Zewen at SemEval-2018 Task 1: An Ensemble Model for Affect Prediction in Tweets

Zewen Chi, Heyan Huang, Jiangui Chen, Hao Wu, Ran Wei
School of Computer Science, Beijing Institute of Technology, Beijing, China
czwin32768@gmail.com, hhy63@bit.edu.cn, chenjiangui@outlook.com,
wuhao123@bit.edu.cn, weiranbit@163.com

Abstract

This paper presents a method for Affect in Tweets, which is the task to automatically determine the intensity of emotions and intensity of sentiment of tweets. The term affect refers to emotion-related categories such as anger, fear, etc. Intensity of emotions need to be quantified into a real valued score in [0, 1]. We propose an ensemble system including four different deep learning methods which are CNN, Bidirectional LSTM (BLSTM), LSTM-CNN and a CNN-based Attention model (CA). Our system gets an average Pearson correlation score of 0.682 in the subtask EI-reg and an average Pearson correlation score of 0.784 in subtask V-reg, which ranks 19th among 48 systems in EI-reg and 17th among 38 systems in V-reg.

1 Introduction

Affect determination is a significant part of nature language processing. Especially, affect in tweets becomes a focus in recent years. Sentiment Analysis in Twitter, which is a task of SemEval, was firstly proposed in 2013 and not replaced until 2018. In SemEval 2018, the task Affect in Tweets (AIT) (Mohammad et al., 2018) was proposed and the objective is to automatically determine the intensity of emotions (E) and intensity of sentiment (aka valence V) of tweets. In this paper, we focus on two subtasks:

- EI-reg (emotion intensity regression) – Given a tweet and an emotion E, determine the intensity of E that best represents the mental state of the tweeter – a real-valued score between 0 (least E) and 1 (most E)
- V-reg (sentiment intensity regression) – Given a tweet, determine the intensity of sentiment or valence (V) that best represents the mental state of the tweeter – a real-valued score between 0 (most negative) and 1 (most positive)

Before 2016, most systems use Support Vector Machine (SVM), Naive Bayes, maximum entropy and linear regression (Nakov et al., 2013; Rosenthal et al., 2014, 2015). In SemEval 2014, deep learning methods started to appear and a team using them won the second place. Since 2015, more and more teams who were rank at the top used deep learning methods and now deep learning methods including CNN and LSTM networks become really popular (Nakov et al., 2016; Rosenthal et al., 2017).

The system described in this paper is an ensemble of four different DNN methods including CNN, Bidirectional LSTM (Bi-LSTM), LSTM-CNN and a CNN-based Attention model (CA). In these methods, words in tweets are firstly mapped to word vectors. After intensity scores are calculated by these models, we use a logistic regression and finally give the scores.

The rest of the paper is organized as follows. Section 2 describes the four various methods and the ensemble method used in our system. Section 3 and Section 4 give the implementation and training details of our system for subtask EI-reg and V-reg. Section 5 states the results and discussion in the evaluation period. Finally, Section 6 makes a conclusion on this work.

2 System Description

2.1 CNN

Inspired by Kim's work on sentence classification (Kim, 2014), the architecture of the CNN model used in our system is almost identical to his model. As it is shown in Figure 1, tweets are first fed into the embedding layer, which converts words into word vectors. Then the tweet is mapped into a matrix M of size $n \times d$. In order to reduce the number

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 313–318
New Orleans, Louisiana, June 5–6, 2018.

