed by a positional embedding of the corpus to classify. The second model, the FCM model implemented with a neural network of perceptron type, is based on a *compositional WE* strategy, using as input a combination of generic WE with specific syntactical features from the corpus to classify relations. In Section 4 we present the results of experiments obtained with these two relation classification models on the SemEval 2010 reference corpus using different *WEs*. Finally, we conclude by reviewing our work and presenting some perspectives for future research.

2 A Syntactical Word Embedding taking into account Dependencies

In Deep Learning approach, relation classification models generally use as input a representation of the words of a specific natural language obtained by lexical

immersion or Word Embedding (WE). We can distinguish two main WE models: Skip-Gram and Continuous Bag-of-Word. These WEs only consider the local context of a word, in a window of a few words before and a few words after. Syntactic dependencies are not taken into account in these WE models, whereas these syntactic dependencies play a very important role in NLP tasks.

Given a classic Bag-of-Words WE taking into account the neighbours upstream and downstream of a word, according to a defined window, we may consider the following sentence: "*Australian scientist discovers star with telescope*". With a 2-word window WE, the contexts of the word *discovers* are *Australian*, *scientist*, *star* and *with*. This misses the important context of the *telescope*. By setting the window to 5 in the WE, you can capture more topical content, by example the word *t*elescope, but also weaken the importance of targeted information on the target word.

A more relevant word contextualization consists to integrate the different syntactic dependencies in which this word participates, dependencies that can involve words that are very far in the text. The syntactic dependencies that we consider in this paper are those of Stanford (Stanford Dependencies) defined by [2]. Note that syntactic dependencies are both more inclusive and more focused than Bag-of-Words. They capture relations with distant words and therefore out of reach with a small window Bag-of-Words (for example, the discovery instrument is the *telescope/preposition with*), and also filter out incidental contexts that are in the window, but not directly related to the target word (for example, *Australian* is not used as a context for *discovery*).

Levy and Goldberg [6] have proposed a generalization of the Skip-gram approach by replacing Bag-of-Word contexts with contexts related to dependencies. They proposed a generalization of the WE Skip-Gram model in which the linear contexts of Bag-of-Words are replaced by arbitrary contexts. This is especially the case with contexts based on syntactic dependencies, which produces similarities of very different kinds. They also demonstrated how the resulting WE model can be questioned about discriminatory contexts for a given word, and observed that the learning procedure seems to favour relatively local syntactic contexts, as well as conjunctions and preposition objects. Levy and Goldberg [6] developed a variant of *Word2Vec* tool [9], named *Word2Vec-f*[1], based on a such syntactic dependency-based contextualization.

3 Two Models for Relation Classification using syntactical dependencies

In this section, two models for relation classification by supervised Deep Learning are presented and used for our experiments developed in Section 4. Firstly, we developed a model based on CNN using WEs trained on a large corpus. The second, proposed by [4], is based on a *compositional WE* combining a generic WE and a syntactic annotation of the corpus whose relations are to be classified.

[1] Word2Vec-f : https ://bitbucket.org/yoavgo/Word2Vecf

3.1 A CNN based Relation Classification Model (CNN)

The first model, that we have developed, is based on CNN using a generic WE trained on a large corpus. This model is inspired by the one used by Nguyen and Grishman (2015)[10] and it takes as input a WE either using *word2Vec* or *Word2Vec-f* tools and a Positional Embedding relative to the corpus from which we want to extract relations. The architecture of our CNN network (Fig. 1) consists of five main layers:

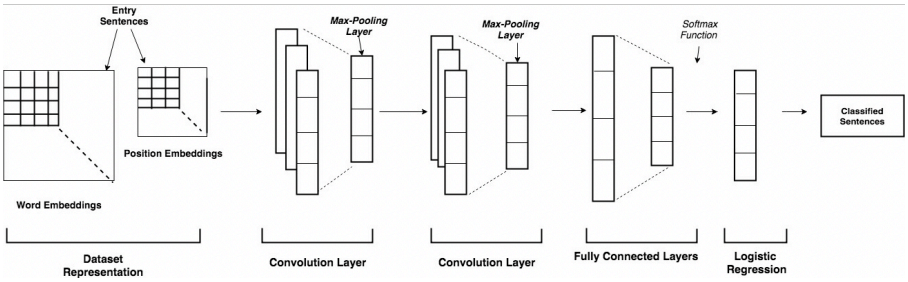

Fig. 1. CNN Architecture.

