YNU-HPCC at SemEval-2018 Task 1: BiLSTM with Attention Based Sentiment Analysis for Affect in Tweets

You Zhang, Jin Wang and **Xuejie Zhang**
School of Information Science and Engineering
Yunnan University
Kunming, P.R. China
Contact: xjzhang@ynu.edu.cn

Abstract

This paper describes the system we built as the YNU-HPCC team in the SemEval-2018 competition. As participants of Task 1, named Affect in Tweets, we implemented the sentiment system for all five subtasks in English and Spanish. All subtasks involved predicting emotion or sentiment intensity (regression and ordinal classification) and determining emotions (multi-label classification). Our system mainly applied the bidirectional long-short term memory (BiLSTM) model with an attention mechanism. We used BiLSTM in order to extract word information from both directions. The attention mechanism was used to find the contribution of each word to improving the scores. Furthermore, based on the BiLSTM with an attention mechanism, a few deep-learning algorithms were employed for different subtasks. For regression and ordinal classification tasks, we used domain adaptation and ensemble learning methods to leverage the base model, while a single base model was used for the multi-label task. Our system achieved very competitive results on the official leaderboard.

1 Introduction

Sentiment analysis is an area of natural language processing (NLP), which aims to systematically identify and study affective state, and to quantify subjective sentiment expressed in texts. Tweets in Twitter always constitute a challenging task among NLP problems because of the colorful writing styles used.

In previous work on sentiment analysis tasks, researchers usually used a variety of hand-crafted features and sentiment lexicons to generate the solution system by combining traditional methods such as naive Bayes, support vector machines (SVMs) (Mohammad et al., 2013), and decision trees (Blake, 2007). Recently, many ensemble learning models based on these traditional methods (Giorgis et al., 2016) have attracted the interest of researcher and have shown good results. These approaches require long-term studies to gather information from massive or unstructured datasets, and often result in redundant or missing features. In contrast, the novel deep learning method (Socher et al., 2013) has immediately and shown exceptionally good results in NLP.

In this paper, we primarily present a deep learning system for the SemEval-2018 shared Task 1: Affect in Tweets. We employ the bidirectional long short-term memory with an attention mechanism (BiLSTM$_{ATT}$) as a base model. For the regression and ordinal classification tasks, we used fine-tuning methods on the base model, combined with multi-tasking and AdaBoost algorithm. We use a simple BiLSTM with an attention mechanism for the multi-label task. Our contributions are as follows:

- We propose a base model combining the BiLSTM with an attention mechanism for the sentiment analysis problem.
- Using the base model, a domain adaptation method of fine-tuning combined with multi-tasking is used for associated tasks.
- An ensemble learning method using the AdaBoost algorithm implemented on the base model is of great use for performing the task with unevenly distributed data.

The remainder of this paper is organized as follows. In Section 2, we describe an overview of our system. The details of the model are presented in Section 3. Finally, comparative results of the experiments are discussed, and a conclusion is drawn in Sections 4 and 5, respectively.

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 273–278
New Orleans, Louisiana, June 5–6, 2018. ©2018 Association for Computational Linguistics

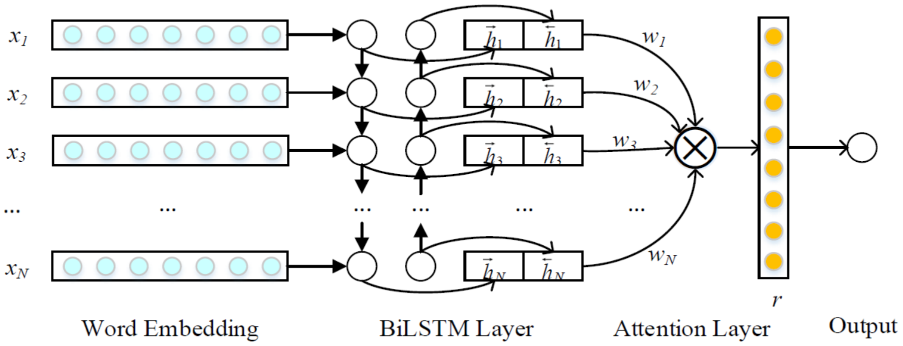
x_1
x_2
x_3
...
x_N
$\vec{h}_1$ $\overleftarrow{h}_1$
$\vec{h}_2$ $\overleftarrow{h}_2$
$\vec{h}_3$ $\overleftarrow{h}_3$
$\vec{h}_N$ $\overleftarrow{h}_N$
w_1
w_2
w_3
w_N
r
Word Embedding
BiLSTM Layer
Attention Layer
Output

Figure 1: Architecture of the BiLSTM with Attention Mechanism.

