SyntNN at SemEval-2018 Task 2: is Syntax Useful for Emoji Prediction? Embedding Syntactic Trees in Multi Layer Perceptrons

Andrea Santilli
Department of Enterprise Engineering
University of Rome Tor Vergata
Italy
andrea.santilli@live.it

Fabio Massimo Zanzotto
Department of Enterprise Engineering
University of Rome Tor Vergata
Italy
fabio.massimo.zanzotto@uniroma2.it

Abstract

In this paper, we present *SyntNN* as a way to include traditional syntactic models in multilayer neural networks used in the task of Semeval Task 2 of emoji prediction (Barbieri et al., 2018). The model builds on the distributed tree embedder also known as distributed tree kernel (Zanzotto and Dell'Arciprete, 2012). Initial results are extremely encouraging but additional analysis is needed to overcome the problem of overfitting.

1 Introduction

Syntactic models of language have always played a crucial role in many natural language processing tasks but, in recent years, it has been marginalized by the advent of neural networks and in particular long-short term memory (LSTM). These latter networks have had a tremendous impact on how linguistic tasks are modeled and, sometimes, solved.

In this paper, we want to explore the use of "traditional" syntactic information within a neural network framework in the task of Semeval Task 2 of emoji prediction (Barbieri et al., 2018, 2017). We propose *SyntNN* as a way to include traditional syntactic models in multilayer neural networks. The model builds on the distributed tree embedder also known as distributed tree kernel (Zanzotto and Dell'Arciprete, 2012; Ferrone and Zanzotto, 2014; Zanzotto et al., 2015) that is a way to transpose syntactic information in small vectors. Initial results are extremely encouraging: *SyntNN* outperforms syntax-unaware neural networks on the trial set. Unfortunately, these promising results are not confirmed on the test set. Hence, we analyzed these results to try to understand why this has happened.

2 SyntNN: a Syntax-aware Multilayer Perceptron

SyntNN is a simple, non-recurrent neural network that aims to exploit traditional syntactic interpretations of tweets in classification tasks. This network wants to explore two questions: first, investigating whether "traditional" syntactic interpretation can be used within neural networks and, second, understanding if syntactic information is useful in modeling tweets for the specific task of emoji prediction.

The architecture of *SyntNN* is then organized around two sub-networks (see Fig. 1): (1) a syntactic sub-network and (2) a semantic sub-network. These two sub-networks are then merged to obtain the final classification layer.

The rest of the section describes the two sub-networks and the merging layer.

2.1 Syntactic Subnetwork with a Distributed Tree Embedding Layer

The syntactic subnetwork aims to take syntactic interpretations of tweets and make them available to a simple, non-recurrent neural network. The key point is how to obtain the transformation of the symbolic representation of syntactic trees into tensor-based representations that have meaningful properties.

The *Distributed Tree Embedding* Layer (DTE) (see Fig. 1) is the core component of the syntactic subnetwork. DTE is based on a technique to embed the structural information of syntactic tree into dense, low-dimensional vectors of real numbers (Zanzotto and Dell'Arciprete, 2012; Ferrone and Zanzotto, 2014; Zanzotto et al., 2015). This technique has been originated as a way to replace syntactic kernel functions (Collins and Duffy, 2002) in kernel machines (Cristianini and Shawe-Taylor, 2000) but it can be seen as a principled way to

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 477–481
New Orleans, Louisiana, June 5–6, 2018.

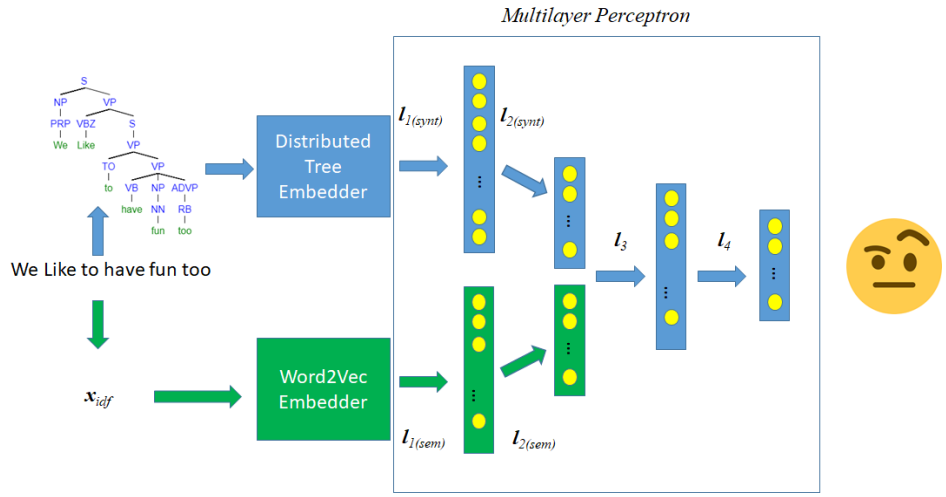

Figure 1: MultiLayer Perceptron Architecture for Syntactic and Semantic Representation of Tweets.