- *Two convolutional layers* using a number and a size defined for convolutional filters to capture the characteristics of the pretreated input. The filter size is different for each layer. For each layer there are also attached grouping layers (Max Pooling) with an aggregation function (max) for the identification of the most important characteristics produced by the output vector of each convolutional layer.
- *A fully connected layer* that uses the RELU (Rectified Linear Unit) activation function.
- *A fully connected layer* using the Sotfmax activation function to classify the relations to be found.
- *A logistic regression layer* making the optimization of network weighting values with a function to update these values iteratively on the training data. This architecture will be implemented in the *TensorFlow*[2] platform (version 1.8), using the *Tflearn* API[3] that facilitates its implementation and experimentation.

3.2 A Compositional Word Embedding based Relation Classification Model (FCM)

This model, named FCM (for Factor-based Compositional embedding Model) is proposed by Gormley, Yu and Dredze (2014-2015) [4][3]. The FCM model is based on a *compositional WE*, which combines a generic WE trained on a large

[2] TensorFlow : https ://www.tensorflow.org/
[3] Tflearn : http ://tflearn.org/

corpus with specific features at a syntactical level from the corpus to classify relations. More precisely, This *compositional WE* is developed by combining a classic Skip-Gram WE using *Word2Vec* with features extracted from the sentences' corpus words, from which we want to extract relations. This expressive model is implemented through a perceptron-type neuromimetic network [12]. The key idea is to combine/compose a generic lexical WE with non-lexical, arbitrary and manually defined features,especially syntactic ones.

This non-lexical linguistic context is based on arbitrary, hand-defined linguistic structures called HCF (Hand-Crafted Features). These features can be perceived as simple questions addressed to the word and its context, for example, whether the word is an adjective, whether it is preceded by a verb or whether it is an entity in the sentence. In fact, there is a large number of more or less complex features allowing to capture different information.

Thus, the model capitalizes on arbitrary types of linguistic annotations by making better use of the features associated with the substructures of these annotations, including global information. In relation classification the FCM model uses a feature vector fw_i over the word w_i, the two target entities M_1, M_2 , and their dependency path. The main HCF sets used are *HeadEmb, Context, In-between and On-path*. By example the *In-between* features indicate whether a word w_i is in between two target entities. The *On-path* features indicate whether the word is on the dependency path, on which there is a set of words P, between the two entities [4].

The FCM model constructs a representation of text structures using both lexical and non-lexical characteristics, thus creating an abstraction of text sentences based on both generic WE and HCF, capturing relevant information about the text. To construct this representation, the FCM takes a sentence as input and its annotations for each word (generated for example by morphosyntactic labelling) and proceeds in three steps:

– *Step 1*: The FCM first breaks down the annotated sentence into substructures, each substructure is actually a word with its annotations. Fig. 2 illustrates the sentence "*The movie I watched depicted hope*" with its annotations, especially those related to its syntactic dependencies. Its entities are marked with M_1 and M_2 and the dependency paths are indicated by arcs, in blue when they are between entities M_1 and M_2.

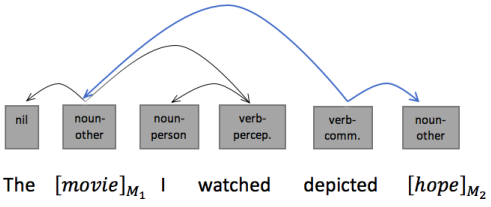

Fig. 2. A sentence and its FCM annotations [4].