2 Overview

This section shows an overview of our system or experiments, which consists of three steps:(1) the data processing step, in which we use some text processing tools for preparing the data as input to the deep learning models, (2) the training step, where we build and train our models, and then predict and (3) evaluate our test results.

Task description. In all five subtasks, we take participant in all subtasks for English and Spanish (Mohammad et al., 2018). Subtasks EI-reg and V-reg, which require the system to detect emotion and sentiment intensity (a real-value score between 0 and 1) from given tweets, are both treated as regression problems. The difference between them is that subtask EI-reg has four different emotion sub-datasets (anger, fear, joy and sadness). In subtasks EI-oc and V-oc, we are given the message and scores, which are ordinal values from four-level and seven-level scales corresponding to positive or negative emotion and sentiment intensity, respectively. Subtask E-c is a multi-label task that requires the system to identify the tweets as ”no emotion” or as one, or more, of eleven given emotions.

2.1 Data processing

We built our text processing tools in order to utilize more information from the original text. The objectives of the tools are word-splitting, word annotation, processing of unknown word, and so on.

Text pre-processing. It is difficult to feed original tweets directly into a deep learning model. Imported from the NLTK API [1], the twitter-tokenizer shows great usefulness in fast word segmentation. The tokenizer is able to identify all the words, most of the emoticons and emojis, and omits all useless punctuation. The English (or Spanish) dataset primarily contains English (or Spanish) text. Therefore, all non-English (or non-Spanish) letters are treated as unknown words. Moreover, we converted all words to lower case and normalized the construction of user (@user), URLs (http://ie.com), and numbers and hashtags (#hashtag).

Pre-trained word embedding. Word embedding techniques aim to use continuous low-dimension vectors representing the features of the words (Mikolov et al., 2013), captured in context. For English tasks, a pre-trained word vector with a dimension of 300, which combined word embedding from training with the GloVe algorithm (Pennington et al., 2014) with the emoji embedding (Barbieri et al., 2016), which included most of the emoticons and emojis, was used to map the tweets. For the Spanish task, we used only the word embedding training by Barbieri et al. (2016). Unknown words were added to the vocabulary, and their vectors were randomly generated from a uniform distribution of $U(-0.25, 0.25)$. The pre-trained word embeddings were used for initializing the word embedding layer (the input layer) of our deep learning models.

2.2 Deep Learning models

Recently, most advanced work in NLP employs deep learning methods.

Convolutional Neural Networks (CNNs). Although CNNs were first applied for computer vision, they also show great importance for NLP problems (Zhang and Wallace, 2015). CNNs are able to quickly extract local n-gram features, and are easy to train. However, CNNs have difficulty

[1] http://www.nltk.org/.

capturing long-distance dependencies.

Recurrent Neural Networks (RNNs). Another effective neural network is the RNN, which captures dynamic information in serial data by periodically connecting hidden layer nodes. RNNs can store a state of context or even a story, learn and express relevant information in any long context window, unlike CNN's fixed-input formation. An RNN is able to overcome the problem of long-distance dependency. However, it is difficult to train because gradients may explode or vanish over long sequences (Hochreiter, 1998). One way to address this problem is by employing a variant of the regular RNN, the LSTM (Graves, 2012). LSTMs have a more complex internal structure with cells replacing RNN nodes, which allows LSTMs to remember information for either a long or short time.

Attention Mechanism. Between sequences, an attention mechanism shows a considerable improvement by changing the contribution of each word to the analysis of the whole text (Rocktschel et al., 2015; Raffel and Ellis, 2015). Before the RNN model summarizes the hidden states for the output, an attention mechanism amplifies the results by aggregating the hidden states and weighting their relative importance.

Domain Adaptation. Domain adaptation enhances learning in target domains by transferring learning from source domains that may have a distribution different from the target domain. Domain adaptation not only addresses the difference between source and target domains, but also pays attention to the relevance of both domains. The method provides an elegant way to access the full resources of similar tasks for target tasks (Mou et al., 2016).