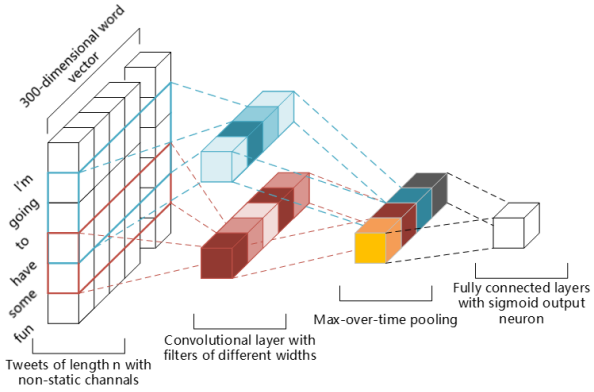

Figure 1: The architecture of our CNN model

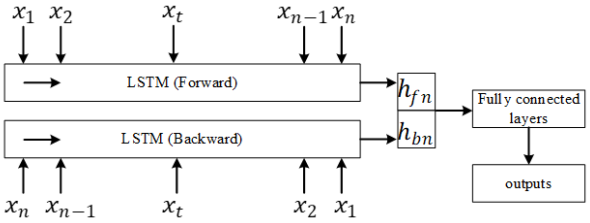

Figure 2: The architecture of our bidirectional LSTM model, where h_fn and h_bn represent the last hidden state of the forward and backward LSTM respectively.

of parameters in the neural network, we just use the single channel non-static model, which sets pre-trained word vectors in the embedding layer and can be modified in the training period. In the convolution layer, convolution operations are applied on the submatrixes of M. The convolution operation here is defined as:

$$c_k = f_k(\sum_i \sum_j \omega_{ij} x_{[i:i+h-1]} + b)$$

where $b \in \mathbb{R}$ is a bias term and f is a non-linear function such as ReLU (Jarrett et al., 2009), which is used in our approach. Filters are applied with different size of windows and in each window of size h, feature matrix $c \in \mathbb{R}^{(n-h+1)\times m}$ is produced corresponding to the filters:

$$c = [c_1, c_2, ..., c_k, ..., c_m]$$

where m is the number of filters and $c_k \in \mathbb{R}^{n-h+1}$ represents the features extracted from a word sequence. In the pooling layer, we apply a max-over-time pooling operation (Collobert et al., 2011) over feature matrix and take the maximum in each column to preserve the most important features. These maximums are concatenated and then fed into a fully-connected network (L1, L2). L2 is followed by a single sigmoid neuron node to generate the prediction of the affect on the interval $[0, 1]$.

2.2 Bidirectional LSTM

The LSTM architecture used in our system is a kind of modern Recurrent Neural Networks (RNN). Comparing to CNN, the way RNN work is more similar to that how humans read sentences. A word vector sequence x, which is converted from a tweet, will be fed to the RNN in order.

$$h_t = \sigma(W_{hx} x_t + W_{hh} h_{t-1} + b_h)$$
$$y^t = softmax(W_{yh} h_t + b_y)$$

At time t, the RNN takes the input from the current word x_t and also from the previous hidden state h_{t-1} to calculate the hidden state h_t and the output $\hat{y}_t$, which means $\hat{y}_t$ at time t is in the influence of all previous input words $x_1, ..., x_{t-1}$. However, this regular RNN suffers from the exploding and vanishing gradient problem when using the backpropagation algorithm (Hochreiter, 1998), which makes RNN hard to train. Therefore, we use the Long short-term memory (LSTM) network (Hochreiter and Schmidhuber, 1997) to overcome this problem. Each ordinary node of hidden layer in LSTMs is replaced by a memory cell and the following equations describe the LSTM:

$$g_t = \phi(W_{gx} x_t + W_{gh} h_{t-1} + b_g)$$
$$i_t = \sigma(W_{ix} x_t + W_{ih} h_{t-1} + b_i)$$
$$f_t = \sigma(W_{fx} x_t + W_{fh} h_{t-1} + b_f)$$
$$o_t = \sigma(W_{ox} x_t + W_{oh} h_{t-1} + b_o)$$
$$s_t = g_t \odot i_t + s_{t-1} \odot f_t$$
$$h_t = \phi(s_t) \odot o_t$$

The vector h_t is the value of hidden layer of LSTM at time t, g_t is the input node, i_t is the input gate, f_t is the forget gate, o_t is the output gate and s_t is the internal state where $\odot$ is pointwise multiplication. According to Zaremba and Sutskever (2014), the function ϕ used here is the tanh function.

