SwissCheese at SemEval-2016 Task 4: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision

Jan Deriu*
ETH Zurich
Switzerland
jderiu@student.ethz.ch

Maurice Gonzenbach*
ETH Zurich
Switzerland
mauriceg@student.ethz.ch

Fatih Uzdilli
Zurich University of Applied Sciences
Switzerland
uzdi@zhaw.ch

Aurelien Lucchi
ETH Zurich
Switzerland
alucchi@inf.ethz.ch

Valeria De Luca
ETH Zurich
Switzerland
vdeluca@vision.ee.ethz.ch

Martin Jaggi
ETH Zurich
Switzerland
jaggi@inf.ethz.ch

Abstract

In this paper, we propose a classifier for predicting message-level sentiments of English micro-blog messages from Twitter. Our method builds upon the convolutional sentence embedding approach proposed by (Severyn and Moschitti, 2015a; Severyn and Moschitti, 2015b). We leverage large amounts of data with distant supervision to train an ensemble of 2-layer convolutional neural networks whose predictions are combined using a random forest classifier. Our approach was evaluated on the datasets of the SemEval-2016 competition (Task 4) outperforming all other approaches for the *Message Polarity Classification* task.

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 of human language, which makes use of rhetorical devices such as sarcasm or irony. 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, 2015a; 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 [1], extending the 1-layer architecture proposed by (Severyn and Moschitti, 2015a). More specifically, we train a neural network using the following three-phase procedure: i) creation of word embeddings for initialization of the first layer; ii) distant supervised phase, where the network weights and word embeddings are trained to capture aspects related to sentiment; and iii) supervised phase, where the network is trained on the provided supervised training data. We also combine the predictions of several neural networks using a random forest meta-classifier. The proposed approach was evaluated on the datasets of the SemEval-2016 competition, Task 4 (Nakov et al., 2016)[2] for which it reaches state-of-the-art results.

2 System Description

2.1 Convolutional Neural Networks

We combine the outputs of two 2-layer CNNs having similar architectures but differing in the choice of certain parameters (such as the number of convolutional filters). These two networks were also initialized using different word embeddings and used slightly different training data for the distant supervised phase. The common architecture of the two

* These authors contributed equally to this work

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

[2] http://alt.qcri.org/semeval2016/

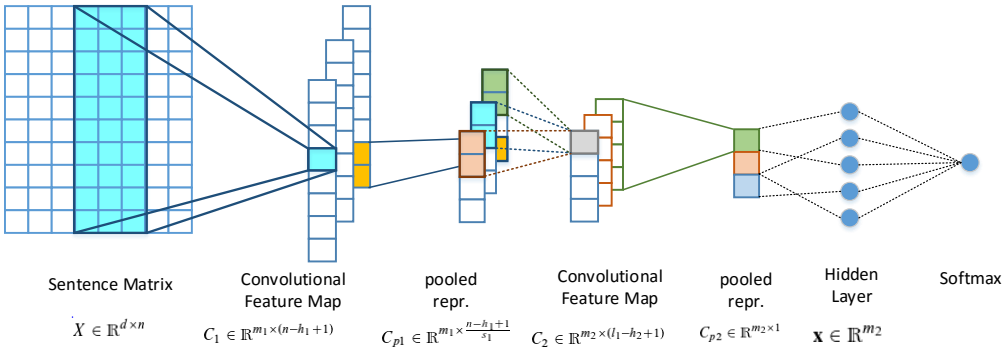

Figure 1: The architecture of the CNNs used in our approach.

CNNs is shown in Figure 1 and described in details below.

Sentence model. Each word 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 $\mathbf{X} \in \mathbb{R}^{\mathrm{d} \times \mathrm{n}}$, which is used as input to the convolutional neural network.

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

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

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 final pooling 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. i.e.,

$$\begin{aligned} \hat{y} &:= \arg\max_j P(y = j | \mathbf{x}, \mathbf{w}, \mathbf{a}) \\ &= \arg\max_j \frac{e^{\mathbf{x}^\intercal \mathbf{w}_j + a_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\intercal \mathbf{w}_k + a_j}}, \end{aligned} \tag{2}$$

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 sentence 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.

2.2 Ensemble of classifiers

We combine the results of the two 2-layer CNN described in Section 2.1 with the intent of increasing the generalizability of the final classifier. This is

achieved relying on two systems trained using different procedures as well as different word embeddings. The network parameters of the two CNNs are summarized in Table 1. The preprocessing and training phases of the two systems are described below.