Ensemble Learning. Ensemble learning is a supervised learning algorithm that ensembles two or more weak learners to amplify system performance (Maclin and Opitz, 1999). The AdaBoost algorithm (Li et al., 2008) is one of the ensemble learning algorithms that repeats training and adjusts the weights of all weak learners continuously to take into consideration the previous iteration error prediction samples. Therefore, the AdaBoost algorithm focuses more attention on a small proportion of special samples in a dataset for better scores.

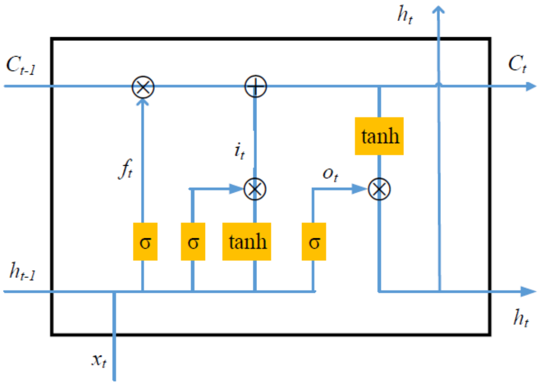

Figure 2: Architecture of cell in LSTM.

3 Model Description

We proposed the base BiLSTM model with an attention mechanism for subtask E-c (3.1). Two additional models (3.2 and 3.3) based on the BiLSTM with an attention mechanism are used for other subtasks.

4 BiLSTM with Attention Mechanism (BiLSTM$_{ATT}$)

Figure 1 shows the architecture of BiLSTM with an attention mechanism, which has four different layers as follows.

Embedding Layer. After the pre-processing of text, tweets are transformed into a sequence of words,$X = (x_1, x_2, ..., x_N), X \in R^{N\times d}$, where N is the number of a tweet, and d denotes the dimension of a word vector. The word tokens are then directly fed into the model embedding layer, which was initialized by the pre-trained word embeddings.

BiLSTM Layer. LSTM replaces the nodes of a regular RNN model with special structures (cells). The architecture of the LSTM is shown in Figure 2. It calculates the hidden state h_t at time t using the following equations:

- Gates

$$\begin{aligned} f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \\ i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\ o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \end{aligned} \quad (1)$$

- Transformation

$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (2)$$

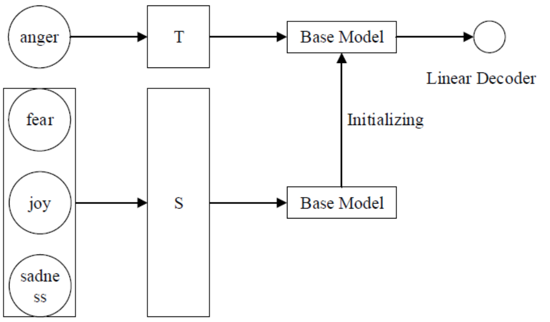

Figure 3: The Model of EIM. Here anger sub-dataset is the target domain and other three sub-datasets regarded as source domain.

- State update

$$\begin{aligned} C_t &= f_t * C_{t-1} + i_t * \tilde{C_t} \\ h_t &= o_t * \tanh(C_t) \end{aligned} \tag{3}$$

where σ denotes the sigmoid function, x_t is the t-th word vector, C_t, f_t, i_t and o_t are all gate vectors of the cell, and W and b are cell parameters.

We use bidirectional LSTM so as to obtain word features $H = (h_1, h_2, ..., h_n)$ concatenated from both directions. A forward LSTM processes the tweet from x_1 to x_n, while a backward LSTM processes from x_n to x_1. For word x_t, a forward LSTM obtains a word feature as $\overrightarrow{h}$ and a backward LSTM obtains the feature as $\overleftarrow{h}$. Then, h is calculated as follows:

$$h_i = \overrightarrow{h_i} \oplus \overleftarrow{h_i}, h_i \in R^{2L} \tag{4}$$

Where $\oplus$ denotes the function of concatenation and L is the size of the one-directional LSTM.

Attention Layer. We add an attention layer for finding the contribution of each word to the whole sequence. The attention mechanism assigns a weight w_i to each word feature h_i with a focus on results. The hidden states are finally calculated to produce a hidden sentence feature vector r by a weighted sum function. Formally:

$$\begin{aligned} &e_i = \tanh(W_h h_i + b_h), e_i \in [-1, 1] \\ &w_i = \frac{\exp(e_i)}{\sum_{t=1}^{N} \exp(e_t)}, \sum_{i=1}^{N} w_i = 1 \\ &r = \sum_{i=1}^{N} w_i h_i, r \in R^{2L} \end{aligned} \tag{5}$$

where W_h and b_h are the weight and bias from the attention layer.

