Senti17 at SemEval-2017 Task 4: Ten Convolutional Neural Network Voters for Tweet Polarity Classification

Hussam Hamdan
Labex Observatoire de la vie littraire (OBVIL)
Laboratoire d'Informatique de Paris 6 (LIP6), Pierre and Marie Curie University, UMR 7606
4 place Jussieu, 75005, Paris, France
Hussam.Hamdan@lip6.fr

Abstract

This paper presents Senti17 system which uses ten convolutional neural networks (ConvNet) to assign a sentiment label to a tweet. The network consists of a convolutional layer followed by a fully-connected layer and a Soft- max on top. Ten instances of this network are initialized with the same word embeddings as inputs but with different initializations for the network weights. We combine the results of all instances by selecting the sentiment label given by the majority of the ten voters. This system is ranked fourth in SemEval-2017 Task4 over 38 systems with 67.4% average recall.

1 Introduction

Polarity classification is the basic task of sentiment analysis in which the polarity of a given text should be classified into three categories: positive, negative or neutral. In Twitter where the tweet is short and written in informal language, this task needs more attention. SemEval has proposed the task of Message Polarity Classification in Twitter since 2013, the objective is to classify a tweet into one of the three polarity labels (Rosenthal et al., 2017).

We can remark that in 2013, 2014 and 2015 most best systems were based on a rich feature extraction process with a traditional classifier such as SVM (Mohammad et al., 2013) or Logistic regression (Hamdan et al., 2015). In 2014, Kim (2014) proposed to use one convolutional neural network for sentence classification, he fixed the size of the input sentence and concatenated its word embeddings for representing the sentence, this architecture has been exploited in many later works. Severyn and Moschitti (2015) adapted the convolutional network proposed by Kim (2014) for sentiment analysis in Twitter, their system was ranked second in SemEval-2015 while the first system (Hagen et al., 2015) combined four systems based on feature extraction and the third ranked system used logistic regression with different groups of features (Hamdan et al., 2015).

In 2016, we remark that the number of participations which use feature extraction systems were degraded, and the first four systems used Deep Learning, the majority used a convolutional network except the fourth one (Amir et al., 2016). Despite of that, using Deep Learning for sentiment analysis in Twitter has not yet shown a big improvement in comparison to feature extraction, the fifth and sixth systems (Hamdan, 2016) in 2016 which were built upon feature extraction process were only (3 and 3.5% respectively) less than the first system. But We think that Deep Learning is a promising direction in sentiment analysis. Therefore, we proposed to use convolutional networks for Twitter polarity classification.

Our proposed system consists of a convolutional layer followed by fully connected layer and a softmax on top. This is inspired by Kim (2014), we just added a fully connected layer. This architecture gives a good performance but it could be improved. Regarding the best system in 2016 (Deriu et al., 2016), it uses different word embeddings for initialisation then it combines the predictions of different nets using a meta-classifier, Word2vec and Glove have been used to vary the tweet representation.

In our work, we propose to vary the neural network weights instead of tweet representation which can get the same effect of varying the word embeddings, therefore we vary the initial weights of the network to produce ten different nets, a voting system over the these ten voters will decide the sentiment label for a tweet.

The remaining of this paper is organized as follows: Section 2 describes the system architecture, Section 3 presents our experiments and results and Section 4 is devoted for the conclusion.

2 System Architecture

The architecture of our convolutional neural network for sentiment classification is shown on Fig. 1. Our network is composed of a single convolutional layer followed by a non-linearity, max pooling, Dropout, fully connected layer and a soft-max classification layer. Here we describe this architecture:

2.1 Tweet Representation

We first tokenize each tweet to get all terms using HappyTokenizer[1] which captures the words, emoticons and punctuations. We also replace each web link by the term *url* and each user name by *uuser*. Then, we used Structured Skip-Gram embeddings (SSG) (Ling et al., 2015) which was compiled by (Amir et al., 2016) using 52 million tweets.

Each term in the tweet is replaced by its SSG embedding which is a vector of d dimensions, all term vectors are concatenated to form the input matrix where the number of rows is d and the number of columns is set to be maxl: the max tweet length in the training dataset. This 2-dim matrix is the input layer for the neural network.

2.2 Convolutional Layers

We connect the input matrix with different convolutional layers, each one applies a convolution operation between the input matrix and a filter of size m x d. This is an element-wise operation which creates f vectors of $maxl\text{-}m\text{+}1$ dimension where f is the number of filters or feature maps.

This layer is supposed to capture the common patterns among the training tweets which have the same filter size but occur at any position of the tweet. To capture the common patterns which have different sizes we have to use more than one layer therefore we defined 8 different layers connected to the input matrix with different filter sizes but the same number of feature maps.

[1]http://sentiment.christopherpotts.net/tokenizing.html

2.3 Activation Layer

Each convolutional layer is typically followed by a non-linear activation function, RELU (Rectified Linear Unit) layer will apply an element-wise operation to swap the negative numbers to 0. The output of a ReLU layer is the same size as the input, just with all the negative values removed. It speeds up the training and is supposed to produce more accurate results.

2.4 Max-Pooling Layer

This layer reduces the size of the output of activation layer, for each vector it selects the max value. Different variation of pooling layer can be used: average or k-max pooling.

2.5 Dropout Layer

Dropout is used after the max pooling to regularize the ConvNet and prevent overfitting. It assumes that we can still obtain a reasonable classification even when some of the neurones are dropped. Dropout consists in randomly setting a fraction p of input units to 0 at each update during training time.