2.2.1 System I

Preprocessing and word embeddings. The word embeddings are initialized using word2vec (Mikolov et al., 2013a) and then trained using an unlabelled corpus of 200M tweets. We apply a skipgram model of window size 5 and filter words that occur less than 5 times (Severyn and Moschitti, 2015b). The dimensionality of the vector representation is set to $d = 52$.

Training. During a first distant-supervised phase, we use emoticons to infer the polarity of a balanced set of 90M tweets (Read, 2005; Go et al., 2009). The resulting dataset contains 45M tweets for both the positive and negative class. The neural network is trained on these 90M tweets for one epoch, before training for 10 to 15 epochs on the labelled data provided by SemEval-2016. The word-embeddings $\mathbf{X} \in \mathbb{R}^{\mathrm{d} \times \mathrm{n}}$, are updated during both the distant and the supervised training phases, as back-propagation is applied through the entire network.

2.2.2 System II

Preprocessing and word embeddings. A corpus of 90M tweets[3] (30M contain positive emoticons, 30M negative ones and 30M contain none) is employed to create embedding vectors of $d = 50$ dimensions using *GloVe* (Pennington et al., 2014). Words which appear less than 5 times are discarded. Additionally, special flags $\in \{0, 1\}$ are assigned to some words, by appending a flag vector to their word embeddings. Four different flags can be set, marking *(i)* words that belong to hashtags, *(ii)* words that have been elongated (e.g. 'hellooo', which is mapped to the same vector as 'hello'), *(iii)* words in which all characters are capitalized, and *(iv)* punctuations that are repeated more than three times (e.g. '!!!!' and '!!!' being mapped to the same vector).

Training. In the distant supervised phase, the network is trained for one epoch on a set of 60M tweets, containing an equal amount of samples with positive and negative emoticons. Similarly to System I, this pre-trained network is further refined by supervised training for about 15 epochs on the SemEval-2016 data. We apply L_2 regularization to reduce overfitting to the cost function (negative log likelihood) by adding a penalty of the form of $\lambda \|\theta\|_2^2$, with regularization strength[4] λ, where $\theta \in \mathbf{\Theta}$ are the network parameters of each layer.

2.2.3 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 hyper-parameters $\epsilon = 1\mathrm{e}{-6}$ and $\rho = 0.95$ as suggested by (Zeiler, 2012).

2.2.4 Meta-Classifier

Each aforementioned system outputs three real values $\hat{y}$ corresponding to the three sentiment classes. In addition, it outputs the categorical value for the predicted sentiment class. The meta-classifier uses these values (sentiment class and categorical value of systems I and II) as input features. We trained a random forest using the `Weka` (Hall et al., 2009) library on the training data. We selected the number of trees (300), maximum depth of the forest (2) and the number of features used per random selection (18) as to obtain the best overall performance over the previous years' test sets.

2.3 Computing Resources

The core routines are written in `Python`, making heavy use of mathematical routines in `Theano` (Bergstra et al., 2010) that exploits GPU acceleration. For further performance improvement, we used the `CuDNN` library (Chetlur and Woolley, 2014). The framework requires approximately 10 hours for the distant supervised phase and only 20-30 minutes for the supervised phase.

Experiments were conducted on g2.2xlarge instances of *Amazon Web Services* (AWS), which offer a *GRID K520* GPU with 3072 CUDA cores and 8 GB of GDDR5 RAM.

[3] This set differs to the 90M set used in System I, but is drawn from the same larger corpus of tweets

[4] $\mathbf{X}: \lambda = 1^{-6}$, $\mathbf{F}_1: \lambda = 1^{-5}$, $\mathbf{F}_2: \lambda = 1^{-5}$, $\mathbf{W}: \lambda = 1^{-7}$.

	SYSTEM I	SYSTEM II
Number of convolutional filters	$m = 200$	$m = 300$
Filter window size h	$h_1 = 6, h_2 = 3$	$h_1 = 6, h_2 = 4$
Size of first max-pooling interval	width = 6, striding = 2	width = 3, striding = 3
Activation function α	$relu$	$relu$

Table 1: Summary of the parameters used in System I and II

3 Data

The training and development datasets used in our experiments were provided by the SemEval-2016 competition. A fraction of the tweets (10-15%) from the period 2013-2015 were no longer available on Twitter, which made the results of this year competition not directly comparable to the ones of previous years. For testing, in addition to last year's data (tweets, SMS, LiveJournal), new tweets were accessible. An overview of the data available for download is given in Table 2.

Table 2: Overview of datasets and number of tweets (or sentences) provided in SemEval-2016. The data was divided into training, development and testing sets.

