UniPI at SemEval-2016 Task 4: Convolutional Neural Networks for Sentiment Classification

Giuseppe Attardi, Daniele Sartiano
Dipartimento di Informatica
Università di Pisa
Italy
{attardi,sartiano}@di.unipi.it

Abstract

The paper describes our submission to the task on Sentiment Analysis on Twitter at SemEval 2016. The approach is based on a Deep Learning architecture using convolutional neural networks. The approach used only word embeddings as features. The submission used embeddings created from a corpus of news articles. We report on further experiments using embeddings built for a corpus of tweets as well as sentiment specific word embeddings obtained by distant supervision.

1 Introduction

Up until recently, the typical approaches to sentiment analysis of tweets were based on classifiers trained using several hand-crafted features, in particular lexicons of words with an assigned polarity value. At the SemEval 2015 task 10 on Sentiment Analysis of Twitter (Rosenthal et al., 2015), most systems relied on features derived from sentiment lexicons. Other important features included bag-of-words features, hash-tags, handling of negation, word shape and punctuation features, elongated words, etc. Moreover, tweet pre-processing and normalization were an important part of the processing pipeline.

Quite significantly, the top scoring system in subtask A: Phrase-Level Polarity (Moschitti and Severyn, 2015) was instead based on the use of a convolutional neural network, which used word embeddings as its only features. Word embeddings were created by unsupervised learning from a collection of 50 million tweets, using the SkipGram model by Mikolov et al, (2013). The tweets used for training were collected by querying the Twitter API on the presence of a set of emoticons representing positive or negative sentiment. The winning team achieved an F1 of 84.79 on the Twitter2015 test set. The team participated with a similar approach also to subtask B: Message-Level Polarity, achieving the second best score with an F1 of 64.59. The fourth F1 score of 64.17 was achieved also by a system exploiting word embeddings by INESC-ID. The top scoring system instead consisted in an ensemble combining four Twitter sentiment classification approaches that participated in previous editions, with an F1 of 64.84.

We decided to explore a similar approach for tackling SemEval 2016 task 4 on Sentiment Analysis in Twitter (Nakov et al., 2016).

2 Approach

The classifier is based on a Deep Learning architecture consisting of the following layers:

1. a lookup layer for word embeddings
2. a convolutional layer with relu activation
3. a maxpooling layer
4. a dropout layer
5. a linear layer with tanh activation
6. a softmax layer

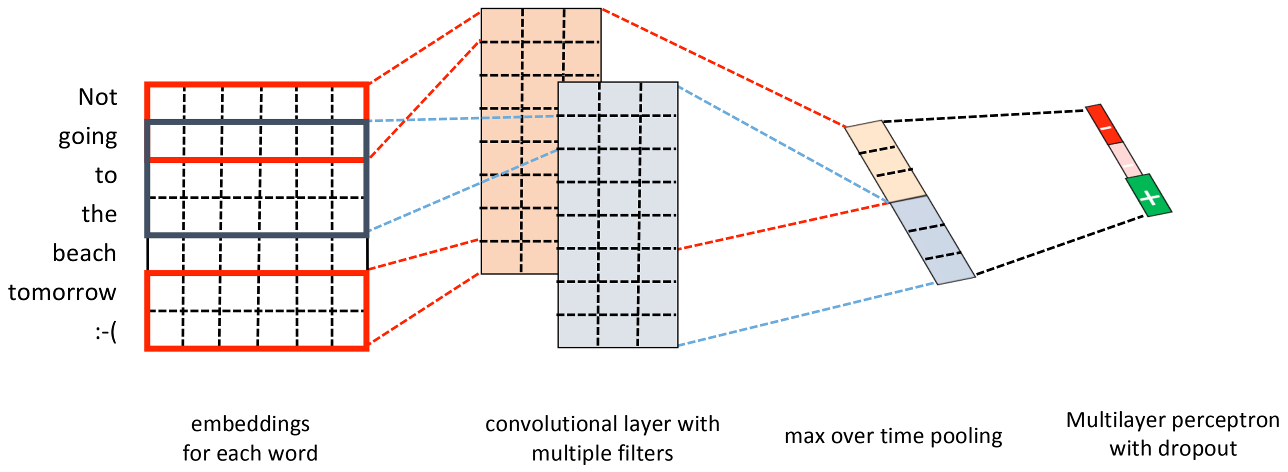
Not
going
to
the
beach
tomorrow
:-(
embeddings for each word
convolutional layer with multiple filters
max over time pooling
Multilayer perceptron with dropout

Figure 1. Architecture of the Deep Learning classifier.

The architecture is illustrated in Figure 1. The first layer looks up the embedding for words, i.e. vectors in $\mathbb{R}^d$. In our experiments we set $d = 300$. The convolutional layer consists of multiple convolutional filters of sliding windows of various sizes. These are then combined through a max-pooling layer and then passed through a multilayer perceptron with dropout.

A sentence s is represented by a sentence matrix S, where each row S_i corresponds to the embedding for the i-th word in a sentence (in the figure the matrix S is transposed).