insert syntactic information into vectors. In this technique, tree are transformed into *distributed trees* that are low dimensional vectors. These distributed trees build on the recently revitalized research in Distributed Representations (DR) (Hinton et al., 1986; Plate, 1994; Bengio, 2009; Collobert et al., 2011; Socher et al., 2011).

DTE is a layer that maps trees into low-dimensional vectors in a space $\mathbb{R}^d$. This space is a low dimensional space that embeds the space representing trees according to their subtrees. DTE is then represented as the following embedding layer:

$$\mathbf{y} = W^{(DTE)}S(T) \quad (1)$$

where $S(T) = \mathbf{t}$ is a onehot layer that maps trees to vectors in the space of subtrees $\mathbb{R}^n$ and $W^{(DTE)}_{d\times n}$ is a transformation matrix that embeds the huge space of subtrees $\mathbb{R}^n$ in a smaller space $\mathbb{R}^d$.

DTE is based on the Johnson-Lindenstrauss Tranform (JLT) (Johnson and Lindenstrauss, 1984; Dasgupta and Gupta, 1999) and it is not learned. Using JLT, the layer DTE maps vectors representing trees in the space of subtrees in vectors in a reduced space where similarity is approximately preserved, that is, given two syntactic trees T_1 and T_2 and given a ϵ, it is possible to find a $\mathbf{W}$ for which, if $\mathbb{R}^d$ has a specific relation with $\mathbb{R}^n$ (see (Dasgupta and Gupta, 1999)), this property holds:

$$|\mathbf{t_1}-\mathbf{t_2}|-\epsilon < |\mathbf{W}\mathbf{t_1}-\mathbf{W}\mathbf{t_2}| < |\mathbf{t_1}-\mathbf{t_2}|+\epsilon \quad (2)$$

where $\mathbf{t_1} = S(T_1)$ and $\mathbf{t_2} = S(T_2)$. Hence, the space $\mathbb{R}^d$ embeds the space $\mathbb{R}^n$ of the subtrees.

However, directly producing a DTE with JLT is unfeasible as the space of subtrees $\mathbb{R}^n$ is huge and intractable.

Our solution is using the recursive model for computing distributed trees (Zanzotto and Dell'Arciprete, 2012; Zanzotto et al., 2015), which empirically guarantees the property in Equation 2. This model is a mapping function that encodes trees in vectors by assigning random vectors to node labels and by recursively computing vectors for subtrees by composing vectors for nodes. The mapping function has a linear complexity with respect to the size of the tree.

After the DTE, the syntactic subnetwork has two dense layers with ReLU activation functions. These dense layers should select subtrees that are relevant for the final task of emoji prediction.

As it is designed, the syntactic subnetwork should take into consideration the syntactic information of tweets and it should be a valuable model to experiment with syntactic information on the task of emoji prediction.

2.2 Semantic Subnetwork with a (Bag-of-)Word Embedding Layer

The semantic subnetwork is composed by a classic multilayer perceptron (MLP) network that takes as input tweets represented as $\mathbf{x}_{idf}$. These vectors represent tweets in the following way. Each dimension represents a word w and, if a tweet contains the word w, the dimension has the inverse document frequency (idf) value of the word w, otherwise it has a 0. Hence, the first layer of the semantic subnetwork is a word embedding layer

working in the following way:

$$y = E\mathbf{x}_{idf} \quad (3)$$

where E is a word embedding matrix. As word embeddings, we used the Stanford's Glove (twitter 27B, 200d) for the English task and the word embedding used in the paper *How Cosmopolitan Are Emojis?*(Barbieri et al., 2016)[1] for the Spanish task.

2.3 Merging Syntactic and Semantic Subnetworks

The merging layer of the syntactic and semantic subnetworks is composed by a concatenate layer that concatenate the syntactic vector with the semantic vector among the features axis. Then, a batch normalization layer performs the operation of batch normalization (Ioffe and Szegedy, 2015). At the end a 20 units layer compute the emoji's probability through a softmax layer.

