ITAmoji 2018: Emoji Prediction via Tree Echo State Networks

Daniele Di Sarli, Claudio Gallicchio, Alessio Micheli
Department of Computer Science
University of Pisa, Pisa, Italy
d.disarli@studenti.unipi.it, {gallicch,micheli}@di.unipi.it

CEUR-WS.org/Vol-2263/paper019.pdf

Abstract

English. For the "ITAmoji" EVALITA 2018 competition we mainly exploit a Reservoir Computing approach to learning, with an ensemble of models for trees and sequences. The sentences for the models of the former kind are processed by a language parser and the words are encoded by using pretrained FastText word embeddings for the Italian language. With our method, we ranked 3rd out of 5 teams.

Italiano. *Per la competizione EVALITA 2018 sfruttiamo principalmente un approccio Reservoir Computing, con un ensemble di modelli per sequenze e per alberi. Le frasi per questi ultimi sono elaborate da un parser di linguaggi e le parole codificate attraverso degli embedding FastText preaddestrati per la lingua italiana. Con il nostro metodo ci siamo classificati terzi su un totale di 5 team.*

1 Introduction

Echo State Networks (Jaeger and Haas, 2004) are an efficient class of recurrent models under the framework of Reservoir Computing (Lukoševičius and Jaeger, 2009), where the recurrent part of the model ("reservoir") is carefully initialized and then left untrained (Gallicchio and Micheli, 2011). The only weights that are trained are part of a usually simple readout layer[1]. Echo State Networks were originally designed to work on sequences, however it has been shown how to extend them to deal with recursively structured data, and trees in particular, with Tree Echo State Networks (Gallicchio and Micheli, 2013), also referred to as TreeESNs.

We follow this approach for solving the ITAmoji task in the EVALITA 2018 competition (Ronzano et al., 2018). In particular, we parse the input texts into trees resembling the grammatical structure of the sentences, and then we use multiple TreeESN models to process the parse trees and make predictions. We then merge these models by using an ensemble to make our final predictions.

[1]Trained in closed form, e.g. by Moore-Penrose pseudo-inversion, or Ridge Regression.

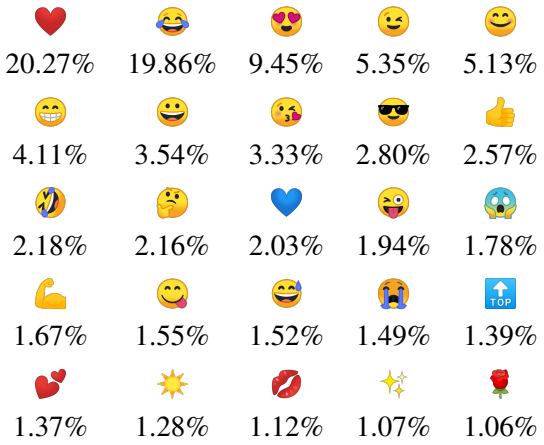

Figure 1: Emojis under consideration and their frequency within the dataset.

2 Task and Dataset

Given a set of Italian tweets, the goal of the ITAmoji task is to predict the most likely emoji associated with each tweet. The dataset contains 250,000 tweets in Italian, each of them originally containing only one (possibly repeated) of the 25 emojis considered in the task (see Figure 1). The emojis are removed from the sentences and used as targets.

The test dataset contains 25,000 tweets similarly processed.

3 Preprocessing

The provided dataset has been shuffled and split into a training set (80%) and a validation set (20%).

We preprocessed the data by first removing any URL from the sentences, as most of them did not contain any informative content (e.g. "`https://t.co/M3StiVOzKC`"). We then parsed the sentences by using two different parsers for the Italian language: Tint[2] (Palmero Aprosio and Moretti, 2016) and spaCy (Honnibal and Johnson, 2015). This produced two sets of trees, both including information about the dependency relations between the nodes of each tree. We finally replace each word with its corresponding pretrained FastText embedding (Joulin et al., 2016).

4 Description of the system

Our ensemble is composed by 13 different models, 12 of which are TreeESNs and the other one is a Long Short-Term Memory (LSTM) over characters. Different random initializations ("trials") of the model parameters are all included in the ensemble in order to enrich the diversity of the hypotheses. We summarize the entire configuration in Table 1.