Dataset	Total	Posit.	Negat.	Neutr.
Train 2013 (Tweets)	8224	3058	1210	3956
Dev 2013 (Tweets)	1417	494	286	637
Train 2016 (Tweets)	5355	2749	762	1844
Dev 2016 (Tweets)	1269	568	214	487
DevTest 2016 (Tweets)	1779	883	276	620
Test: *Twitter2016*	20632	7059	3231	10342
Test: *Twitter2015*	2390	1038	365	987
Test: *Twitter2014*	1853	982	202	669
Test: *Twitter2013*	3813	1572	601	1640
Test: *SMS2013*	2093	492	394	1207
Test: *LiveJournal2014*	1142	427	304	411
Test: *Tw2014Sarcasm*	86	33	40	13

Data preparation. Before extracting features, the tweets were preprocessed using the following procedure:

- URLs and usernames were substituted by a replacement token
- The text was lowercased
- The NLTK twitter tokenizer was employed in System I and a customized version of the *CMU ARK Twitter Part-of-Speech Tagger* (Gimpel et al., 2011) in System II.

4 Results

The F-1 score was computed by the competition organizers as evaluation measure. As a result, the presented system was ranked 1st out of 34 participants, with an F1-score of 63.30 on the Twitter-2016 test set. See (Nakov et al., 2016) for further details.

Table 4 summarizes the results of individual subsystems, as well as the final system on each test set. For each test set the best score is marked in bold face. In case of the *Twitter2016* and *Twitter2015* test sets we marked the best performing subsystem in italics. For System I (S1), we observed that during the supervised phase, the F-1 scores measured on the different test sets presented large deviations. Hence, to improve robustness, we considered six different models of S1 (S1a, ..., S1f), varying the number of epochs between 12 and 25 during the supervised phase, stopping determined by the validation score on different sets. For System II (S2), the number of epochs, equal to 12, is determined by the one achieving the highest score on the *DevTest2016* set.

The final system (FS) using the meta-classifier trained on the outputs of systems S1a-f and S2, achieves the highest accuracy on the 2016 test set with the competition-winning F1-score of 63.30%. These results improve the score of the best performing subsystem (S1b) by 0.57 points. For the 2015 test set, FS shows an improvement of 0.35 points with respect to the score of the best subsystem (S1f).

	S1a	S1b	S1c	S1d	S1e	S1f	S2	FS
Twitter2016	60.47	62.73	61.89	60.58	57.19	62.20	62.36	**63.30**
Twitter2015	64.26	65.80	64.80	64.20	61.02	66.70	66.63	**67.05**
Twitter2014	73.98	74.60	**75.70**	74.15	69.12	72.00	72.45	71.55
Twitter2013	**71.52**	70.10	70.90	71.50	67.00	68.00	70.05	70.01
LiveJournal2014	73.86	70.57	72.54	**74.00**	71.32	68.00	70.86	69.51
Tw2014Sarcasm	57.84	52.04	51.50	57.84	62.00	57.30	**62.74**	56.63

Table 3: Overall results of the proposed subsystems. S1: System(s) I; S2: System II; FS: Final system, using the meta-classifier. Best (second-best) results are highlighted in bold (underlined) face.

5 Conclusion

We described a deep learning framework to predict the sentiment polarity of short phrases, such as tweets. The proposed approach is based on an ensemble of Convolution Neural Networks and relies on a significantly large amount of data for the distant-supervised phase. The final random forest classifier resulted in state-of-the-art performance, ranking 1st in the SemEval-2016 competition for the task of Message Polarity Classification.

Acknowledgments. We thank Aliaksei Severyn and Mark Cieliebak for fruitful discussions.

References

James Bergstra, Olivier Breuleux, Frederic Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. 2010. Theano: a CPU and GPU math compiler in Python. *Proceedings of the Python for Scientific Computing Conference (SciPy)*, (Scipy):1–7.

Sharan Chetlur and Cliff Woolley. 2014. cuDNN: Efficient Primitives for Deep Learning. *arXiv preprint*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-Speech Tagging for Twitter: Annotation, Features, and Experiments. In *HLT '11 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Short-papers*, number 2, pages 42–47.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA Data Mining Software: An Update. In *SIGKDD Explorations*.

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.

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

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

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

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

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation*, SemEval '16, San Diego, California, June. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. GloVe: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL student research workshop*, pages 43–48. Association for Computational Linguistics.

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*.

Matthew D. Zeiler. 2012. ADADELTA: An Adaptive Learning Rate Method. *arXiv preprint*.