3 Experiments and Evaluation

3.1 Experimental Setting

To ensure replicability, this section fully describes the implementation details of $SyntNN$ (Fig. 1) and the values of its metaparameters. Moreover, it introduces the networks used for comparison and the datasets used on the experiments.

For the *Syntactic Subnetwork* of SyntNN, we used the Python implementation of the Distributed Tree Encoder[2]. Tweets' parse trees are obtained by using Stanford's CoreNLP[3] probabilistic context free grammar parser. Distributed trees are represented in a space $\mathbb{R}^d$ with $d = 4000$. Then, the layer $l_{1(synt)}$ is composed of 5512 units. The layer $l_{2(synt)}$ is a cascade of two dense layers composed, respectively, of 2018 units and 1024 units. All these tree layers have dropout 0.5 and a ReLU activation function.

For the *Semantic Subnetwork* of SyntNN, we used pretrained word embeddings as Stanford's Glove[4] and the word embeddings given by the organizers of the task (Barbieri et al., 2016)[5]. The rest of the semantic subnetwork is the following. The first layer, the *input layer* I, is composed by 200/300 neurons. Each neuron take in input a dimension of the BoW vector. The number of input neuron varies according to the word embedding used: 200 if the word embedding used is Glove; 300 if the word embedding used in the other word embedding cited in the word embedding section. The second layer $l_{1(sem)}$ consists of 512 neurons, dropout 0.5 and ReLU activation function. The third layer $l_{2(sem)}$ consists of 1024 neurons, dropout 0.5 and ReLU activation function.

To understand whether SyntNN positively uses syntactic information, we compared our system with two neural networks trained in comparable conditions: (1) BOW-MLP and (2) BiLSTM. BOW-MLP is basically the Semantic Subnetwork of SyntNN without the Syntactic Subnetwork. BiLSTM is a bidirectional LSTM (Huang et al., 2015), which has been proven effective in many natural language processing tasks. For the BiLSTM, we used the same embedding layer used in SyntNN, we used a recurrent layer of 100 Bidirectional Long Short Term Memory (LSTM) neurons with activation function tanh, recurrent activation function hard sigmoid, recurrent dropout and dropout probability 0.5 and weight l2 regularizer with $\lambda = 0.01$. The output layer is composed by 20 neurons and activation function softmax.

All models are implemented using Keras library (Chollet et al., 2015) and run on tensorflow (Abadi et al., 2015) back-end on different cuda GPUs. Models are trained with Adam(Kingma and Ba, 2014) gradient descent algorithm with $lr = 0.0001, \beta_1 = 0.9, \beta_2 = 0.999$. The loss function used is the categorical crossentropy function. The BOW-MLP model and SyntNN model are trained for 140 epochs and batch size = 50, while the BiLSTM model is trained for 18 epochs and batch size = 50.

We performed our tests on the emoji prediction dataset and we used the Macro F1 Score evaluator provided by the organizers (Barbieri et al., 2018). No additional datasets have been used.

3.2 Results and Analysis

Initial experiments (Table 1), performed using the trial dataset as testing set, are extremely positive with respect to our research question: syntactic information is definitely important for both languages and SyntNN seems to be the good way to represent syntactic relations among words. In

[1] https://github.com/fvancesco/acmmm2016
[2] https://github.com/lorenzoferrone/pyDTK
[3] https://stanfordnlp.github.io/CoreNLP/
[4] https://nlp.stanford.edu/projects/glove/
[5] https://github.com/fvancesco/acmmm2016

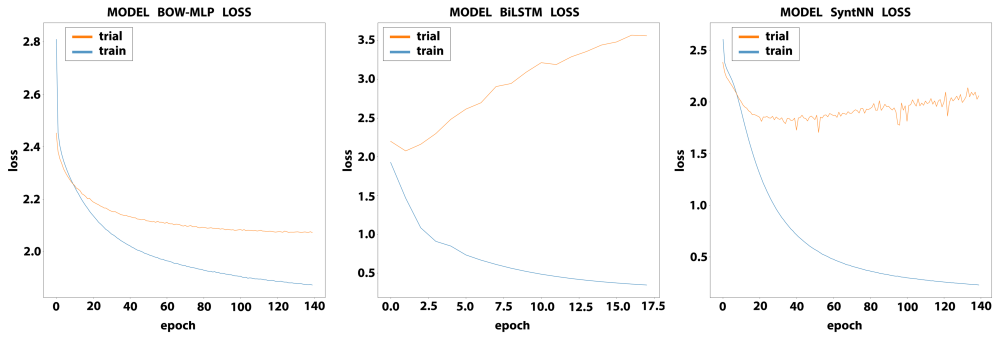

Figure 2: Loss comparison on English datasets.

