TSA-INF at SemEval-2017 Task 4: An Ensemble of Deep Learning Architectures Including Lexicon Features for Twitter Sentiment Analysis

Amit Ajit Deshmane
Infosys Limited
Pune, Maharashtra 411057, India
amitad87@gmail.com

Jasper Friedrichs
Infosys Limited
Newark, CA 94560, USA
jasper_friedrichs@infosys.com

Abstract

This paper describes the submission of team TSA-INF to SemEval-2017 Task 4 Subtask A. The submitted system is an ensemble of three varying deep learning architectures for sentiment analysis. The core of the architecture is a convolutional neural network that performs well on text classification as is. The second subsystem is a gated recurrent neural network implementation. Additionally, the third system integrates opinion lexicons directly into a convolution neural network architecture. The resulting ensemble of the three architectures achieved a top ten ranking with a macro-averaged recall of 64.3%. Additional results comparing variations of the submitted system are not conclusive enough to determine a best architecture, but serve as a benchmark for further implementations.

1 Introduction

The SemEval competitions continually offer suitable dataset and resulting benchmarks for a variety of natural language processing tasks. The SemEval-2017 Task 4 Subtask A addresses the polarity classification task of informal texts (Rosenthal et al., 2017). Tweets serve as a very accessible sample of the abundant social media content. Submitted systems must classify tweets into the categories of negative, positive and neutral opinion. Submitted results are compared over macro-averaged recall.

In recent benchmarks across this task, deep learning implementations achieved top results (Nakov et al., 2016). We seek to combine three varying deep learning approaches in an ensemble. Conventional methods seem to become obsolete since convolutional neural networks (CNN) have first shown state-of-the-art results in sentiment analysis (Kim, 2014). SemEval has since seen successful results by similar models (Severyn and Moschitti, 2015) as well as ensembles of CNNs (Deriu et al., 2016). Long term short term recurrent neural networks (LSTM) (Hochreiter and Schmidhuber, 1997) have also been applied successfully in ensemble with a CNN (Xu et al., 2016). As an alternative to LSTMs gated recurrent neural networks (GRNN) have been shown to be competitive in other domains (Chung et al., 2014). These models are well suited to model sequential data and were successfully applied for sentiment analysis of larger documents (Yang et al., 2016). The core contribution of a recent non deep learning system to win this task (Kiritchenko et al., 2014) back to back in 2013 and 2014, was the integration of opinion lexicons into a support vector machine system. Opinion lexicons have since then also been integrated into CNN architectures (Rouvier and Favre, 2016; Shin et al., 2016). In this work we combine these diverse architectures.

We use a CNN, thoroughly optimized for text classification, as the foundation of our ensemble approach. We add a lexicon integrated CNN to take advantage of lexicon features. In order to diversify the approach we also include a GRNN architecture as a sequential model. The idea is to get better and more robust results with a broader architecture. Results show that adding the latter two systems does not improve overall results, though the results for the core CNN approach were already good. Furthermore, results for individual classes do improve, making this a viable option when prioritizing specific classes or evaluation metrics. With this work we seek to contribute to the growing body of literature that presents comparable and reproducible solutions for this task.

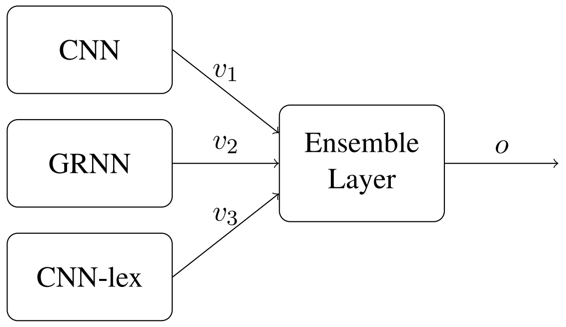

Figure 1: Ensemble component output vectors v_i are used as input to an ensemble, which determines a classification output vector o.

