MDSENT at SemEval-2016 Task 4: A Supervised System for Message Polarity Classification

Hang Gao and **Tim Oates**
hanggao1@umbc.edu, oates@cs.umbc.edu
University of Maryland Baltimore County
11000 Hilltop Circlet
Baltimore, MD 21250, USA

Abstract

This paper describes our system submitted for the Sentiment Analysis in Twitter task of SemEval-2016, and specifically for the Message Polarity Classification subtask. We used a system that combines Convolutional Neural Networks and Logistic Regression for sentiment prediction, where the former makes use of embedding features while the later utilizes various features like lexicons and dictionaries.

1 Introduction

Recently, rapid growth of the amount of user-generated content on the web prompts increasing interest in research on sentiment analysis and opinion mining. A typical example is Twitter, where lots of users express feelings and opinions about various subjects. However, unlike traditional media, language used in social network services like Twitter is often informal, leading to new challenges to corresponding text analysis.

The SemEval-2016 Sentiment Analysis in Twitter task (SESA-16) is a task that focuses on the sentiment analysis of tweets. As a continuation of SemEval-2015 Task 10, SESA-16 introduces several new challenges, including the replacement of classification with quantification, movement from two/three-point scale to five-point scale, etc.

We participated in Subtask A of SESA-16, namely message polarity classification, a task that seeks to predict a sentiment label for some given text. We model the problem as a multi-class classification problem that combines the predictions given by two different classifiers: one is a Convolutional Neural Network (CNN) and the other is Logistic Regression (LR). The former takes embedding-based features while the latter utilizes various features such as lexicons, dictionaries, etc.

The remainder of this paper is structured as follows. In Section 2, we describe our system in detail, including feature description and approaches. In Section 3, we list the details of datasets for the experiments, along with hyperparameter settings and training techniques. In Section 4, we report the experiment results and present the corresponding discussion.

2 System Description

Our system aims at predicting the sentiment of a given message, i.e., whether the message expresses positive, negative or neutral emotion. To achieve that, we adopt two separate classifiers, CNN and LR, designed to utilize different types of features. The final prediction for sentiment is a combination of predictions given by both classifiers.

2.1 Data Preprocessing

Tweets often include informal text, making it essential to preprocess tweets before they are fed to the system. However, we keep the preprocessing to a minimum by only removing URLs and @User tags. We then further tokenize and tag tweets with arktweetnlp (Gimpel et al., 2011). In addition, all tweets are lower-cased.

2.2 Logistic Regression

We use the LR classifier for features from sentiment lexicons and token clusters. We have used the fol-

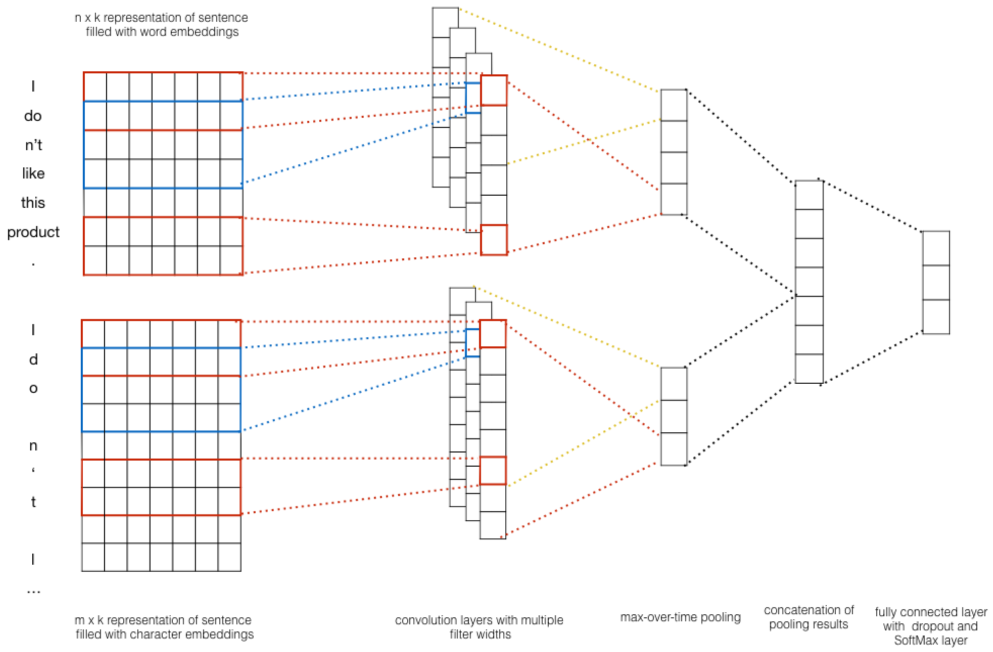
n x k representation of sentence filled with word embeddings
I
do
n't
like
this
product
.
I
d
o
n
'
t
I
...
m x k representation of sentence filled with character embeddings
convolution layers with multiple filter widths
max-over-time pooling
concatenation of pooling results
fully connected layer with dropout and SoftMax layer