– *Step 2*: The FCM extracts the HCFs for each substructure (word), obtaining a binary vector for each word in the sentence. The first *on-path* HCF indicates

for each word whether it is on the dependency path between entities M_1 et M_2. As illustrated in Fig. 3, each word in the previous sentence corresponds to a vector (or column), f_5 representing the HCF of the word *depicted*.

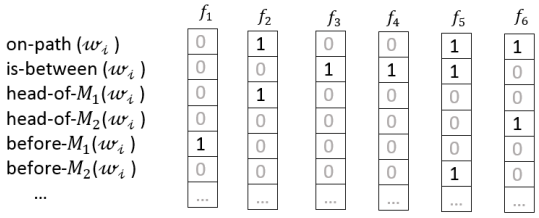

Fig. 3. HCF vector in the FCM model [4].

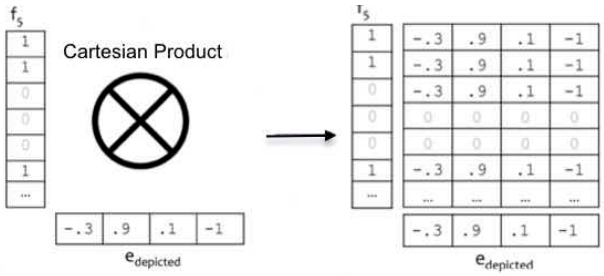

Fig. 4. Embedding Substructure matrix of FCM model for the word "depicted" [4].

– *Step 3*: For each word, the FCM computes the cartesian product, more precisely an inner product, of its HCF vector with its WE vector obtained with Word2Vec. This inner product gives for each word a matrix called *Substructure Embedding* (see Fig. 4). Then the FCM model sums up all the Embedding Substructures (matrices) of the sentence to obtain a final matrix called Annotated Sentence Embedding noted with:

$$e_x = \sum_{i=1}^{n} f_{w_i} \otimes e_{w_i}$$

Thus, the FCM builds an abstraction of the given sentence, first by cutting each word with its annotations (substructure), then by creating for each substructure a matrix of numbers obtained from generic WE and the word features (Substructure Embedding) by summing these matrices to obtain a final matrix that constitutes its representation of the input sentence (Annotated Sentence Embedding).

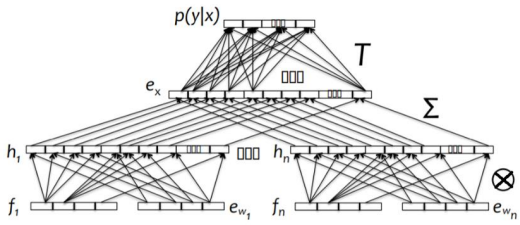

Fig. 5. Neural Network implementing FCM model (Gormley *et al.*, 2015)[4].

The FCM is implemented by a multilayer perceptron neural network, with the architecture presented in Fig. 5. In this network, when a sentence has been provided as an input and transformed into an Annotated Sentence Embedding (number matrix), the network will make a prediction on the y relation between the entities in the sentence. It seeks to determine y knowing $x = (M_1, M_2, S, A)$ with M the entities, S the sentence and A its annotations. A tensor T made up of matrices of the same size as the Annotated Sentence Embedding is used, T will act as a parameter (equivalent to network connections) and serve to establish a score for each possible relation. In relation classification, there is a finite number of possible relations constituting the set L. There are thus as many matrices in T as there are relations in the set of relations L. Ty refers to the score matrix for the relation y.

4 Experiments

This section presents the experiments in relation classification, using the two relation classification models previously presented (CNN and FCM) and different generic WE (Word2Vec and Word2Vec-f) taking or not taking into account syntactical dependencies (. First, we present the SemEval 2010 corpus and then the results obtained with these two models. Finally, we compare these results and try to interpret them.