For every point in a given sequence, Graves et al. (2005) shows that a bidirectional LSTM can preserve more sequential information about all sequential points before and after it. As the Figure 2 shows, we concatenate the hidden states of two separate LSTMs after they process the word sequence in opposite direction and get the concatenated state $h^{'} \in \mathbb{R}^{2m}$, which is fed to fully connected layers and finally give the result with a single sigmoid neuron node.

2.3 LSTM-CNN

The architecture of LSTM-CNN is a combination of previous two model. Instead of feeding the out-

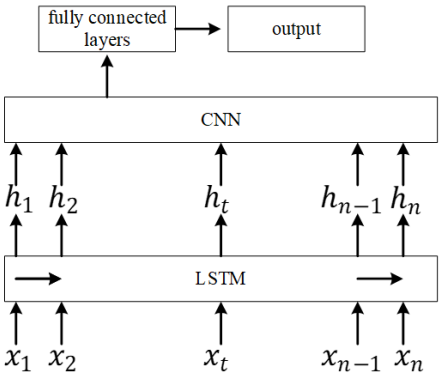

Figure 3: The architecture of our LSTM-CNN model.

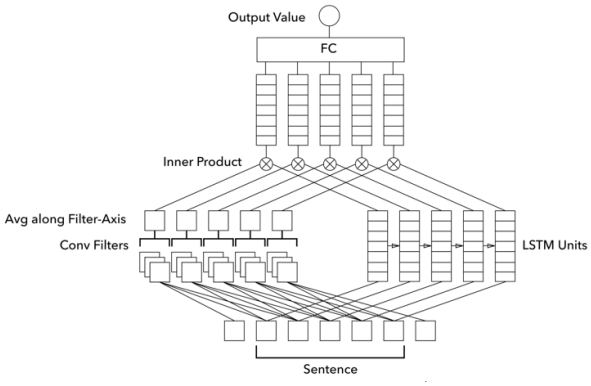

Figure 4: The architecture of CNN-based Attention Model (CA)

put of LSTM to the fully connected layers, the output of LSTM h_t at each time t are regarded as the input of CNN and Figure 3 shows the architecture.

2.4 A CNN-based Attention Model (CA)

Since attention mechanism has achieved significant improvements in many NLP tasks, including machine translation (Bahdanau et al., 2014), caption generation (Xu et al., 2015) and text summarization (Rush et al., 2015), it becomes an integral part of compelling sequence modeling and transduction models in various tasks. Motivated by Du's work on sentence classification (Du et al., 2017), the architecture of our CNN-based attention model resembles his model. We first use a CNN-based network to model the attention signal in sentences. The convolution operation here is same as that described in Section 2.1. The attention signal of original text is represented by the output of convolutional filter. In order to reduce the noise, multiple filters with same size of windows are applied. After that, we get the corresponding attention similarity:

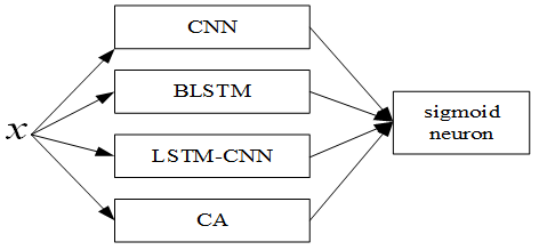

Figure 5: The architecture of the ensemble model

$[c_1, c_2, ..., c_k, ..., c_m]$. Then we obtain the attention signal of each element which represents the importance of the corresponding word by averaging the attention similarities along the filter-axis:

$$c = \frac{1}{m} \sum_{i=1}^{m} c_i$$

An RNN with LSTM units is used to encode the sentence. According to the equation in Section 2.2, the hidden state $h_t \in \mathbb{R}^d$ (where d is the dimension of the RNN) at time t is $h_t = \phi(s_t) \odot o_t$.