A convolution operation involves a filter $F \in \mathbb{R}^{h \times d}$, which is applied to sliding windows of size h words to produce a feature map. A feature c_i is obtained by applying the filter to a window of words $S_{i:i+h-1}$ as:

$$c_i = f(F \otimes S_{i:i+h-1} + b)$$

where $\otimes$ is the Frobenius matrix product, $b \in \mathbb{R}$ is a bias term and f is a non-linear activation function such as the hyperbolic tangent. The filter is applied to each possible window of words in the sentence to produce a feature map $c \in \mathbb{R}^{n-h+1}$, where n is the length of the sentence (padded as necessary).

Several filters F_k are used each with its own filter width h_k.

Max-pooling is applied to the feature maps produced by the filters obtaining a single vector that forms the input to an MLP with a dropout layer. Dropout is used to prevents co-adaptation of hidden units by setting to zero with probability p a portion of the hidden units during forward propagation.

We preprocessed the tweets using the Tanl pipeline (Attardi, 2010) for performing Sentence Splitting and applying a tweet tokenizer capable of recognizing mentions, hashtags and emoticons. All mentions were replaced by @user, all URL were replaced by http://URL. All sentences are padded to the same length in order to enable processing mini-batches in parallel using CUDA matrix operations.

For training we used the training and development set from SemEval 2013 task 10 and from SemEval 2016.

The code for the DL classifier is written in Theano and is available on GitHub[1]. The implementation is derived from the one developed by Yoon Kim for the paper (Kim, 2014), and provides the ability to configure parameters and settings of the network, as well as to control the use of cross-validation.

The network is implemented by a class `ConvNet` that extends the `MLPDropout` class from `https://github.com/mdenil/dropout`.

Training is done by mean of ASGD with updated performed with Adadelta (Zeiler, 2012).

```
class ConvNet(MLPDropout):
  def __init__(self, U, height,
    width, filter_ws,
    conv_non_linear,
    feature_maps, output_units,
    batch_size, dropout_rates,
    activations)
```

[1] https://github.com/attardi/CNN_sentence

	2013		2014			2015	2016 Tweet	
	Tweet	SMS	Tweet	Sarcasm	Live-Journal	Tweet	Avg F1	Acc
SwissCheese	0.700_5	0.637_2	0.716_2	**0.566_1**	0.695_7	**0.671_1**	**0.633_1**	**0.646_1**
UniPI	0.592_{18}	0.585_{11}	0.627_{18}	0.381_{25}	0.654_{12}	0.586_{19}	0.571_{18}	0.639_3
UniPI SWE	0.642	0.606	0.684	0.481	0.668	0.635	0.592	0.652

Table 1. Official results of our submission compared to the top one, and an unofficial run.

where:

U: embedding matrix
height: sentence length
width: word vector dimension
filter_ws: filter window sizes
conv_activation: activation function of the convolutional layer (default relu)
feature_maps: number of feature maps (per filter window)
output_units: number of output variables
batch_size: size of the mini-batches
droput_rates: probability for the dropout laye
activations: the activation functions for each layer in the MLP

3 Experiments

The settings for the experiment were the following: word embeddings of 300 dimensions from a Google News corpus[2], filters of size 7,7,7 each with a 100 feature maps, dropout rate 0.5, MLP hidden units 100, batch size 50, adadelta decay 0.95, convolutional layer activation relu, training epochs 25.

For choosing the setting of our single submission, we took into account the suggestion from the experiments carried out by Zhang and Wallace. (2015).

The experiments were run on a linux server with an nVIDIA Tesla K40 accelerated GPU.

4 Results

The official submission achieved result presented in Table 1, compared to the top scoring system, as well as with an unofficial run using SWE (Sentiment Specific Word Embeddings) as described later.

A remarkable aspect of the submission is the fact that the accuracy on the 2016 Tweet jumps from the 18th position to third place.

[2] https://code.google.com/archive/p/word2vec/

In order to analyze this phenomenon, we present the breakdown of score among the three categories.

This is the breakdown of the scores of our official submission:

UniPI	Prec.	Rec.	F1
positive	69.50	63.97	66.62
negative	48.77	55.73	52.02
neutral	67.68	68.11	67.68
Avg F1			59.32
Accuracy			64.62

Table 2. Detailed scores of UniPI official submission.

Details of the evaluation of the top scoring system:

SwissCheese	Prec.	Rec.	F1
positive	67.48	74.14	70.66
negative	53.26	67.86	59.68
neutral	71.47	59.51	64.94
Avg F1			**65.17**
Accuracy			64.62

Table 3. Detailed scores of top submission for Task 4, subtask A.

After submission we performed further experiments, summarized in the following tables, which report the micro average F1 over the whole test set.

The experiments used embeddings from a corpus of 35 million tweets, and the whole training set, including the 10% that was held out for the 10-fold cross validation in the submission. A run with the same hyper-parameters obtained a slight improvement:

UniPI 2	Prec.	Rec.	F1
positive	76.86	56.87	65.37
negative	47.70	63.07	54.32
neutral	66.52	72.45	69.36
Avg F1			59.85
Accuracy			65.30

Table 4. Unofficial run using the full training set and embeddings from a corpus of 35 million tweets.