4.1 SemEVAL 2010 corpus

The SemEVAL 2010 corpus[4] is a manually annotated reference corpus for extracting nominal relations. Let be a sentence and 2 annotated names, it is necessary to choose the most appropriate relation among the following 9 relation classes: *Cause-Effect ; Instrument-Agency ; Cause-Effect ; Instrument-Agency ; Product-producer ; Content-Container ; Entity-Destination ; Component-Whole ; member-Collection ; Message-Topic*. It is also possible to choose the *Other* class if none of the 9 relation classes seem to be suitable. For instance, given the sentence :

"The <e1>macadamia nuts</e1> in the <e2>cake</e2> also make it necessary to have a very sharp knife to cut through the cake neatly."

The best choice for the following sentence would be: *Component-Whole(e1,e2)*. Note that in the sentence *Component-Whole (e1,e2)* is there, but *Component-Whole (e2,e1)* is not there, i.e. we have *Other (e2,e1)*. Thus, it is a question of determining both the relations and the order of e1 and e2 as arguments.

The corpus is composed of 8,000 sentences for training the 9 relations and the additional relation *O*ther. The test data set consists of 2,717 examples from the 9 relations and the additional *Other* relation. More specifically, the test data set contains data for the first 5 relations mentioned above. However, there are some references to the other 4 relations, which can be considered as *O*ther when experimenting with the test data set. Table 1 presents the 19 relations of the SemEval 2010 corpus.

[4] Corpus SemEval : http://www.aclweb.org/anthology/W09-2415

Table 1. Characterization of the different relation used.

#Rel	Relation	Support	#Rel	Relation	Support
R1	Cause-Effect(e1,e2)	150	R11	Instrument-Agency(e1,e2)	108
R2	Cause-Effect(e2,e1)	123	R12	Instrument-Agency(e2,e1)	134
R3	Component-Whole(e1,e2)	194	R13	Member-Collection(e1,e2)	211
R4	Component-Whole(e2,e1)	134	R14	Member-Collection(e2,e1)	22
R5	Content-Container(e1,e2)	32	R15	Message-Topic(e1,e2)	47
R6	Content-Container(e2,e1)	51	R16	Message-Topic(e2,e1)	153
R7	Entity-Destination(e1,e2)	201	R17	Other	162
R8	Entity-Destination(e2,e1)	39	R18	Product-Producer(e1,e2)	291
R9	Entity-Origin(e1,e2)	1	R19	Product-Producer(e2,e1)	454
R10	Entity-Origin(e2,e1)	210		*Total:*	*2717*

4.2 Employed Word Embeddings

We used both *Word2Vec* and *Word2Vec-f* tools to create our own WE, integrating for the later syntactic dependencies, trained throughout the Wikipedia corpus in English. This corpus has been previously annotated with syntactic dependencies using the *SpaCy*[5] analyzer. Note that we use also a generic WE already done with Word2Vec with GoogleNews corpus. But since we did not have the GoogleNews corpus, we were unable to process it with *Word2Vec-f* tool. The Table 2 characterizes the different WE used for the experiments. Note that the WE *W*ikipedia/Word2Vec-f is significantly smaller in terms of vocabulary size than the other WE, by a factor of 7 with *W*ikipedia/Word2Vec and by a factor of 10 with *G*oogleNews/Word2Vec.

Table 2. Characterization of the different Word Embeddings used.

Word Embedding	Vocabulary Size	Window	Dimension
GoogleNews/Word2Vec	3,000,000	5	300
Wikipedia/Word2Vec	2,037,291	5	300
Wikipedia/Word2Vecf	285,378	5	300

4.3 Experiments with the CNN model

The optimal hyperparameters for learning our CNN, quite close to those adopted by Nguyen and Grishman (2015) [10] and by Zeng et. al. (2015)[13] are : Window size: 2, 3 ; Filter Nb.: 384 ; WE size : 300 ; Position size: 5 ; mini batch size: 35 and Dropout: 0.5). The Table 3 gives the results obtained by our CNN model for each relation classes of the SemEval 2010 corpus with the *W*ord2Vec and *W*ord2Vec-f WEs, trained with Wikipedia.