2.6 Fully Conected Layer

We concatenate the results of all pooling layers after applying Dropout, these units are connected to a fully connected layer. This layer performs a matrix multiplication between its weights and the input units. A RELU non-linarity is applied on the results of this layer.

2.7 Softmax Layer

The output of the fully connected layer is passed to a Softmax layer. It computes the probability distribution over the labels in order to decide the most probable label for a tweet.

3 Experiments and Results

For training the network, we used about 30000 English tweets provided by SemEval organisers and

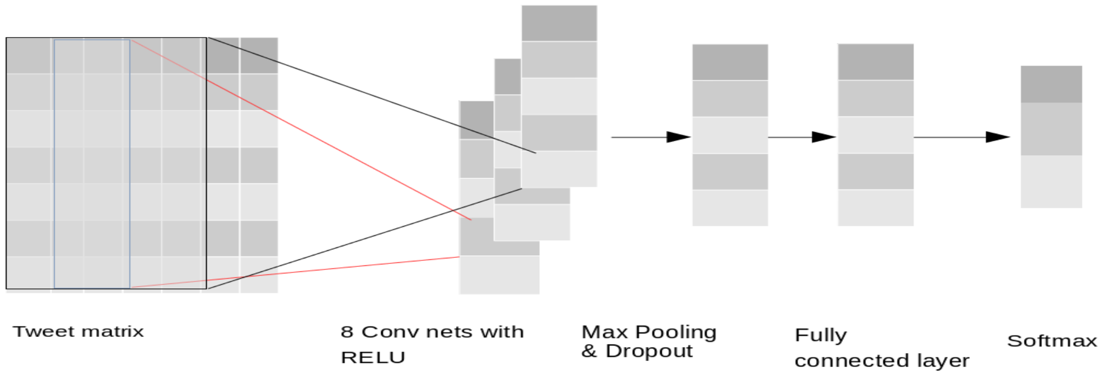

Figure 1: Network architecture.

the test set of 2016 which contains 12000 tweets as development set. The test set of 2017 is used to evaluate the system in SemEval-2017 competition. For implementing our system we used python and Keras[2].

We set the network parameters as follows: SSG embbeding size d is chosen to be 200, the tweet max legnth *maxl* is 99. For convolutional layers, we set the number of feature maps f to 50 and used 8 filter sizes (1,2,3,4,5,2,3,4). The p value of Dropout layer is set to 0.3. We used Nadam optimizer (Dozat, 2015) to update the weights of the network and back-propogation algorithm to compute the gradients. The batch size is set to be 50 and the training data is shuffled after each iteration.

We create ten instances of this network, we randomly initialize them using the uniform distribution, we repeat the random initialization for each instance 100 times, then we pick the networks which gives the highest average recall score as it is considered the official measure for system ranking. If the top network of each instance gives more than 95% of its results identical to another chosen network, we choose the next top networks to make sure that the ten networks are enough different.

Thus, we have ten classifiers, we count the number of classifiers which give the positive, negative and neutral sentiment label to each tweet and select the sentiment label which have the highest number of votes. For each new tweet from the test set, we convert it to 2-dim matrix, if the tweet is longer than *maxl*, it will be truncated. We then feed it into the ten networks and pass the results to the voting system.

[2]https://keras.io

Official ranking: Our system is ranked fourth over 38 systems in terms of macro-average recall. Table 4 shows the results of our system on the test set of 2016 and 2017.

Test Dataset	Avg. Recall	Accuracy	F-score
Test 2017	0.674	0.652	0.665
Test 2016	0.692	0.650	0.643

Table 1: Table 1: Senti17 results on the test sets of 2016 and 2017.

4 Conclusion

We presented our deep learning approach to Twitter sentiment analysis. We used ten convolutional neural network voters to get the polarity of a tweet, each voter has been trained on the same training data using the same word embeddings but different initial weights. The results demonstrate that our system is competitive as it is ranked forth in SemEval-2017 task 4-A.

References

Silvio Amir, Ramn Fernndez Astudillo, Wang Ling, Mrio J. Silva, and Isabel Trancoso. 2016. INESC-ID at SemEval-2016 Task 4-A: Reducing the Problem of Out-of-Embedding Words. In *SemEval@NAACL-HLT*.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurlien 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. In *SemEval@NAACL-HLT*.

Timothy Dozat. 2015. Incorporating Nesterov Momentum into Adam.

Matthias Hagen, Martin Potthast, Michel Bchner, and Benno Stein. 2015. Webis: An Ensemble for Twitter Sentiment Detection.

Hussam Hamdan, Patrice Bellot, and Frederic Bechet. 2015. Lsislif: Feature Extraction and Label Weighting for Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 568–573, Denver, Colorado, June. Association for Computational Linguistics.

Hussam Hamdan. 2016. SentiSys at SemEval-2016 Task 4: Feature-Based System for Sentiment Analysis in Twitter. In *SemEval@NAACL-HLT*.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. *CoRR*, abs/1408.5882.

Wang Ling, Chris Dyer, Alan Black, and Isabel Trancoso. 2015. Two/Too Simple Adaptations of word2vec for Syntax Problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Denver, Colorado. Association for Computational Linguistics.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRCCanada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *In Proceedings of the International Workshop on Semantic Evaluation, SemEval 13*.

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 '17, Vancouver, Canada, August. Association for Computational Linguistics.

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)*, pages 464–469, Denver, Colorado, June. Association for Computational Linguistics.