Figure 1: CNN architecture for an example with both word-based and character-based input maps.

lowing:

- **clusters**: 1000 token clusters provided by the CMU tweet NLP tool. These clusters are produced with the Brown clustering algorithm on 56 million English-language tweets.
- **manually-constructed sentiment lexicons**: NRC Emotion Lexicon (Mohammad and Turney, 2010), MPQA (Wilson et al., 2005), Bing Liu Lexicon (Hu and Liu, 2004) and AFINN-111 Lexicon (Nielsen, 2011).
- **automatically-constructed sentiment lexicons**: Hashtag Sentiment Lexicon and Sentiment140 Lexicon (Mohammad et al., 2013).

For the Sentiment140 Lexicon and Hashtag Sentiment Lexicon, we compute separate lexicon features for uni-grams and bi-grams, while for other Lexicons, only uni-gram lexicon features are produced. For each lexicon, let t be the token(uni-gram or bi-gram), p be the polarity and s be the score provided by the lexicon. We use the same features that are also adopted by the NRC-Canada system (Mohammad et al., 2013):

- the total count of tokens in a tweet with $s(t,p) > 0$.
- the total score of tokens in a tweet $\sum_w s(t,p)$.
- the maximum score of tokens in a tweet $\max_w s(t,p)$.
- the score of the last token in the tweet with $s(t,p) > 0$.

For each token, we also use features to describe whether it is present or absent in each of the 1000 token clusters. There are in total 1051 features for a tweet.

2.3 Convolutional Neural Network

Deep learning models have achieved remarkable results for various NLP tasks, with most of them based on embeddings that represent words, characters, etc. with vectors of real values. Some work on embeddings suggests that word vectors generated by some embedding algorithms preserve many linguistic regularities (Mikolov et al., 2013a).

Among the various deep learning models, we use Convolutional Neural Networks, which have already been used for sentiment classification with promising results (Kim, 2014). We show the network architecture in Figure 1.

In general, the architecture contains two separate CNNs: one is for word-based input maps while the other is for character-based input maps. In our system, an input map for a tweet is a stack of the embeddings of its words/characters w.r.t. their order in the tweet. We initialize word embeddings with the publicly available 300 dimension Google News embeddings trained with Word2Vec, but randomly initialize character embeddings with the same dimension. We fine tune both kinds of embeddings during the training procedure.

Each of the two separate CNNs has its own set of convolutional filters. We fix the width of all filters to be the same as the corresponding embedding dimension, but set their height according to predefined types of n-grams. For example, a filter for bi-grams on an input map constructed with 300 dimensional word embeddings will have shape (2, 300), where 2 is the height and 300 is the width. In other words, we use each filter to capture and extract features w.r.t. a specific type of n-gram from an input map.

The feature maps generated by a particular filter may have different shapes for different input maps, due to variable tweet lengths. Thus we adopt a pooling scheme called max-over-time pooling (Collobert et al., 2011), which captures the most important feature, i.e., the one with highest value, for each feature map. This pooling scheme naturally deals with the variable tweet length problem.

After pooling, we first generate a representation for each CNN by concatenating its own pooled features, and then form a final representation by concatenating the two separate representations. The final representation is then fed into a multi-layer perceptron (MLP) classifier for predictions.

2.3.1 Regularization

For regularization we employ dropout with a constraint on l_2-norms of the weight vectors (Hinton et al., 2012). The key idea of dropout is to prevent coadaptation of feature detectors (hidden units) by randomly dropping out a portion of hidden units in the training procedure. At test time, the learned weight vectors are scaled according to the portion while no dropout is needed.

In addition to dropout, we constrain weight vectors by introducing an upper limit on their l_2-norms. That is, for a weight vector w, we rescale it to have $||w||_2 = l$, whenever it has $||w||_2 > l$, after gradient descent step.

2.4 Combination

We combine the predictions of the two classifiers in the form of a weighted summation. Given the prediction P_{LR} by Logistic Regression and the prediction P_{CNN} by the CNN, we introduce a scalar w, such that the final prediction is given as,

$$P_{final} = (1 - w)P_{LR} + wP_{CNN} \tag{1}$$

In other words, let x be the input instance,

$$\begin{aligned} P_{final}(Y = y|x) = wP_{CNN}(Y = y|x) \\ + (1 - w)P_{LR}(Y = y|x) \end{aligned} \tag{2}$$

We do not simply feed the features of LR along with the features generated by the CNN into a single classifier because they are naturally different. The features from LR are highly relevant with manually-created or automatically-generated dictionaries, scores, clusters, etc. They are a mixture of binary and real-value features with high variance. While for the CNN, the features are generated by convolutional kernels on distributed representations (embeddings), leading to strong correlation and relatively smaller variance. Our preliminary experiments show that by simply adding LR features to CNN features, the performance of our system does not increase, but drops.