[5] SpaCy : https ://spacy.io/

Table 3. Results for CNN model with Word2Vec (W2V) & Word2Vec-f (W2V-f) WE.

	Precision			Recall			F1-score		
Rel,	**W2V**	**W2V-f**	***Delta %***	**W2V**	**W2V-f**	***Delta %***	**W2V**	**W2V-f**	***Delta %***
R1	0,88	0,93	**5,68**	0,84	0,84	0,00	0,86	0,88	**2,33**
R2	0,83	0,84	**1,20**	0,90	0,90	0,00	0,87	0,87	0,00
R3	0,84	0,76	-9,52	0,64	0,75	**17,19**	0,73	0,76	**4,11**
R4	0,79	0,73	-7,59	0,57	0,64	**12,28**	0,66	0,68	**3,03**
R5	0,76	0,75	-1,32	0,84	0,82	-2,38	0,80	0,78	-2,50
R6	0,85	0,83	-2,35	0,72	0,77	**6,94**	0,78	0,80	**2,56**
R7	0,87	0,79	-9,20	0,86	0,90	**4,65**	0,86	0,84	-2,33
R8	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
R9	0,79	0,73	-7,59	0,81	0,89	**9,88**	0,80	0,80	0,00
R10	0,90	0,89	-1,11	0,81	0,85	**4,94**	0,85	0,87	**2,35**
R11	0,67	0,59	-11,94	0,27	0,45	**66,67**	0,39	0,51	**30,77**
R12	0,69	0,63	-8,70	0,51	0,61	**19,61**	0,59	0,62	**5,08**
R13	0,70	0,47	-32,86	0,44	0,59	**34,09**	0,54	0,53	-1,85
R14	0,74	0,82	**10,81**	0,93	0,88	-5,38	0,82	0,85	**3,66**
R15	0,75	0,79	**5,33**	0,79	0,80	**1,27**	0,77	0,79	**2,60**
R16	0,76	0,78	**2,63**	0,49	0,69	**40,82**	0,60	0,73	**21,67**
R17	0,39	0,48	**23,08**	0,51	0,40	-21,57	0,45	0,44	-2,22
R18	0,75	0,77	**2,67**	0,62	0,69	**11,29**	0,68	0,73	**7,35**
R19	0,67	0,70	**4,48**	0,50	0,67	**34,00**	0,57	0,69	**21,05**
Mean	*0,72*	*0,72*	0,00	*0,71*	*0,72*	**1,41**	*0,71*	*0,73*	**2,82**

Globally, for all relations, for *Word2Vec* and *Word2Vec-f* WEs, the following F-measure values are obtained: *Word2Vec*: F-measurement (macro: 0.663; micro: 0.705; weighted: 0.707). *Word2Vec-f*: F-measurement (macro: 0.692 ; micro: 0.728 ; weighted: 0.722). Thus, for the SemEval 2010 corpus, the CNN model obtains the best results with the WE *Wikipedia/Word2Vec-f* taking into account syntactic dependencies, there is an improvement of about 10% compared to the use of the WE *Wikipedia/Word2Vec* not taking into account dependencies.

4.4 Experiments with the FCM Model

There are several implementations of the FCM model, the one developed in Java by M. Gromley in his thesis[6] and the one of M. Yu, developed in C++[7]. It is the latter, more efficient, that have been used in our experiments. For the experiments with the FCM model, the best results were obtained with a learning rate set at 0.005 and a number of epochs at 30, without early-stopping.

We tested the FCM model on the SemEval 2010 corpus using several WEs, obtained with Word2Vec and Word2Vec-f tools, and trained with Wikipedia in English (treated before with *SpaCy* for *Word2Vec-f*) in sizes 200 and 300 with

[6] https://github.com/mgormley/pacaya-nlp
[7] https://github.com/Gorov/FCM_nips_workshop

Table 4. Results for FCM model with Word2Vec (W2V) & Word2Vec-f (W2V-f) WE.

