Sentiment Analysis using Convolutional Neural Networks with Multi-Task Training and Distant Supervision on Italian Tweets

Jan Deriu
Zurich University of Applied Sciences
Switzerland
deri@zhaw.ch

Mark Cieliebak
Zurich University of Applied Sciences
Switzerland
ciel@zhaw.ch

Abstract

English. In this paper, we propose a classifier for predicting sentiments of Italian Twitter messages. This work builds upon a deep learning approach where we leverage large amounts of weakly labelled data to train a 2-layer convolutional neural network. To train our network we apply a form of multi-task training. Our system participated in the EvalItalia-2016 competition and outperformed all other approaches on the sentiment analysis task.

In questo articolo, presentiamo un sistema per la classificazione di soggettività e polarità di tweet in lingua italiana. L'approccio descritto si basa su reti neurali. In particolare, utilizziamo un dataset di 300M di tweet per addestrare una convolutional neural network. Il sistema è stato addestrato e valutato sui dati forniti dagli organizzatori di Sentipolc, task di sentiment analysis su Twitter organizzato nell'ambito di Evalita 2016..

1 Introduction

Sentiment analysis is a fundamental problem aiming to give a machine the ability to understand the emotions and opinions expressed in a written text. This is an extremely challenging task due to the complexity and variety of human language.

The sentiment polarity classification task of EvalItalia-2016 [1] (sentipolc) consists of three subtasks which cover different aspects of sentiment detection: $T1$: Subjectivity detection: is the tweet subjective or objective? $T2$: Polarity detection: is the sentiment of the tweet neutral, positive, negative or mixed? $T3$: Irony detection: is the tweet ironic?

The classic approaches to sentiment analysis usually consist of manual feature engineering and applying some sort of classifier on these features (Liu, 2015). Deep neural networks have shown great promises at capturing salient features for these complex tasks (Mikolov et al., 2013b; Severyn and Moschitti, 2015a). Particularly successful for sentiment classification were Convolutional Neural Networks (CNN) (Kim, 2014; Kalchbrenner et al., 2014; Severyn and Moschitti, 2015b; Johnson and Zhang, 2015), on which our work builds upon. These networks typically have a large number of parameters and are especially effective when trained on large amounts of data.

In this work, we use a distant supervision approach to leverage large amounts of data in order to train a 2-layer CNN [2]. More specifically, we train a neural network using the following three-phase procedure: $P1$: creation of word embeddings for the initialization of the first layer based on an unsupervised corpus of 300M Italian tweets; $P2$: distant supervised phase, where the network is pre-trained on a weakly labelled dataset of 40M tweets where the network weights and word embeddings are trained to capture aspects related to sentiment; and $P3$: supervised phase, where the network is trained on the provided supervised training data consisting of 7410 manually labelled tweets.

As the three tasks of EvalItalia-2016 are closely related we apply a form of multitask training as proposed by (Collobert et al., 2011), i.e. we train one CNN for all the tasks simultaneously. This has two advantages: i) we need to train only one model instead of three models, and ii) the CNN has access to more information which benefits the score. The experiments indicate that the multi-task CNN performs better than the single-

[1] http://www.di.unito.it/~tutreeb/sentipolc-evalita16/index.html

[2] We here refer to a layer as one convolutional and one pooling layer.

task CNN. After a small bugfix regarding the data-preprocessing our system outperforms all the other systems in the sentiment polarity task.

2 Convolutional Neural Networks

We train a 2-layer CNN using 9-fold cross-validation and combine the outputs of the 9 resulting classifiers to increase robustness. The 9 classifiers differ in the data used for the supervised phase since cross-validation creates 9 different training and validation sets.

The architecture of the CNN is shown in Figure 1 and described in detail below.

Sentence model. Each word in the input data is associated to a vector representation, which consists in a d-dimensional vector. A sentence (or tweet) is represented by the concatenation of the representations of its n constituent words. This yields a matrix $\mathbf{S} \in \mathbb{R}^{\mathrm{d}\times\mathrm{n}}$, which is used as input to the convolutional neural network.
The first layer of the network consists of a lookup table where the word embeddings are represented as a matrix $\mathbf{X} \in \mathbb{R}^{\mathrm{d}\times|\mathrm{V}|}$, where V is the vocabulary. Thus the i-th column of X represents the i-th word in the vocabulary V.