4.1 TreeESN models

The TreeESN that we are using is a specialization of the description given by Gallicchio and Micheli (2013), and the reader can refer to that work for additional details. Here, the state corresponding to node n of an input tree $\mathbf{t}$ is computed as:

$$\mathbf{x}(n) = f\left(\mathbf{W}_{in}\mathbf{u}(n) + \frac{1}{k}\sum_{i=1}^{k}\hat{\mathbf{W}}_i^n\mathbf{x}(ch_i(n))\right), \tag{1}$$

where $\mathbf{u}(n)$ is the label of node n in the input tree, k is the number of children of node n, $ch_i(n)$ is the i-th child of node n, $\mathbf{W}_{in}$ is the input-to-reservoir weight matrix, $\hat{\mathbf{W}}_i^n$ is the recurrent reservoir weight matrix associated to the grammatical relation between node n and its i-th child, and f is the element-wise applied activation function of the reservoir units (in our case, it is a $\tanh$). All matrices in Equation 1 are left untrained.

[2]Emitting data in the CoNLL-U format (Nivre et al., 2016), a revised version of the CoNLL-X format (Buchholz and Marsi, 2006).

Note that Equation 1 determines a recursive application (bottom-up visit) over each node of the tree $\mathbf{t}$ until the state for all nodes is computed, which we can express in structured form as $\mathbf{x}(\mathbf{t})$. The resulting tree $\mathbf{x}(\mathbf{t})$ is then mapped into a fixed-size feature representation via the χ state mapping function. We make use of *mean* and *sum* state mapping functions, respectively yielding the mean and the sum of all the states. The result, $\chi(\mathbf{x}(\mathbf{t}))$, is then projected into a different space by a matrix $\mathbf{W}_\phi$:

$$\hat{\mathbf{y}} = f_\phi\left(\mathbf{W}_\phi\ \chi(\mathbf{x}(\mathbf{t}))\right), \tag{2}$$

where f_ϕ is an activation function.

For the readout we use both a linear regression approach with L2 regularization known as Ridge regression (Hoerl and Kennard, 1970) and a multilayer perceptron (MLP):

$$\mathbf{y} = \text{readout}(\hat{\mathbf{y}}), \tag{3}$$

where $\mathbf{y} \in \mathbb{R}^{25}$ is the output vector, which represents a score for each of the classes: the index with the highest value corresponds to the most likely class.

4.2 CharLSTM model

The CharLSTM model uses a bidirectional LSTM (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005) with 2 layers, which takes as input the characters of the sentences expressed as pretrained character embeddings of size 300. The LSTM output is then fed into a linear layer with 25 output units.

Similar models have been used in recent works related to emoji prediction, see for example the model used by Barbieri et al. (2017), or the one by Baziotis et al. (2018), which is however a more complex word-based model.

4.3 Ensemble

We take into consideration two different ensembles, both containing the models in Table 1, but with different strategies for weighting the N_P predictions. In the following, let $\mathbf{Y} \in \mathbb{R}^{N_P \times 25}$ be the matrix containing one prediction per row.

The weights for the first ensemble (corresponding to the run file `run1.txt`) have been produced by a random search: at each iteration we compute a random vector $\mathbf{w} \in \mathbb{R}^{N_P}$ with entries sampled from a random variable W^2, $W \sim \mathcal{U}[0, 1]$. The square increases the probability of sampling

#	Class	Reservoir units	f_ϕ	Readout	Parser	Trials
1	TreeESN	1000	ReLU	MLP	Tint	10
2	TreeESN	1000	Tanh	MLP	Tint	10
3	TreeESN	5000	Tanh	MLP	Tint	1
4	TreeESN	5000	Tanh	MLP	spaCy	2
5	TreeESN	5000	ReLU	MLP	Tint	1
6	TreeESN	5000	ReLU	MLP	spaCy	1
7	TreeESN	5000	Tanh	Ridge regression	Tint	1
8	TreeESN	5000	Tanh	Ridge regression	spaCy	3
9	TreeESN	5000	ReLU	Ridge regression	Tint	1
10	TreeESN	5000	ReLU	Ridge regression	spaCy	3
11	TreeESN	5000	Tanh	Ridge regression	Tint	1
12	TreeESN	5000	Tanh	Ridge regression	spaCy	2
13	CharLSTM	–	–	–	–	1

Table 1: Composition of the ensemble, highlighting the differences between the models.

near-zero weights. After selecting the best configuration on the validation set, the predictions from each of the models are merged together in a weighted mean:

$$\bar{\mathbf{y}} = \mathbf{w}\mathbf{Y} \tag{4}$$

For the second type of ensemble (corresponding to the run file `run2.txt`) we adopt a multilayer perceptron. We feed as input the N_P predictions concatenated into a single vector $\mathbf{y}^{(1\dots N_P)} \in \mathbb{R}^{25N_P}$, so that the model is:

$$\bar{\mathbf{y}} = \tanh\left(\mathbf{y}^{(1\dots N_P)}\mathbf{W}_1 + \mathbf{b}_1\right)\mathbf{W}_2 + \mathbf{b}_2, \tag{5}$$

where the hidden layer has size 259 and the output layer is composed by 25 units.