3 Experiment

3.1 Datasets

We test our model on the SemEval-2016 benchmark dataset with two different settings. Setting 1 uses only the 2016 datasets while Setting 2 uses a combination of 2016 and 2013 datasets. We list the details of the two settings in Table 1.

For setting 2, the merge of two datasets is conducted w.r.t. the train/dev splits. Although we did

Settings	Train	Dev	Test
Setting 1	5975	1997	32009
Setting 2	12964	3100	32009

Table 1: Statistics of our two settings of datasets for experiments. Setting 1: a dataset with only the SemEval-2016 dataset. Setting 2: a dataset that is a combination of the SemEval-2016 and SemEval-2013 datasets. In Setting 2, the merge is conducted w.r.t. train/dev splits, with "Not Available" tweets removed.

not remove any "Not Available" tweets for setting 1, we found a relatively high amount of such tweets in the combined dataset, which may significantly influence the system performance, thus we removed all the "Not Available" tweets for setting 2.

3.2 Hyperparameters and Training

3.2.1 CNN

For both settings, we use rectified linear units. For the word-based CNN, we use filters of height 1,2,3,4, while for the character-based CNN, we use filters of height 3,4,5. And 100 feature maps are used for each filter. We also use a dropout rate of 0.5, l_2-norm constraint of 3, and mini-batch size of 50. These values were picked on the Dev dataset of Setting 1.

We perform early stop on dev datasets during training. We use Adadelta as the optimization algorithm (Zeiler, 2012).

3.2.2 LR

We use the publicly available tool LibLinear for LR training. The cost is set to be 0.5 with all other parameters assigned with default settings. The cost is chosen based on the Dev dataset of Setting 1.

3.2.3 Combination

The scalar w is picked via grid search on the Dev dataset for both settings. Because of the random initialization of weights and random shuffling of batches for the CNN during the training procedure, w is different for different runs. Thus we consider it as a weight to be trained with other weights.

3.3 Embeddings

It is popular to initialize word vectors with pre-trained embeddings obtained by some unsupervised algorithms trained over a large corpus to improve system performance (Kim, 2014) (Socher et al., 2011). We use the publicly available Word2Vec vectors trained on 100 billion words from Google News using the continuous bag-of-words architecture (Mikolov et al., 2013b) to initialize word embeddings, but randomly initialize character embeddings. All embeddings have dimensionality of 300. We also randomly initialize word embeddings that are not present in the vocabulary of those pre-trained word vectors.

		Actual		
		Pos	Neu	Neg
Predicted	Pos	PP	PU	PN
	Neu	UP	UU	UN
	Neg	NP	NU	NN

Table 2: The confusion matrix for Subtask A. Cell XY stands for "the number of tweets that were labeled as X and should have been labeled as Y", where P U N stand for Positive Neutral Negative, respectively.

4 Results and Discussion

The same evaluation measure as the one used in previous years is adopted, i.e.,

$$F_1^{PN} = \frac{F_1^{Pos} + F_1^{Neg}}{2} \tag{3}$$

where F_1^{Pos} is defined as,

$$F_1^{Pos} = \frac{2\pi^{Pos}\rho^{Pos}}{\pi^{Pos} + \rho^{Pos}} \tag{4}$$

with ρ^{Pos} defined as the precision of predicted positive tweets, i.e., the fraction of tweets predicted to be positive that are indeed positive,

$$\rho^{Pos} = \frac{PP}{PP + PU + PN} \tag{5}$$

and π^{Pos} defined as the recall of predicted positive tweets, i.e., the fraction of positive tweets that are predicted to be such,

$$\pi^{Pos} = \frac{PP}{PP + UP + NP} \tag{6}$$

where PP, PU, PN, UP, NP are defined in Table 2, a confusion matrix for Subtask A provided by (Nakov et al.,). F_1^{Neg} is defined similarly as F_1^{Pos}.