Convolutional layer. In this layer, a set of m filters is applied to a sliding window of length h over each sentence. Let $\mathbf{S}_{[i:i+h]}$ denote the concatenation of word vectors $\mathbf{s}_i$ to $\mathbf{s}_{i+h}$. A feature c_i is generated for a given filter $\mathbf{F}$ by:

$$c_i := \sum_{k,j} (\mathbf{S}_{[i:i+h]})_{k,j} \cdot \mathbf{F}_{k,j} \qquad (1)$$

A concatenation of all vectors in a sentence produces a feature vector $\mathbf{c} \in \mathbb{R}^{\mathrm{n-h+1}}$. The vectors $\mathbf{c}$ are then aggregated over all m filters into a feature map matrix $\mathbf{C} \in \mathbb{R}^{\mathrm{m}\times(\mathrm{n-h+1})}$. The filters are learned during the training phase of the neural network using a procedure detailed in the next section.

Max pooling. The output of the convolutional layer is passed through a non-linear activation function, before entering a pooling layer. The latter aggregates vector elements by taking the maximum over a fixed set of non-overlapping intervals. The resulting pooled feature map matrix has the form: $\mathbf{C_{pooled}} \in \mathbb{R}^{m\times\frac{n-h+1}{s}}$, where s is the length of each interval. In the case of overlapping intervals with a stride value s_t, the pooled feature map matrix has the form $\mathbf{C_{pooled}} \in \mathbb{R}^{m\times\frac{n-h+1-s}{s_t}}$. Depending on whether the borders are included or not, the result of the fraction is rounded up or down respectively.

Hidden layer. A fully connected hidden layer computes the transformation $\alpha(\mathbf{W}*\mathbf{x}+\mathbf{b})$, where $\mathbf{W} \in \mathbb{R}^{\mathrm{m}\times\mathrm{m}}$ is the weight matrix, $\mathbf{b} \in \mathbb{R}^{\mathrm{m}}$ the bias, and α the rectified linear ($relu$) activation function (Nair and Hinton, 2010). The output vector of this layer, $\mathbf{x} \in \mathbb{R}^{\mathrm{m}}$, corresponds to the sentence embeddings for each tweet.

Softmax. Finally, the outputs of the hidden layer $\mathbf{x} \in \mathbb{R}^{\mathrm{m}}$ are fully connected to a soft-max regression layer, which returns the class $\hat{y} \in [1, K]$ with largest probability,

$$\hat{y} := \arg\max_j \frac{e^{\mathbf{x}^\intercal \mathbf{w}_j + a_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\intercal \mathbf{w}_k + a_j}}, \qquad (2)$$

where $\mathbf{w}_j$ denotes the weights vector of class j and a_j the bias of class j.

Network parameters. Training the neural network consists in learning the set of parameters $\mathbf{\Theta} = \{\mathbf{X}, \mathbf{F}_1, \mathbf{b}_1, \mathbf{F}_2, \mathbf{b}_2, \mathbf{W}, \mathbf{a}\}$, where $\mathbf{X}$ is the embedding matrix, with each row containing the d-dimensional embedding vector for a specific word; $\mathbf{F}_i, \mathbf{b}_i (i = \{1, 2\})$ the filter weights and biases of the first and second convolutional layers; $\mathbf{W}$ the concatenation of the weights $\mathbf{w}_j$ for every output class in the soft-max layer; and $\mathbf{a}$ the bias of the soft-max layer.

Hyperparameters For both convolutional layers we set the length of the sliding window h to 5, the size of the pooling interval s is set to 3 in both layers, where we use a striding of 2 in the first layer, and the number of filters m is set to 200 in both convolutional layers.

Dropout Dropout is an alternative technique used to reduce overfitting (Srivastava et al., 2014). In each training stage individual nodes are dropped with probability p, the reduced neural net is updated and then the dropped nodes are reinserted. We apply Dropout to the hidden layer and to the input layer using $p = 0.2$ in both cases.

Optimization The network parameters are learned using *AdaDelta* (Zeiler, 2012), which adapts the learning rate for each dimension using only first order information. We used the default hyper-parameters.

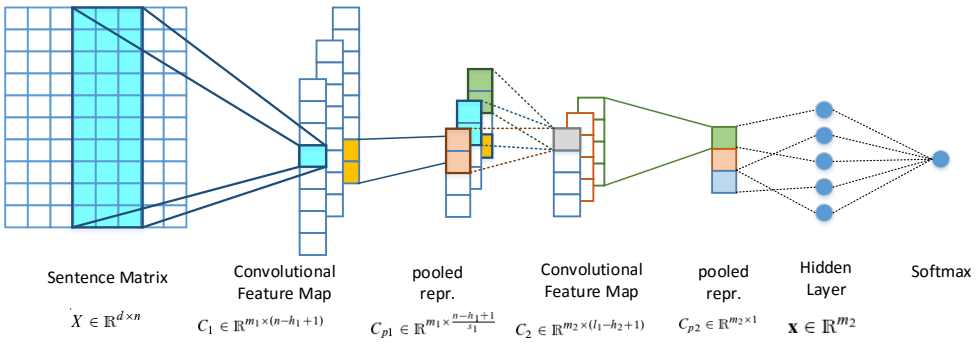

Figure 1: The architecture of the CNN used in our approach.