2 Approach

This section outlines the overall approach before detailing the subcomponents of the architecture. The purpose of the system is to classify an input tweet into an element of the opinion classes C, with $|C| = 3$. This is determined by the maximal value of an system output vector $o \in \mathbb{R}^{|C|}$. As outlined in Fig. 1 we propose an ensemble of three deep learning architectures to solve this task. A CNN and a GRNN over word embeddings as well as a CNN over lexicon embeddings. The ensemble components output vector representations $v_{1/2/3}$ can be considered an abstraction of the input tweet. These representations are the input to an ensemble system which determines the final output. We will describe preprocessing steps to create the ensemble layer input before outlining the ensemble architecture.

2.1 Preprocessing

First of all the tweet data is tokenized with NLP4J[1] as a preprocessing step for creating embeddings. In the following we refer to a tweet as a document, which is a sequence of tokens, constrained to $n = 120$. If the actual document is less in size, it is padded with zero vectors, otherwise it is truncated.

The tokens are then converted into either word embeddings of dimension d or lexicon embeddings of dimension l. We use pretrained word embeddings from Frederic Godin[2] (Godin et al., 2015), with $d = 400$. The embeddings were trained on 400 million tweets. The lexicon embeddings are polarity scores from three different lexicons, thus $l = 3$. We use Bing Liu's Opinion lexicon (Hu and Liu, 2004), the Hashtag Sentiment Lexicon and the Sentiment 140 Lexicon (Mohammad et al., 2013) to form the complete lexicon embedding.

Tensorflow[3](Abadi et al., 2016) is used as the deep learning framework for implementing the system. The next subsections describe the components of this system, followed by an outline of how their outputs are combined into an ensemble.

2.2 Convolution Neural Network

This component is based on a standard CNN architecture used for text classification (Kim, 2014). We make small changes for fine tuning to the task. The input to this component are the word embeddings described in the preceding subsection. The embeddings of dimension d are formed into a document matrix $D \in \mathbb{R}^{n \times d}$ across the n input tokens. The document matrix D is passed through k filters of filter size s. The convolution weights belong to $\mathbb{R}^{s \times d}$. The convolutions result in k convolution output vectors of dimension $\mathbb{R}^{n-s+1}$. A max pool layer converts these vectors to a vector of size $\mathbb{R}^{k}$. We then add a normalization layer (Ioffe and Szegedy, 2015) so as to merge outputs from different filter sizes. With f filter sizes, we finally arrive at vector $v_1 \in \mathbb{R}^{k \times f}$. This vector is passed to a dense layer of 256 ReLu units. The dense layer is followed by an output softmax layer. We apply dropout (Hinton et al., 2012) at the beginning of the dense layer as well as the output layer. The implementation uses following the configuration:

- Weights are initialized using Xavier weight initialization (Glorot and Bengio, 2010).
- The Learning rate is set to 0.0001 with a batch size of 100.
- The architecture uses $f = 5$ filter sizes, [1,2,3,4,5], and $k = 256$ filters per filter size over $n = 120$ word embedding vectors of dimension $d = 400$.
- The input vector to the dense layer v_1 thus has a dimension of 1280.
- At the dense layer and output layer, we use dropout with a keep probability of 0.7.
- We run 200 training iterations and select the model that performed best on development data.

[1]https://github.com/emorynlp/nlp4j
[2]http://www.fredericgodin.com/software/
[3]https://www.tensorflow.org/

2.3 Gated Recurrent Neural Network

The GRNN is based on the gated recurrent unit (GRU) (Cho et al., 2014), which uses a gating mechanism while tracking the input sequence with a latent variable. GRUs seemed to perform better compared to other RNN cells like LSTM in our experiments, which go beyond the scope of this paper. The input to the GRNN are the word embeddings described in Section 2.1. The input is read sequentially by a GRU layer. The GRU cell is designed to learn how to represent a state, based on previous inputs and the current input. The GRU layer consists of g hidden units. After the last token of the sequence is processed, the output vector $v_2 \in \mathbb{R}^g$ of this layer is collected to be merged into other architectures. The implementation is configured by:

- $g = 256$ hidden units with tanh activation,
- resulting in the 256 dimensional output vector v_2.

