# thecerealkiller at SemEval-2016 Task 4: Deep Learning based System for Classifying Sentiment of Tweets on Two Point Scale

**Vikrant Yadav**
Amazon.in
Hyderabad, India
vikrantiitr1@gmail.com

## Abstract

In this paper, we propose a deep learning system for classification of tweets on a two-point scale. Our architecture consists of a multilayered recurrent neural network having gated recurrent units. The network is pre-trained with a weakly labeled dataset of tweets to learn the sentiment specific embeddings. Then it is fine tuned on the given training dataset of the task 4B in SemEval-2016. The network does very little pre-processing for raw tweets and no post-processing at all. The proposed system achieves 3rd rank on the leaderboard of task 4B.

## 1 Introduction

Task 4 of SemEval-2016 (Nakov et al., 2016) - Sentiment Analysis in Twitter- turned out to be the most popular task of SemEval-2016. Among its subtasks, sub-task B - Tweet classification according to a two-point scale - was the 2nd most popular subtask. A total of 19 teams participated in it including ours.

In this paper, we propose a multilayerd RNN architecture for classifying tweets on a two-point scale, namely, positive and negative. We first pretrain the network with a weakly labeled corpus of tweets, where labels are assigned based on the sentiment of the emoticon present in the tweets. It helps the network to learn sentiment specific embeddings of the words in the tweet. The network is then fine-tuned on the dataset of tweets provided as part of the sub-task 4B along with the training and development dataset of SemEval-2013 task 2.

In the second section, we describe our architecture. In third section, we explain our approach to train the network. After that, we describe the experimental setup and statistical properties of the data used. In the end, we discuss our results on the test dataset. We stood 3rd on the final leaderboard for sub-task 4B.

## 2 Proposed System

In this section, we elaborate our proposed architecture. The network is fed pre-processed tweets as input and it predicts the binary label of the tweets. It has 3 RNN layers each with gated recurrent units. Then a sum layer is added to sum the hidden states of the last recurrent layer over time. After that, a dense layer is present followed by a single sigmoid unit.

Now, we describe the components of our architecture in brief.

### 2.1 Pre-processing

The tweets are pre-processed to remove any punctuation if present. All URLs are encoded into a `URL` token. All the user accounts mentioned in a tweet are encoded as `USER` token. The tweet is converted to lower-case before feeding to the network. Note that we don't remove any stop-words as they define useful relationships between words and phrases. This amount of pre-processing turned out to be sufficient for the network to learn useful semantic and sentiment specific word-embeddings.



*Proceedings of SemEval-2016*, pages 100–102,
San Diego, California, June 16-17, 2016. 

### 2.2 Embedding layer

The embedding layer maps a sequence of words present in the input tweet to the corresponding fixed length real valued vectors. The fixed length `d` is called the dimension of the embeddings. The embedding layer keeps track of the mapping so that the correct embeddings gets updated while doing back-propagation of the errors.

### 2.3 Recurrent layers

Recurrent neural networks are proved to be useful in handling variable length sequences. A stacking of recurrent layers on the top of each other allows the semantic composition of representations of words and phrases over time. We use the same intuition in our architecture.

The embedding layer passes the embedding matrix to the first recurrent layer. Each layer takes into input an embedding matrix which it processes i.e. updates its hidden state over time. The hidden states are stored after each time-step and fed to the next layer as input. Thus, the last layer outputs a matrix of hidden states.

We prefer GRU(Gated Recurrent Unit) (Cho et al., 2014) compared to a vanilla RNN unit. A classic RNN is difficult to train, because the gradients either tend to vanish or explode (Bengio et al., 1994). A GRU unit takes care of this problem and is able to cope with vanishing or exploding gradients while capturing the information for longer periods of time.

### 2.4 Sum over time

This layer receives the time-distributed hidden state matrix of the last recurrent layer as input where the nth column describes the hidden state at nth time-step. It outputs a vector where the kth element is sum of the kth row in the hidden state matrix. This helps to combine the sentiment specific representation of the phrases so as to yield an aggregate representation.

### 2.5 Dense layer and output layer

The output of the sum layer is fully connected with the dense layer consisting of rectified linear units. The dense layer in turn is connected with the output sigmoid units which predicts the probability of assigning a positive or negative label for the input tweet.