In both types of ensemble, as before, the output vector contains a score for each of the classes, providing a way to rank them from the most to the least likely. The most likely class $\tilde{c}$ is thus computed as $\tilde{c} = \arg\max_i \bar{\mathbf{y}}_i$.

5 Training

The training algorithm differs based on the kind of model taken under consideration. We address each of them in the following paragraphs.

Models 1-6 The first six models are TreeESNs using a multilayer perceptron as readout. Given the fact that the main evaluation metric for the competition is the Macro F-score, each of the models has been trained by rebalancing the frequencies of the different target classes. In particular, the sampling probability for each input tree has been skewed so that the data extracted during training follows a uniform distribution with respect to the target class. For the readout part we use the Adam algorithm (Kingma and Ba, 2015) for the stochastic optimization of the multi-class cross entropy loss function.

Models 7-10 Models from 7 to 10 are again TreeESNs, but with a Ridge Regression readout. In this case, 25 classifiers are trained with a 1-vs-all method, one for each class, using binary targets.

Models 11-12 Models 11 and 12 are again TreeESNs with a Ridge Regression readout, but they are trained to distinguish only between the most frequent class, the second most frequent class and all the other classes aggregated together. This is done to try to improve the ensemble precision and recall for the top two classes.

Model 13 The last model is a sequential LSTM over character embeddings. Like in the first 6 models, the Adam algorithm is used to optimize the cross entropy loss function.

6 Results

The ensemble seems to bring a substantial improvement to the performance on the validation set, as highlighted in Table 2. This is possible thanks to the number and diversity of the different models, as we can see in Figure 2 where we show the Pearson correlation coefficients between the predictions of the models in the ensemble.

On the test set we scored substantially lower,

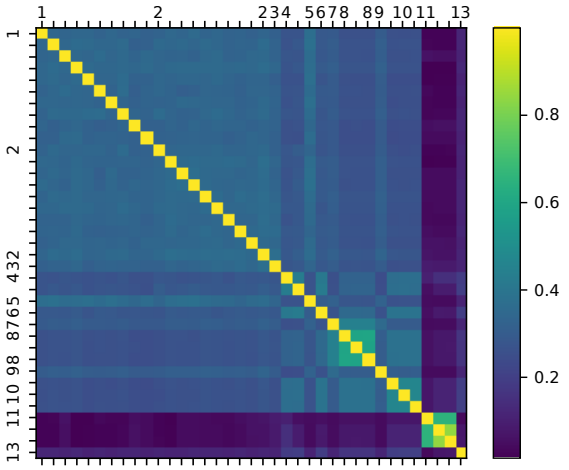

Figure 2: Plot of the correlation between the predictions of the models in the ensemble. For reasons of space, not all labels are shown on the axes.

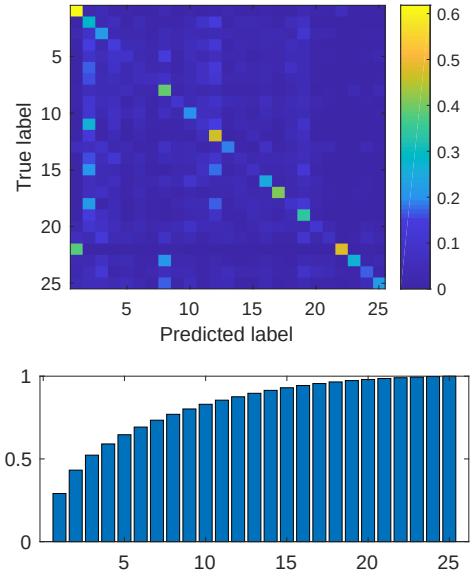

Figure 3: Confusion matrix (top) and accuracy at top-N (bottom) on the test set. Labels are ordered by frequency.

Run	Avg F1	Max F1	Ens. F1	CovE
run1	14.4	18.5	24.9	4.014
run2	14.4	18.5	26.7	3.428

Table 2: Performance obtained on the validation set for the two submitted runs. The columns are, in order, the average and maximum Macro-F1 over the models in the ensemble, and the Macro-F1 and Coverage Error of the ensemble.