2.4 Lexicon Integrated Convolution Neural Network

The lexicon integrated CNN (CNN-lex) is similar to the previously described CNN architecture. The fundamental difference is that convolutions are done over lexicon embeddings, described in Section 2.1. The input to this component is a document matrix $L \in \mathbb{R}^{n \times l}$ across the l dimensional lexicon embeddings of n input tokens. The architecture uses j filters per e convolution filter sizes. The convolution layer output is passed through a max pool and normalization layer. This results in an output vector $v_3 \in \mathbb{R}^{j \times e}$ that is collected to be merged into other architectures. The implementation uses following configuration:

- The architecture uses $e = 3$ filter sizes, [3,4,5], and $j = 64$ filters per filter size over $n = 120$ lexicon embeddings of dimension $l = 3$.
- The ensemble output vector v_3 thus has a dimension of 192.

2.5 Ensembles

The previously described architectures can be combined into ensemble systems. The CNN-lex and GRNN architectures are already defined as ensemble subsystems through their output vectors v_2 and v_3. While the CNN architecture was previous introduced as a stand alone system it is naturally described as an ensemble component. The vector v_1 described in Section 2.2 is the output vector of the CNN as a subsystem. The three vectors are concatenated as inputs to the ensemble layer. The ensemble layer consists of a dense layer of 256 ReLu units followed by a softmax output layer, which results in the output vector o of dimension $|C|$. The CNN is combined into three ensembles by concatenating its input vector v_1 with v_2 (CNN, GRNN) and with v_3 (CNN, CNN-lex) as well as with both v_2 and v_3 (CNN, CNN-lex, GRNN). The latter is the submission architecture while the other two are evaluated for comparison. These ensembles are trained as a single system. Training is conducted as previously described for the CNN architecture.

Twitter Corpus	Pos	Neg	Neut	Total
2013-train	3640	1458	4586	9684
2013-dev	575	340	739	1654
2013-test	1475	559	1513	3547
2014-sarcasm	33	40	13	86
2014-test	982	202	669	1853
2015-test	1038	365	987	2390
2016-train	3094	863	2043	6000
2016-dev	843	391	765	1999
2016-devtest	994	325	681	2000
2016-test (A)	7059	3231	10342	20632
2017-test (B)	2375	3972	5937	12284

Table 1: SemEval data subsets as available to authors. Aside from development test (A) and test (B) split, all sets where combined for a train and dev split.

3 Data

The training data for this approach was constrained to data published in the context of the SemEval workshops. Table 1 lists the data available to the authors. It is important to note that the data is heavily imbalanced. Before submission the system was tested with the 2016-test set as development test data. The results described in this paper focus on the 2017-test data, which was used to rank the submissions. All other data in Table 1 was combined into one data set, shuffled and split four to one into training and development data.

4 Results

The following results compare the core CNN architecture against ensembles with the CNN as a subsystem. The ranked submission marked by $*$ in Table 4 ranked ninth out of 37 participants.

2017-test, detailed	Recall			Precision			F_1		
	pos	neg	neut	pos	neg	neut	pos	neg	neut
CNN	68.3	78.3	47.6	**54.8**	57.7	71.7	**60.8**	66.5	57.2
CNN, CNN-lex	**74.3**	**85.6**	33.1	50.3	54.7	**76.7**	60.0	**66.7**	46.3
CNN, GRNN	73.2	71.0	50.1	50.4	61.2	70.3	59.7	65.7	58.5
CNN, CNN-lex, GRNN $*$	74.1	64.0	**55.0**	50.9	**63.6**	67.6	60.3	63.8	**60.6**

Table 2: Per class results of recall precision and F1 on test data (2017-test, Table 1) for CNN and ensemble architectures, where $*$ marks the submission system

2016-test	ρ	F_1^{PN}	*Acc.*
CNN	**66.1**	**62.6**	62.8
CNN, CNN-lex	64.9	61.0	56.4
CNN, GRNN	64.2	61.9	61.3
CNN, CNN-lex, GRNN	64.0	62.4	**64.1**

Table 3: Macro-averaged recall ρ, negative positive macro-averaged F1 and accuracy on development test data (2016-test, Table 1) for CNN and ensemble architectures.