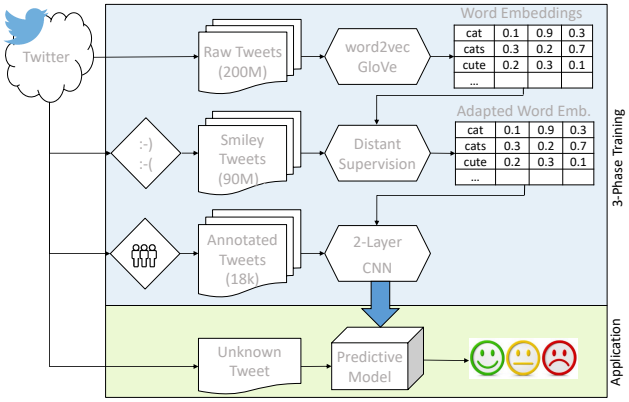

Figure 2: The overall architecture of our 3-phase approach.

3 Training

We train the parameters of the CNN using the three-phase procedure as described in the introduction. Figure 2 depicts the general flow of this procedure.

3.1 Three-Phase Training

Preprocessing We apply standard preprocessing procedures of normalizing URLs, hashtags and usernames, and lowercasing the tweets. The tweets are converted into a list of indices where each index corresponds to the word position in the vocabulary V. This representation is used as input for the lookup table to assemble the sentence matrix S.

Word Embeddings We create the word embeddings in phase $P1$ using word2vec (Mikolov et al., 2013a) and train a skip-gram model on a corpus of 300M unlabelled Italian tweets. The window size for the skip-gram model is 5, the threshold for the minimal word frequency is set to 20 and the number of dimensions is $d = 52$ [3]. The resulting vocabulary contains $890K$ unique words. The word embeddings account for the majority of network parameters ($42.2M$ out of $46.6M$ parameters) and are updated during the next two phases to introduce sentiment specific information into the word embeddings and create a good initialization for the CNN.

Distant Supervised Phase We pre-train the CNN for 1 epoch on an weakly labelled dataset of 40M Italian tweets where each tweet contains an emoticon. The label is inferred by the emoticons inside the tweet, where we ignore tweets with opposite emoticons. This results in 30M positive tweets and 10M negative tweets. Thus, the classifier is trained on a binary classification task.

Supervised Phase During the supervised phase we train the pre-trained CNN with the provided annotated data. The CNN is trained jointly on all tasks of EvalItalia. There are four different binary labels as well as some restrictions which result in 9 possible joint labels (for more details, see Section 3.2). The multi-task classifier is trained to predict the most likely joint-label.
We apply 9-fold cross-validation on the dataset generating 9 equally sized buckets. In each round we train the CNN using early stopping on the held-out set, i.e. we train it as long as the score improves on the held-out set. For the multi-task training we monitor the scores for all 4 subtasks simultaneously and store the best model for each subtask. The training stops if there is no improvement of any of the 4 monitored scores.

Meta Classifier We train the CNN using 9-fold cross-validation, which results in 9 different models. Each model outputs nine real-value numbers

[3] According to the gensim implementation of word2vec using d divisible by 4 speeds up the process.

$\hat{y}$ corresponding to the probabilities for each of the nine classes. To increase the robustness of the system we train a random forest which takes the outputs of the 9 models as its input. The hyper-parameters were found via grid-search to obtain the best overall performance over a development set: Number of trees (100), maximum depth of the forest (3) and the number of features used per random selection (5).

3.2 Data

The supervised training and test data is provided by the EvalItalia-2016 competition. Each tweet contains four labels: $L1$: is the tweet subjective or objective? $L2$: is the tweet positive? $L3$: is the tweet negative? $L4$: is the tweet ironic? Furthermore an objective tweet implies that it is neither positive nor negative as well as not ironic. There are 9 possible combination of labels.

To jointly train the CNN for all three tasks $T1, T2$ and $T3$ we join the labels of each tweet into a single label. In contrast, the single task training trains a single model for each of the four labels separately.

Table 1 shows an overview of the data available.

Table 1: Overview of datasets provided in EvalItalia-2016.