So far, we have obtained attention signal c_t and the corresponding hidden state vector of RNN h_t. The representation of the whole sentence can be computed by

$$s = \frac{1}{T} \sum_{t=0}^{T-1} c_t h_t$$

And then $s \in \mathbb{R}^d$ is fed into a fully-connected network (L1, L2). L2 is followed by a single sigmoid neuron node to generate the prediction of the affect on the interval [0, 1]. The architecture of this model is shown in Figure 4.

2.5 Ensemble Model

According to the results of SemEval-2017 task 4, the use of ensembles stood out clearly. Therefore, we use a mix of deep learning methods to make our system obtain better predictive performance. Inspired by the boosting algorithms, we use a logistic regression to improve the accuracy of these four methods and the architecture is shown in Figure 5. In order to make the model simple, it only takes the output of the four methods as input rather than training data.

3 Implementation

We implemented our system with PyTorch (Paszke et al., 2017) in Python 3.

Preprocessing: For making tweets string clean, we apply a preprocessing procedure on the input tweets which removes the abbreviations like 's, 've and make them lowercased.

Word Embeddings: We utilize pre-trained 300-dimensional word embeddings of Stanford's

Methods	L1	L2
CNN	300	150
BLSTM	30	Nil
LSTM-CNN	256	100
CA	150	75

Table 1: Fully connected layers hyper-parameters, the numbers represent the size of outputs of liner layers.

Methods	p	CNN	LSTM
CNN	0.2	[2, 3, 4], 256	Nil
BLSTM	0.5	Nil	300
LSTM-CNN	0.5	[3], 200	300
CA	0.5	[3], 50	150

Table 2: Network hyper-parameters for the filters of CNN and hidden size of LSTM, and p is the dropout rate. For example, [2, 3, 4], 256 means the filter height is set to 2, 3 and 4, and the number of filters is set to 256 for different sizes of filters.

GloVe (Pennington et al., 2014) trained by Common Crawl.

Model Hyper-parameters: Table 1 and Table 2 show the hyper-parameters we use in our system.

For fully connected layers, no more than two fully-connected layers are used in the four methods and all fully-connected layers are followed by ReLU. Before the outputs of pooling layers and LSTMs are fed to the fully connected layers, a dropout is applied and the details are described in Table 2.

4 Training

The dataset used in our system is provided by the AIT task and no external datasets are used in training period. For the subtask EI-reg and subtask V-reg, they are trained with the same model hyper-parameters which are listed in Table 1 and Table 2. Also, the four methods use the same word embeddings, which is a pre-trained 300-dimensional word vectors with common crawl by GloVe algorithm. For different emotions, we train the models for 10 epochs respectively. The network parameters are learned by minimizing the Mean Absolute Error (MAE) between the gold labels and predictions and the four methods used in our system are trained separately. We optimize the loss function by back-propagating algorithm via Mini-batch Gradient descent with batch size of 8 for the 4 deep learning models and full batch learning for the ensemble model, as well as the Adam optimization algorithm (Kingma and Ba, 2014) for all models with initial leaning rate of 0.001 and 0.01 for the four deep learning models and the ensemble model, respectively.

5 Result and Discussion

We compare the results of the four methods used in our system, the ensemble system, the SVM_Unigrams_Baseline provided from the AIT task and the best-performing system – SeerNet in Table 3. The metric for evaluating performance is Pearson Correlation.

Its remarkable that, comparing to the individual models, our ensemble model has an improvement of at least 2% on EI-reg subtask and 1.1% on V-reg subtask. However, it's obvious that there is a gap between our models and the best-performance system. The rough preprocessing method of our system is one of the reason for the low score. Because of some words in tweets are misspelled or in a special format like 'yaaaaay!', some of the information is lost in this process. So we added an experiment on the V-reg task to study the effect of preprocessing method. We replace the text preprocessing method with the *ekphrasis*[1] for the tokenization, word normalization, word segmentation (for splitting hashtags) and spell correction and the keep the other parameters unchanged. As it is shown in Table 4, the four methods as well as the ensemble model all get an improvement on the results. Actually, some expressions like dates, urls, hashtags and emoticons are converted into the special tokens like <date> , <url>, <hashtag> and <joy>, but these tokens are not in the dictionary of pre-trained word vectors, which means the information of these tokens is still wasted in the embedding process.