Dataset	*BOW-MLP*	*BiLSTM*	*SyntNN*
en	30.832	47.535	**61.777**
es	74.077	72.875	**80.474**

Table 1: Macro F1 score on the Trial Set.

Dataset	*BOW-MLP*	*BiLSTM*	*SyntNN*
en	16.298	25.877	18.385
es	15.427	15.008	14.99

Table 2: Macro F1 score on the Test Set.

fact, SyntNN largely outperformed BOW-MLP in both languages: 61.777 vs. 30.832 for the English dataset (en) and 80.474 vs. 74.077 for the Spanish dataset (es). Moreover syntactic information captured by SyntNN seems to be totally different and more effective than the structural information exploited by recurrent neural networks as BiLSTMs. SyntNN has different results with respect to BiLSTM on both datasets with SyntNN outperforming BiLSTM (61.777 vs. 47.535 for *en* and 80.474 vs. 72.875 for *es*). These results seem to show that syntactic information is useful and distributed tree embedders are a possible, effective way to take into consideration syntactic information in multilayer perceptrons.

Surprisingly, results on the official test set did not confirm results on the trial set (Table 2). The first observation is that Macro F1 scores on the test set are definitely lower of the results obtained with different models on the trial set. Moreover, the relative rank among the models is not fully respected. In fact, SyntNN outperforms BOW-MLP only for the *en* dataset but it is definitely worser than BiLSTM. Whereas, models are performing similarly for the *es* dataset. The question is: *Why? What happened?*

Dataset	*Test*	*Trial*
en	44.10%	19.81%
es	40.67%	7.21%

Table 3: Out-of-vocabulary words in the different datasets.

We then tried to analyze where the poor results on the test set came from. The first observation is that unknown words in the test set (Table 3) are larger than for the trial set for both datasets. The unknown words on the test set is more than double with respect to the unknown words in the trial for the English dataset and more than 4 times for the Spanish dataset. Test is definitely farer than trial with respect to the training set. This seems to be the first reason why results are poorer on the test set. The second observation is on the degree of overfitting. In fact, this seems to to be the major problem of SyntNN and of the other two models (see Fig. 2). By looking at the loss function, three models largely overfit with respect to the epochs: the loss functions on the train set and on the trial set diverge. This can partially explain poor results.

However, to have a more in-dept analysis we need to know what and how these networks are modelling symbols in general and syntactic information in particular.

4 Conclusions

In this paper we presented a way to include traditional syntactic information in neural networks and we experimented with this model within the emoji prediction task. Although results on the test set does not confirm results on the trial set, this approach is promising as it opens to an higher explainability of decisions of the neural network (Zanzotto and Ferrone, 2017).

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 105–111. Association for Computational Linguistics.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa-Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, United States. Association for Computational Linguistics.

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *Proceedings of the 2016 ACM on Multimedia Conference*, pages 531–535. ACM.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Foundations and Trends in Machine Learning*, 2(1):1–127.

François Chollet et al. 2015. Keras. `https://github.com/keras-team/keras`.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the Conference of the Annual Meeting of the Association of Computational Linguistics*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Sanjoy Dasgupta and Anupam Gupta. 1999. An elementary proof of the johnson-linderstrauss lemma. Technical Report TR-99-006, ICSI, Berkeley, California.

Lorenzo Ferrone and Fabio Massimo Zanzotto. 2014. Towards syntax-aware compositional distributional semantic models. In *COLING 2014, 25th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, August 23-29, 2014, Dublin, Ireland*, pages 721–730.

Geoffrey E. Hinton, James L. McClelland, and David E. Rumelhart. 1986. Distributed representations. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*. MIT Press, Cambridge, MA.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging. *CoRR*, abs/1508.01991.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167.

W. Johnson and J. Lindenstrauss. 1984. Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, 26:189–206.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Tony A. Plate. 1994. *Distributed Representations and Nested Compositional Structure*. Ph.D. thesis.

Richard Socher, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*.

F. M. Zanzotto and L. Ferrone. 2017. Can we explain natural language inference decisions taken with neural networks? inference rules in distributed representations. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 3680–3687.

Fabio Massimo Zanzotto and Lorenzo Dell'Arciprete. 2012. Distributed tree kernels. In *Proceedings of International Conference on Machine Learning*, pages 193–200.

Fabio Massimo Zanzotto, Lorenzo Ferrone, and Xavier Carreras. 2015. Decoding distributed tree structures. In *Statistical Language and Speech Processing*, pages 73–83, Cham. Springer International Publishing.