## 3 Approach to Train the Network

In this section, we describe our choice of training algorithm and the regularization method.

### 3.1 Training algorithm

We use mini-batch gradient descent algorithm as our choice of training algorithm. We utilized two Nvidia GK104 series GPU hardware to make matrix-matrix multiplications efficient. A mini-batch size of 128 is chosen for pre-training the network. We use rmsprop as an update rule for the parameters, an optimizer which divides the gradient by an exponential moving average of its squares.

### 3.2 Regularization

We use dropout (Srivastava, 2013) as the regularizer to prevent our network from overfitting. Dropout selects a fraction of the hidden units at random and sets their output to zero and thus, prevents co-adaptation of the features. However, it is tricky to be applied in the RNN as it is capable of unsettling the recurrent connections and thus, interfere with our recurrent layer's ability of retaining information for longer periods of time.

We choose the approach proposed in (Zaremba et al., 2014) to apply dropout in our network where it is being applied between inter-layer connections instead of intra-layer connections. This doesnt interfere with the recurrent updates in a layer and helps prevent co-adaptation of the features at the same time.

### 3.3 Pre-training the network

The training dataset provided as part of sub-task 4B in Semeval-2016 contain very few samples to effectively train our deep architecture. Thus, we used a weakly labeled corpus of tweets, namely, Sentiment140 (Go et al., 2009), to pre-train our network so as to learn semantic and sentiment specific representation of words and phrases.

We take the learned weights of the trained network as it is and fine tune them on the provided training dataset of sub-task 4B along with training and development dataset of task 2 in SemEval-2013. The network uses validation scores as the metric to do an early-stop while training. Macro-averaged recall, the official scoring metric of sub-task 4B, was

| Type | Dataset | Positives | Negatives |
|---|---|---|---|
| Training | Twitter'16 + Twitter'13 | 8250 | 2500 |
| Testing | Twitter'16 | 8212 | 2337 |

**Table 1:** Statistical information of training and testing datasets.

| Scoring-metric | Score | Best-score | Rank |
|---|---|---|---|
| **AvgR** | **0.784** | **0.797** | **3** |
| AvgF1 | 0.762 | 0.799 | 5 |
| Accuracy | 0.823 | 0.862 | 9 |

**Table 2:** Resulting scores on testing dataset. AvgR was the official scoring metric of the task 4B.

also used as the validation scoring metric.

## 4 Experiments

### 4.1 Experimental settings

The statistical properties of training and testing datasets are provided in Table 1.

For evaluation, we use official scoring metric of Semeval-2016 task 4B - macro-averaged recall - average of recalls for both positive and negative classes.

The chosen parameters of our network are as follows: the maximum input sequence length is set to 30, vocabulary size is 400000, dimensionality of word embedding (d) is 100, recurrent units hidden state vector size is 128, number of recurrent layers is 3, number of hidden unit in dense layer is 256 with relu activation. We used a dropout of 50% after each layer while training.

### 4.2 Results

Results are shown in Table 2.

Our system produces a macro-averaged recall of 0.784, while the best system scored 0.797. Our system's performance with other scoring metrics is also good, achieving 5th and 9th rank for macro-averaged F1 and accuracy metric, respectively.

Our architecture uses very little pre-processing compared to the other systems of Semeval-2016. It is able to capture useful semantic relationships and sentiment specific embeddings of words and phrases using just the raw tweets. It can be improved by adding handcrafted features such as topics, number of positive-negative lexicons, etc. which we would like to try in future.

## 5 Conclusion

In this paper, we proposed a deep learning system for sentiment classification of tweets on a two-point scale. Our architecture was able to capture complex semantic relationships between words and phrases of the input tweets to decide their final sentiment. We show how to pre-train and fine-tune a deep network like ours well from end-to-end using weakly supervised dataset. Our system used very little pre-processing before feeding the raw tweets to the network. In future, we would like to use handcrafted features in addition to the raw tweets to see if they improve the overall score.

## References


Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Trans. Neur. Netw.*, 5(2):157–166, March.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report, Stanford University.

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.

Nitin Srivastava. 2013. *Improving neural networks with dropout. PhD thesis*. University of Toronto.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.