Rank	System	2013		2014			2015	2016
		Tweet	SMS	Tweet	Tweet sacasm	Live-Journal	Tweet	Tweet
1	SwissCheese	0.700_5	0.637_2	0.716_5	0.566_1	0.695_7	0.671_1	0.633_1
2	SENSEI-LIF	0.706_4	0.634_3	0.744_2	0.467_8	0.741_1	0.662_2	0.630_2
3	unimelb	0.687_7	0.593_9	0.706_7	0.449_{11}	0.683_9	0.651_4	0.617_3
4	INESC-ID	0.723_2	0.609_6	0.727_3	0.554_3	0.702_4	0.657_3	0.610_4
5	aueb*	0.666_8	0.618_5	0.708_6	0.410_{17}	0.695_7	0.623_7	0.605_5
6	SentiSys	0.714_3	0.633_4	0.723_4	0.515_5	0.726_2	0.644_5	0.598_6
7	I2RNTU	0.693_6	0.597_7	0.680_8	0.469_6	0.696_6	0.638_6	0.596_7
8	INSIGHT-1	0.602_{16}	0.582_{12}	0.644_{16}	0.391_{23}	0.559_{23}	0.595_{16}	0.593_8
9	twise	0.610_{15}	0.540_{17}	0.645_{14}	0.450_{10}	0.649_{13}	0.621_8	0.586_9
10	ECNU	0.643_{10}	0.593_9	0.662_9	0.425_{14}	0.663_{10}	0.606_{11}	0.585_{10}
11	NTNUSentEval	0.623_{12}	0.641_1	0.651_{11}	0.427_{13}	0.719_3	0.599_{13}	0.583_{11}
12	MDSENT	0.589_{19}	0.509_{21}	0.587_{20}	0.386_{24}	0.606_{19}	0.593_{18}	$\mathbf{0.580_{12}}$
12	CUFE	0.642_{11}	0.596_8	0.662_9	0.466_9	0.697_5	0.598_{14}	0.580_{12}
14	THUIR	0.616_{13}	0.575_{14}	0.648_{12}	0.399_{20}	0.640_{15}	0.617_{10}	0.576_{14}
14	PUT	0.565_{21}	0.511_{20}	0.614_{19}	0.360_{27}	0.648_{14}	0.597_{15}	0.576_{14}
-	MDSENT*	0.664_9	0.610_6	0.676_9	0.410_{17}	0.689_9	0.628_7	$\mathbf{0.601_6}$
	baseline	0.292	0.190	0.346	0.277	0.272	0.303	0.255

Table 3: Evaluation Results of the top 15 systems with ranks provided as subscripts. aueb* stands for "aueb.twitter.sentiment". Our model with setting 1 ranks 12th among 34 systems. We also show the evaluation results and our reported ranks of MDSENT with setting 2 among the 34 systems in MDSENT*.

Runs	Setting 1		Setting 2	
	w	F_1^{PN}	w	F_1^{PN}
Run 1	0.66	0.582	1.00	0.603
Run 2	0.81	0.583	1.00	0.604
Run 3	0.60	0.587	0.98	0.607
Run 4	0.60	0.591	0.97	0.603
Run 5	0.60	0.592	0.95	0.601
Average	0.654	0.587	0.98	0.604

Table 4: Statistics of 5 individual runs for both settings.

We show the evaluation results of our system in Table 3, along with the top 15 systems reported. Originally we tested the system with only setting 1 and it ranks 12th among 34 systems. However, we find the system with setting 1 perform poorly on older datasets, which may due to the lack of training data. Thus we then test our model with setting 2 and report ranks generated from the same list of evaluation results reported by the 34 systems. It is apparent that our system can benefit from more training data and shows significant performance improvement (rank 6th).

Another interesting observation is that when provided with large amounts of training data, the CNN itself can perform very well, with LR assigned a very small weight during the combination procedure. We further test this finding by making 5 individual runs for both settings and checking the combination scalar weight w and final evaluation score F_1^{PN}. We list corresponding results in Table 4. With more training data, w increased from an average of 0.654 to an average of 0.98, which is very close to 1, while the performance improved from an average of 0.587 to an average of 0.604. This suggests the possibility to use only deep learning techniques along with embeddings to achieve similar or even better performance than traditional systems that require a lot of human engineered features and knowledge bases.

Our future work includes finer-design of the CNN, e.g., performing two stages of classification: first for subjectivity detection and then for polarity classification. We will also seek the possibility of conducting unsupervised learning with the CNN, which allows us to make use of the large amount of tweets on the Internet. With such increased amount of training data, our system may further improve its performance.

References

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011.

Natural language processing (almost) from scratch. *The Journal of Machine Learning Research*, 12:2493–2537.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

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*.

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*, pages 168–177. ACM.

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

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Saif M Mohammad and Peter D Turney. 2010. Emotions evoked by common words and phrases: Using mechanical turk to create an emotion lexicon. In *Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text*, pages 26–34. Association for Computational Linguistics.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. Evaluation measures for the semeval-2016 task 4 'sentiment analysis in twitter'(draft: Version 1.1).

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pages 801–809.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.