Label	Training Set	Test Set
Total	7410	2000
Subjective	5098	1305
Overall Positive	2051	352
Overall Negative	2983	770
Irony	868	235

3.3 Experiments & Results

We compare the performance of the multi-task CNN with the performance of the single-task CNNs. All the experiments start at the third-phase, i.e. the supervised phase. Since there was no predefined split in training and development set, we generated a development set by sampling 10% uniformly at random from the provided training set. The development set is needed when assessing the generalization power of the CNNs and the meta-classifier. For each task we compute the averaged F1-score (Barbieri et al., 2016). We present the results achieved on the dev-set and the test-set used for the competition. We refer to the set which was held out during a cross validation iteration as fold-set.

In Table 2 we show the average results obtained by the 9 CNNs after the cross validation. The scores show that the CNN is tuned too much towards the held-out folds since the scores of the held-out folds are significantly higher. For example, the average score of the positivity task is 0.733 on the held-out sets but only 0.6694 on the dev-set and 0.6601 on the test-set. Similar differences in sores can be observed for the other tasks as well. To mitigate this problem we apply a random forest on the outputs of the 9 classifiers obtained by cross-validation. The results are shown in Table 3. The meta-classifier outperforms the average scores obtained by the CNNs by up to 2 points on the dev-set. The scores on the test-set show a slightly lower increase in score. Especially the single-task classifier did not benefit from the meta-classifier as the scores on the test set decreased in some cases.

The results show that the multi-task classifier outperforms the single-task classifier in most cases. There is some variation in the magnitude of the difference: the multi-task classifier outperforms the single-task classifier by 0.06 points in the negativity task in the test-set but only by 0.005 points in the subjectivity task.

Set	Task	Subjective	Positive	Negative	Irony
Fold-Set	*Single Task*	0.723	**0.738**	0.721	0.646
	Multi Task	**0.729**	0.733	**0.737**	**0.657**
Dev-Set	*Single Task*	0.696	0.650	0.685	0.563
	Multi Task	**0.710**	**0.669**	**0.699**	**0.595**
Test-Set	*Single Task*	**0.705**	0.652	0.696	0.526
	Multi Task	0.681	**0.660**	**0.700**	**0.540**

Table 2: Average F1-score obtained after applying cross validation.

Set	Task	Subjective	Positive	Negative	Irony
Dev-Set	*Single Task*	0.702	**0.693**	0.695	0.573
	Multi Task	**0.714**	0.686	**0.717**	**0.604**
Test-Set	*Single Task*	0.712	0.650	0.643	0.501
	Multi Task	**0.714**	**0.653**	**0.713**	**0.536**

Table 3: F1-Score obtained by the meta classifier.

4 Conclusion

In this work we presented a deep-learning approach for sentiment analysis. We described the three-phase training approach to guarantee a high quality initialization of the CNN and showed the effects of using a multi-task training approach. To increase the robustness of our system we applied a meta-classifier on top of the CNN. The system was evaluated in the EvalItalia-2016 competition where it achieved 1^{st} place in the polarity task and high positions on the other two subtasks.

References

Francesco Barbieri, Valerio Basile, Danilo Croce, Malvina Nissim, Nicole Novielli, and Viviana Patti. 2016. Overview of the EVALITA 2016 SENTiment POLarity Classification Task. In Pierpaolo Basile, Anna Corazza, Franco Cutugno, Simonetta Montemagni, Malvina Nissim, Viviana Patti, Giovanni Semeraro, and Rachele Sprugnoli, editors, *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)*. Associazione Italiana di Linguistica Computazionale (AILC).

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (Almost) from Scratch. *JMLR*, 12:2493–2537.

Rie Johnson and Tong Zhang. 2015. Semi-supervised Convolutional Neural Networks for Text Categorization via Region Embedding. In *NIPS 2015 - Advances in Neural Information Processing Systems 28*, pages 919–927.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *ACL - Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics*, pages 655–665, Baltimore, Maryland, USA, April.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *EMNLP 2014 - Empirical Methods in Natural Language Processing*, pages 1746–1751, August.

Bing Liu. 2015. *Sentiment Analysis*.

Tomas Mikolov, Quoc V Le, and Ilya Sutskever. 2013a. Exploiting Similarities among Languages for Machine Translation. *arXiv*, September.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

Aliaksei Severyn and Alessandro Moschitti. 2015a. Twitter Sentiment Analysis with Deep Convolutional Neural Networks. In *38th International ACM SIGIR Conference*, pages 959–962, New York, USA, August. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2015b. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *SemEval 2015 - Proceedings of the 9th International Workshop on Semantic Evaluation*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv*, page 6.