2017-test	ρ	F_1^{PN}	*Acc.*
CNN	64.7	**63.6**	61.5
CNN, CNN-lex	64.3	63.4	58.0
CNN, GRNN	**64.8**	62.7	61.3
CNN, CNN-lex, GRNN	64.3$*$	62.0	**61.6**

Table 4: Macro-averaged recall ρ, negative positive macro-averaged F1 and accuracy on test data (2017-test, Table 1) for CNN and ensemble architectures, where $*$ marks the ranked submission.

For detailed rankings we refer to the task description (Nakov et al., 2016), we only put this result in context with the described architectures. Three ensembles are used for comparison, the basic CNN combined with either the GRNN or the lexicon integrated CNN as well as both. The two result data sets are the 2016-test set as pre-submission test data and the final 2017-test data set used to benchmark the submissions.

Overall the CNN performs en par or better than the ensembles on macro-averaged recall and macro-averaged positive negative F1. For both development test data in Table 3 and test data in Table 4 we observe that the CNN outperforms the ensembles across macro-averaged F1. Though there is a substantial difference between macro-averaged recall of the CNN versus the ensembles on the development test data, macro-averaged recall on the test data is consistent across all systems.

The strongest fluctuation in averaged results is the drop in accuracy for the CNN, CNN-lex ensemble across both data sets. Detailed results in Table 2 show that this is due to a steep drop in neutral class recall, a class the data is heavily biased towards (Table 1). We note that though macro-averaged recall stays consistent on the test data (Table 4), per class results do fluctuate substantially (Table 2). These class trends were generally consistent across both 2017-test and 2016-test data, thus the later results are omitted for brevity.

5 Conclusions

In the previous sections we described experiments of adding various deep learning architecture elements to a basic CNN. Results show that the derived ensembles of approaches did not improve performance over the more relevant metrics of macro-averaged recall and F1. To give further context it is important to mention that substantially more effort went into engineering and tuning of the CNN model than of the additional architectures. Just as the submission system, the CNN architecture itself would have ranked within the top ten of this sentiment analysis task. Room for improvement was thus limited. We do note that per class results do fluctuate quite a bit across ensembles, which means these architecture can be used to prioritize class specific recall and precision.

It remains open whether the more complex architectures perform more robustly across diverse datasets. We will seek more clarity on this issue by experimenting with different data sets. Another architecture choice to pursue is to include an attention mechanism so that the ensemble system can learn which subcomponents to prioritize.

Acknowledgments

We thank assistant professor Jinho D. Choi at Emory University for guidance.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467* .

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder- decoder approaches. *arXiv preprint arXiv:1409.1259* .

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurelien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. Swisscheese at semeval-2016 task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. *Proceedings of SemEval* pages 1124–1128.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Aistats*. volume 9, pages 249–256.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab @ acl w-nut ner shared task: Named entity recognition for twitter microposts using distributed word representations. *ACL-IJCNLP* 2015:146–153.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580* .

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, pages 168–177.

Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* .

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* .

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research* 50:723–762.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the seventh international workshop on Semantic Evaluation Exercises (SemEval-2013)*. Atlanta, Georgia, USA.

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 2016)*. Association for Computational Linguistics, San Diego, California.

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 task 4: Sentiment analysis in Twitter. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

Mickael Rouvier and Benoit Favre. 2016. Sensei-lif at semeval-2016 task 4: Polarity embedding fusion for robust sentiment analysis. *Proceedings of SemEval* pages 202–208.

Aliaksei Severyn and Alessandro Moschitti. 2015. Unitn: Training deep convolutional neural network for twitter sentiment classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Association for Computational Linguistics, Denver, Colorado*. pages 464–469.

Bonggun Shin, Timothy Lee, and Jinho D Choi. 2016. Lexicon integrated cnn models with attention for sentiment analysis. *arXiv preprint arXiv:1610.06272* .

XingYi Xu, HuiZhi Liang, and Timothy Baldwin. 2016. Unimelb at semeval-2016 tasks 4a and 4b: An ensemble of neural networks and a word2vec based model for sentiment classification. *Proceedings of SemEval* pages 183–189.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*. pages 1480–1489.