Using instead a set of filters of sizes 3,5,7,7, improved further, achieving an unofficial best accuracy score:

UniPI 3	Prec.	Rec.	F1
positive	70.88	65.35	68.00
negative	50.29	58.93	54.27
neutral	68.02	68.12	68.07
Avg F1			61.14
Accuracy			**65.64**

Table 5. Unofficial run using embeddings from tweets and filters of sizes 3,5,7,7.

The good F1 score on the neutral category seems to indicate that there is a certain degree of confusion between the two other categories.

We tested also filters of sizes 3,4,5,6,7,7,7 and 10 filters of size 7, with no overall improvements.

A more interesting experiment was the use of sentiment specific word embeddings (SWE), which encode sentiment information in the continuous representation of words. We incorporated sentiment polarity from the SemEval training corpus into the embeddings built from the corpus of 35 million of tweets, by using the technique by Tang et al. (2014), and implemented in DeepNL[3] (Attardi, 2015). A neural network with a suitable loss function provides the supervision for transferring the sentiment polarity of text (e.g. sentences or tweets) to the embeddings from generic tweets.

Training the same convolutional network with filters of size 3,5,7,7 using these sentiment specific word embeddings, produced an overall improvement that lead to an Avg F1 on Tweets 2016 of 0.595, and in detail:

UniPI SWE	Prec.	Rec.	F1
positive	68.87	68.38	68.62
negative	49.49	60.02	54.25
neutral	69.07	64.50	66.70
Avg F1			61.44
Accuracy			65.18

Table 6. Run using SWE and filters of size 3,5,7,7.

In order to compare the approach by Moschitti and Severyn (2015), which is quite similar to ours, we tested the code that Severyn kindly provided us. The experiment was run in a similar configuration, using filters of size 3,5,7,7 and also using the sentiment specific embeddings. Contrary to (Moschitti and Severyn, 2015), which used a feature map of a larger size (300), in our experiments we could not see particular benefits in increasing the feature map.

[3] https://github.com/attardi/deepnl

Moschitti and Severyn had used their own embeddings, created by distant supervision on a corpus of 50 million tweets that they collected, hence the comparison can only be considered indicative.

MS-2015	Prec.	Rec.	F1
Positive	68.03	61.73	64.73
negative	49.29	42.06	45.39
neutral	57.17	66.13	61.33
Avg F1			55.06
Accuracy			59.85

Table 7. Test performed using the code by Moschitti&Severyn for SemEval 2015.

5 Conclusions

The submission confirmed the effectiveness of convolutional neural networks in the task of tweet sentiment classification. Our systems achieved good precision scores and the third best accuracy score, achieving an overall above average official score.

After submission experiments showed a good ability in separating neutral tweets from the others, with an F1 of 70 on neutral tweets, and a good precision on positive tweets, while the recall on negative tweets was low. The use of sentiment specific word embeddings seems a promising approach for overcoming this problem.

Both the code for the classifier and for computing the sentiment specific embeddings are available on GitHub.

Acknowledgments

We grateful acknowledge the support of the University of Pisa through project PRA.
We gratefully acknowledge the support by NVIDIA Corporation with a donation of the Tesla K40 GPU used in the experiments.

References

Giuseppe Attardi, Stefano Dei Rossi, Maria Simi. 2010, The Tanl pipeline. Proc. of LREC Workshop on WSPP, Malta, 2010.

Giuseppe Attardi. 2015. DeepNL: a Deep Learning NLP pipeline. Workshop on Vector Space Modeling for NLP. Proc. of NAACL HLT 2015, Denver, Colorado (June 5, 2015).

Ramón Astudillo, Silvio Amir, Wang Ling, Bruno Martins, Mario J. Silva and Isabel Trancoso. 2015. INESC-ID: Sentiment Analysis without Hand-Coded

Features or Linguistic Resources using Embedding Subspaces. *Proceedings of the 9th International Workshop on Semantic Evaluation* (SemEval 2015). Denver, Colorado, Association for Computational Linguistics.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. Proceedings of EMNLP 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, Veselin Stoyanov. 2016. SemEval-2016 Task 3: Sentiment Analysis in Twitter. Proceedings of *of the 10th International Workshop on Semantic Evaluation* (SemEval 2016). ACL.

Aliaksei Severyn and Alessandro Moschitti. 2015. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. *Proceedings of the 9th International Workshop on Semantic Evaluation* (SemEval 2015). Denver, Colorado, Association for Computational Linguistics.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. *Proceedings of the 9th International Workshop on Semantic Evaluation* (SemEval 2015). Denver, Colorado, Association for Computational Linguistics.

D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. 2014. Learning Sentiment-Specific Word Embedding for Twitter Sentiment Classification. In ACL, pp. 1555-1565.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. CoRR, vol. abs/1212.5701, http://arxiv.org/abs/1212.5701.

Ye Zhang and Byron Wallace. 2015. A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. CoRR, vol. abs/1510.03820, http://arxiv.org/abs/1510.03820