	Precision			Recall			F1-score		
#	W2V	W2V-f	*Delta %*	W2V	W2V-f	*Delta %*	W2V	W2V-f	*Delta %*
R1	0,77	0,79	**1,85**	0,78	0,77	-0,86	0,78	0,78	0,48
R2	0,80	0,80	0,65	0,74	0,76	**3,30**	0,77	0,78	**2,01**
R3	0,91	0,91	-0,04	0,91	0,90	-0,56	0,91	0,91	-0,32
R4	0,76	0,78	**2,65**	0,81	0,75	-7,41	0,78	0,76	-2,47
R5	0,83	0,81	-1,55	0,75	0,69	-8,33	0,79	0,75	-5,22
R6	0,78	0,76	-1,39	0,75	0,76	**2,63**	0,76	0,76	0,62
R7	0,81	0,83	**2,43**	0,92	0,91	-1,62	0,86	0,87	0,49
R8	0,88	0,86	-2,47	0,74	0,77	**3,44**	0,81	0,81	0,65
R9	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00	0,00
R10	0,83	0,84	**0,98**	0,86	0,89	**3,32**	0,85	0,87	**2,11**
R11	0,81	0,82	**1,33**	0,75	0,81	**7,41**	0,78	0,81	**4,40**
R12	0,93	0,90	-2,99	0,93	0,92	-0,81	0,93	0,91	-1,91
R13	0,82	0,83	1,51	0,84	0,87	**3,37**	0,83	0,85	**2,42**
R14	0,60	0,67	**11,12**	0,41	0,45	**11,10**	0,49	0,54	**11,10**
R15	0,86	0,86	0,00	0,79	0,79	0,00	0,82	0,82	0,00
R16	0,82	0,82	-0,34	0,87	0,88	**1,51**	0,84	0,85	0,56
R17	0,83	0,85	**1,73**	0,80	0,79	-1,55	0,82	0,82	0,04
R18	0,84	0,88	**4,26**	0,91	0,91	-0,38	0,87	0,89	**1,98**
R19	0,55	0,56	**2,30**	0,50	0,54	**7,43**	0,53	0,55	**4,91**
Mean	*0,78*	*0,79*	1,36	*0,79*	*0,79*	1,12	*0,78*	*0,79*	1,29

a window of 5. The Table 4 shows the results obtained. Table 5 gives the F1 measures of the FCM model for various WEs.

Table 5. Measures obtained by FCM model with different Word Embeddings.

Word Embedding	**Macro F1**	**Micro F1**	**Weighted F1**
GoogleNews vect. neg-dim300	0.746	0.789	0.781
Wikipedia/Word2Vec	*0.747*	*0.7858*	*0.782*
Wikipedia/Word2Vec-f	*0.744*	*0.794*	*0.792*

If we compare the Macro F1-Scores obtained for WE Wikipedia/Word2Vec-dim300 and Wikipedia/Word2Vecf-dim300, we obtain a gain of 0.88%. It should be noted that the results of the FCM model obtained on the Semeval 2010 corpus are slightly different from those announced by the authors of the FCM in their article (Gormley *et al.*, 2015)[4], because the latter do not take into account the *Other* relation class of the corpus.

4.5 Discussion

The Table 6 compares results of the two relation classification models, the CNN model (generic WE + CNN) and the FCM model (compositional WE) on SemEval 2010 corpus (all relation classes are considered).

Table 6. All classes F1-measures for SemEval 2010 for the two classification models with different Word Embeddings.