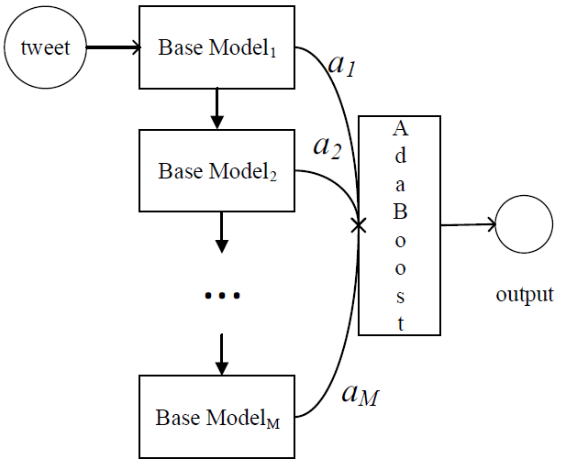

Figure 4: The Model of SIM

Output Layer. The representation r is a sentence feature vector, which we put into a fully-connected layer that outputs the results for the whole sentence. Different tasks require different forms of the output. This base model is dedicated to subtask E-c, with eleven fully-connected sigmoid layers as the output layer.

4.1 Emotion Intensity Model (EIM)

Figure 3 shows an overview of the EIM, the model we used for task EI-reg and EI-oc with more than one sub-dataset. To train one emotion dataset as a target task (T), the other three emotion datasets were treated as source tasks (S). Our approach was to first train the base model on S, and then to directly initialize the base model on T using the tuned parameters. The parameters were then fine-tuned for predicting the results of S. The output layer of the EIM uses the linear decoder for regression. For ordinal classification task EI-oc, real-value scores from the EIM are translated into four-point classes with thresholds according to the training sets for EI-reg and EI-oc.

4.2 Sentiment Intensity Model (SIM)

Figure 4 shows the architecture of the SIM. Based on the base model, we use the AdaBoost algorithm ensemble the M weak learners to a stronger learner for subtask V-reg and V-oc. Initially, each sample has the same weight. After each iteration, the algorithm weights the samples with poor predictions by the previous learner, and the weighted samples are again used to train the next learner. Finally, we use the calculated weight a_i of each

Model	subtask EI-reg				subtask V-reg
	p				p
	anger	fear	joy	sadness	
CNN	0.428	0.498	0.501	0.631	0.700
LSTM	0.551	0.522	0.560	0.500	0.762
CNN-LSTM	0.521	0.532	0.592	0.555	0.753
BiLSTM	0.511	0.533	0.535	0.5003	0.718
BiLSTM$_{ATT}$	0.555	0.655	0.605	0.700	0.773
EIM	**0.654**	**0.715**	**0.630**	**0.728**	-
SIM	0.558	0.659	0.621	0.713	**0.787**

Table 1: Comparable results of experiments for subtask EI-reg and V-reg.

learner for the weighted sum of scores. The output layer of the SIM is the same as the one in the task EI-reg. The results of task V-oc are obtained from the real-value scores of the SIM with thresholds according to the training sets for V-reg and V-oc.

4.3 Training and Hyper-parameters

We train the model for task E-c using the categorical cross-entropy loss function, and for other tasks using mean squared error. For all tasks, we use the Adam (Kingma and Ba, 2014) optimizer to train models, and the Relu activation function for fast calculation. An early stopping (Prechelt, 1998) strategy is used to prevent over-fitting. All models use stochastic gradient descent with mini-batches of size 32.

Hyper-parameters. The dimension of word embeddings (d) is 300; the number of each LSTM (L) is 100; the dropout ratio is 0.25 at all layers for all models. Finally, we set 30 learners from the base model to train the SIM by ensemble learning.

5 Experiment

Corpus. The datasets we used were all provided by the competition, with no other external corpus. Except for subtasks EI-reg and EI-oc, which had four sub-datasets, subtasks had only one dataset each for English and Spanish. We thank Mohammad and Kiritchenko (Mohammad et al., 2013) for contributions to the data.