There is much room for the improvement of our method:

1. In our system, a single pre-trained word embedding is used, which lack experimental evidence. For future work, combining more kinds of word embeddings should be taken into consideration.

2. We adjust the hyper-parameters by doing evaluation on dev dataset. For future work, we can apply a more advanced strategy like Cross Validation.

[1] github.com/cbaziotis/ekphrasis

Methods	Average(EI-reg)	Anger	Fear	Joy	Sadness	V-reg
CNN	0.668	0.673	0.684	0.670	0.644	0.773
BLSTM	0.625	0.619	0.645	0.630	0.604	0.731
LSTM-CNN	0.641	0.620	0.680	0.627	0.636	0.759
CA	0.640	0.606	0.662	0.670	0.624	0.761
Ensemble	0.682	0.673	0.700	0.690	0.665	0.784
SeerNet	**0.799**	**0.827**	**0.779**	**0.792**	**0.798**	**0.873**
Baseline	0.520	0.526	0.525	0.575	0.453	0.585

Table 3: Results on Subtask EI-reg and V-reg.

Methods	Rough method	ekphrasis
CNN	0.773	**0.788**
BLSTM	0.731	**0.733**
LSTM-CNN	0.759	**0.767**
CA	0.761	**0.773**
Ensemble	0.784	**0.793**

Table 4: Results of different text preprocessing method on V-reg task when the other parameters are kept unchanged.

3. For the input features, we only use the word vectors. We are supposed to experiment with more features like lexicons.

4. In our system, we just use a simple logistic regression but achieve an impressive result on the two subtasks. There is an interesting idea that we can do more work on finding a better ensemble model.

6 Conclusion

In this paper, we propose a model on the sub-task EI-reg and V-reg of SemEval-2018 Task 1: Affect on Tweets. The submitted system is an ensemble model based on CNN, Bidirectional LSTM (BLSTM), LSTM-CNN and a CNN-based Attention model (CA). All methods are described in detail to make our work replicable.

For future work, it would be significant to make an improvement on preprocessing of tweets, doing more experiment on word embeddings and feature selection, model validation and ensemble method.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754, Vancouver, Canada. Association for Computational Linguistics.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537.

Jiachen Du, Lin Gui, Ruifeng Xu, and Yulan He. 2017. A convolutional attention model for text classification. In *National CCF Conference on Natural Language Processing and Chinese Computing*, pages 183–195. Springer.

Pranav Goel, Devang Kulshreshtha, Prayas Jain, and Kaushal Kumar Shukla. 2017. Prayas at emoint 2017: An ensemble of deep neural architectures for emotion intensity prediction in tweets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 58–65.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Hussam Hamdan. 2017. Senti17 at semeval-2017 task 4: Ten convolutional neural network voters for tweet polarity classification. *arXiv preprint arXiv:1705.02023*.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116.

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

Kevin Jarrett, Koray Kavukcuoglu, Yann LeCun, et al. 2009. What is the best multi-stage architecture for object recognition? In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 2146–2153. IEEE.

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

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

Preslav Nakov, Zornitsa Kozareva, Alan Ritter, Sara Rosenthal, and Veselin Stoyanov Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. volume 2.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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 (SemEval-2017)*, pages 502–518.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 451–463.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. Semeval-2014 task 9: Sentiment analysis in twitter. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 73–80, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Mickael Rouvier. 2017. Lia at semeval-2017 task 4: An ensemble of neural networks for sentiment classification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 760–765.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.