Model/WE	Macro F1	Micro F1	Weighted F1
CNN/Word2Vec	0.663	0.705	0.707
CNN/Word2Vec-f	0.692	0.728	0.722
FCM/Word2Vec	*0.747*	*0.785*	*0.782*
FCM/Word2Vec-f	*0.754*	*0.794*	*0.792*

The CNN model (WE + CNN) has very slightly improved performance when the WE takes into account the syntactic dependencies (*Word2Vec-f*) of the large training corpus (Wikipedia). The FCM model (compositional WE + NN), which takes into account syntactic dependencies at the corpus level whose relations have to be classified in its compositional WE, has much better performances than the CNN model (WE + CNN), the FCM model has performances higher than 30%. These performances are slightly improved if we use in the FCM model a WE taking into account the dependencies of the training corpus (with *Word2Vec-f.*)

5 Conclusion

Classification of relations between entities remains a complex task. This article focused on relation classification from a corpus of texts by Deep Learning while considering or not syntactic dependencies at different levels. Two levels of consideration have been distinguished: the generic WE at the input of the classification model, and at the level of the corpus whose relationships are to be classified. Two classification models were studied, the first one is based on a CNN and does not take into account the dependencies of the corpus to be treated and the second one is based on a *compositional WE* combining a generic WE and a syntactic annotation of this corpus.

The impact has been estimated with two generic WE: the first one trained on the Wikipedia corpus in English with the *Word2Vec* tool and the second one trained on the same corpus, previously annotated with syntactic dependencies and generated by the *Word2Vec-f* tool.

The results of our experiments on SemEval 2010 corpus show, first of all, that taking dependencies into account is beneficial for the relation classification task, whatever the classification model used. Then, taking them into account at the level of the corpus to be processed, through a *compositional Word Embedding* is more efficient and results are further slightly improved by using the WE *Word2Vec-f* in input.

Finally, let us recall that a major interest of WE contextualizing words according to syntactic dependencies in which it intervenes is their concision in terms of vocabulary. They can be 7 to 10 times more compact and therefore more efficient regardless of the classification model.

References

1. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. Journal of Machine Learning Research **12**(Aug), 2493–2537 (2011)
2. De Marneffe, M.C., Manning, C.D.: The stanford typed dependencies representation. In: Coling 2008: proceedings of the workshop on cross-framework and cross-domain parser evaluation. pp. 1–8. Association for Computational Linguistics (2008)
3. Gormley, M.R.: Graphical Models with Structured Factors, Neural Factors, and Approximation-Aware Training. Ph.D. thesis, Johns Hopkins University (2015)
4. Gormley, M.R., Yu, M., Dredze, M.: Improved relation extraction with feature-rich compositional embedding models. In: Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing. pp. 1774–1784. Association for Computational Linguistics (2015)
5. LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Muller, U., Sackinger, E., et al.: Comparison of learning algorithms for handwritten digit recognition. In: International Conference on Artificial Neural Networks. vol. 60, pp. 53–60. Perth, Australia (1995)
6. Levy, O., Goldberg, Y.: Dependency-based word embeddings. In: Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). pp. 302–308. Association for Computational Linguistics (2014)
7. Liu, C., Sun, W., Chao, W., Che, W.: Convolution neural network for relation extraction. In: International Conference on Advanced Data Mining and Applications. pp. 231–242. Springer (2013)
8. Ma, M., Huang, L., Zhou, B., Xiang, B.: Dependency-based convolutional neural networks for sentence embedding. In: ACL (2). pp. 174–179. The Association for Computer Linguistics (2015)
9. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in neural information processing systems. pp. 3111–3119 (2013)
10. Nguyen, T.H., Grishman, R.: Relation extraction: Perspective from convolutional neural networks. In: Proceedings of the 1st Workshop on Vector Space Modeling for Natural Language Processing. pp. 39–48 (2015)
11. tau Yih, W., 0001, X.H., Meek, C.: Semantic parsing for single-relation question answering. In: ACL (2). pp. 643–648. The Association for Computer Linguistics (2014)
12. Yu, M., Gormley, M.R., Dredze, M.: Factor-based compositional embedding models. In: In NIPS Workshop on Learning Semantics (2014)
13. Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J.: Relation classification via convolutional deep neural network. In: Hajic, J., Tsujii, J. (eds.) COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland. pp. 2335–2344. ACL (2014)