Run	Macro-F1	Coverage Error
run1	**19.24**	5.4317
run2	18.80	5.1144

Table 3: Performance on the test set. These values have been obtained by retraining the models over the whole dataset (training set *and* validation set) after the final model selection phase.

with the Macro-F1 and Coverage Errors reported in Table 3. These numbers are close to those obtained by the top two models applied to the Spanish language in the "Multilingual Emoji Prediction" task of the SemEval-2018 competition (Barbieri et al., 2018), with F1 scores of 22.36 and 18.73 (Çöltekin and Rama, 2018; Coster et al., 2018). In Figure 3 we report the confusion matrix (with values normalized over the columns to address label imbalance) and the accuracy over the top-N classes.

An interesting characteristic of this approach, though, is computation time: we were able to train a TreeESN with 5000 reservoir units over 200,000 trees in just about 25 minutes, and this is without exploiting parallelism between the trees.

In ITAmoji 2018, our team ranked 3rd out of 5. Detailed results and rankings are available at `http://bit.ly/ITAmoji18`.

7 Discussion and conclusions

Different authors have highlighted the difference in performance between SVM models and (deep) neural models for emoji prediction, and more in general for text classification tasks, suggesting that simple models like SVMs are more able to capture the features which are most important for generalization: see for example the reports of the SemEval-2018 participants Çöltekin and Rama (2018) and Coster et al. (2018).

In this work, instead, we approached the problem from the novel perspective of reservoir computing applied to the grammatical tree structure of the sentences. Despite a significant performance drop on the test set[3] we showed that, paired with a rich ensemble, the method is comparable to the results obtained in the past by other participants in similar competitions using very different models.

[3]Probably due to overtraining: we observed that Macro-F1 overcame 0.40 in training.

References

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are Emojis Predictable? *arXiv preprint arXiv:1702.07285*.

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*, pages 24–33.

Christos Baziotis, Nikos Athanasiou, Georgios Paraskevopoulos, Nikolaos Ellinas, Athanasia Kolovou, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 Task 2: Predicting Emojis using RNNs with Context-aware Attention. *arXiv preprint arXiv:1804.06657*.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on Multilingual Dependency Parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, pages 149–164. Association for Computational Linguistics.

Çağrı Çöltekin and Taraka Rama. 2018. Tübingen-Oslo at SemEval-2018 Task 2: SVMs perform better than RNNs in Emoji Prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 34–38.

Joël Coster, Reinder Gerard Dalen, and Nathalie Adriënne Jacqueline Stierman. 2018. Hatching Chick at SemEval-2018 Task 2: Multilingual Emoji Prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 445–448.

Claudio Gallicchio and Alessio Micheli. 2011. Architectural and Markovian factors of echo state networks. *Neural Networks*, 24(5):440–456.

Claudio Gallicchio and Alessio Micheli. 2013. Tree Echo State Networks. *Neurocomputing*, 101:319–337.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM networks. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint conference on*, volume 4, pages 2047–2052. IEEE.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Arthur E Hoerl and Robert W Kennard. 1970. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(1):55–67.

Matthew Honnibal and Mark Johnson. 2015. An Improved Non-monotonic Transition System for Dependency Parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal, September. Association for Computational Linguistics.

Herbert Jaeger and Harald Haas. 2004. Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, 304(5667):78–80.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. *arXiv preprint arXiv:1607.01759*.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: Amethod for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Mantas Lukoševičius and Herbert Jaeger. 2009. Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, 3(3):127–149.

Joakim Nivre, Marie-Catherine De Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D Manning, Ryan T McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, et al. 2016. Universal Dependencies v1: A Multilingual Treebank Collection. In *LREC*.

A. Palmero Aprosio and G. Moretti. 2016. Italy goes to Stanford: a collection of CoreNLP modules for Italian. *ArXiv e-prints*, September.

Francesco Ronzano, Francesco Barbieri, Endang Wahyu Pamungkas, Viviana Patti, and Francesca Chiusaroli. 2018. Overview of the EVALITA 2018 Italian Emoji Prediction (ITAMoji) Task. In Tommaso Caselli, Nicole Novielli, Viviana Patti, and Paolo Rosso, editors, *Proceedings of the 6th evaluation campaign of Natural Language Processing and Speech tools for Italian (EVALITA'18)*, Turin, Italy. CEUR.org.