Evaluation Measure. For regression and ordinal tasks (including task EI-reg, EI-oc, V-reg, and V-oc), the official competition metric was the value (p) of the Pearson Correction Coefficient. Moreover, tasks EI-oc and V-oc have a second metric, the quadratic weighted kappa (k). For the multi-label task (task E-c), apart from the official competition metric (multi-label accuracy, a), a micro-averaged F-score $(f1^{micro})$ and a macro-averaged F-score $(f1^{macro})$ were also calculated for our submissions.

Results. On the competition leaderboard, our system placed 22/48 (9/24) for English (Spanish) in task EI-reg, 12/39 (8/16) in task EI-oc, 27/38 (7/14) in task V-reg, 14/31 (6/14) in task V-oc and 7/35 (6/14) in task E-c.

Experiments and Analysis. We trained our models on the training set and evaluated the prediction with the golden scores of the development set. In order to illustrate the good performance of our methods, we compare the results with baseline models of CNN, LSTM, CNN-LSTM (Zhang et al., 2017) and a regular BiLSTM. From the results shown in Table 1, we can see that our approach achieved a significant result. A regular LSTM tends to ignore future contextual information while processing sequences in a time series. The BiLSTM is able to use both past and future contexts by processing the text from both directions. Not all words make the same contribution to sentiment analysis in the text. The attention mechanism is able to shuffle the word annotation weights according to their importance to the meaning of sentence. We can see that the attention based BiLSTM obtained higher scores than the BiLSTM without the attention mechanism. Moreover, the SIM and the EIM showed their best performance on subtasks V-reg and EI-reg, respectively. SIM employed the AdaBoost algorithm so as to integrate 30 the models of BiLSTM$_{ATT}$. The SIM was able to adapt to the training error rate of each learner, so that the whole system was improved effectively. The EIM fine-tuned the parameters for the multi-task approach, which made full use of associated sub-datasets of the task EI-reg. Before training the target dataset, the special parameter initialization gave the target model additional knowledge from the other source datasets. In addition, for the same training tweets that were used in task EI-reg (or V-reg) and EI-oc (or V-oc), we defined the threshold for translating from real-value score to ordinal classes by referring to the training labels across the training dataset.

6 Conclusion

In this paper, we described our deep learning models for the sentiment analysis task SemEval-2018 shared Task 1: Affect in Tweets. We used the BiLSTM with an attention mechanism as a base model and built the SIM and EIM for all subtasks. The final system for submission achieved good results. We would like to further explore text sentiment analysis, and employ more interesting methods for NLP problems.

References

Francesco Barbieri, German Kruszewski, Francesco Ronzano, and Horacio Saggion. 2016. How cosmopolitan are emojis?: Exploring emojis usage and meaning over different languages with distributional semantics. In *ACM on Multimedia Conference*, pages 531–535.

Catherine Blake. 2007. The role of sentence structure in recognizing textual entailment. In *Acl-Pascal Workshop on Textual Entailment and Paraphrasing*, pages 101–106.

Stavros Giorgis, Apostolos Rousas, John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2016. aueb.twitter.sentiment at semeval-2016 task 4: A weighted ensemble of svms for twitter sentiment analysis. In *International Workshop on Semantic Evaluation*, pages 96–99.

Alex Graves. 2012. *Long Short-Term Memory*. Springer Berlin Heidelberg.

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.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Computer Science*.

Xuchun Li, Lei Wang, and Eric Sung. 2008. Adaboost with svm-based component classifiers. *Engineering Applications of Artificial Intelligence*, 21(5):785–795.

R. Maclin and D. Opitz. 1999. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *Computer Science*.

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.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Computer Science*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications?

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

L Prechelt. 1998. Automatic early stopping using cross validation: quantifying the criteria. *Neural Networks the Official Journal of the International Neural Network Society*, 11(4):761.

Colin Raffel and Daniel P. W. Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems.

Tim Rocktschel, Edward Grefenstette, Karl Moritz Hermann, TomKoisky, and Phil Blunsom. 2015. Reasoning about entailment with neural attention.

Richard Socher, Yoshua Bengio, and Christopher D. Manning. 2013. Deep learning for nlp (without magic). *Acl Tutorial*.

Ye Zhang and Byron Wallace. 2015. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *Computer Science*.

You Zhang, Hang Yuan, Jin Wang, and Xuejie Zhang. 2017. Ynu-hpcc at emoint-2017: Using a cnn-lstm model for sentiment intensity prediction. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 200–204, Copenhagen, Denmark. Association for Computational Linguistics.