CUFE at SemEval-2016 Task 4: A Gated Recurrent Model for Sentiment Classification

CUFE at SemEval-2016 Task 4: A Gated Recurrent Model for Sentiment Classification

Mahmoud Nabil[1], Mohamed Aly[2] and Amir F. Atiya[3]

[1,2,3]Computer Engineering, Cairo University, Egypt
[2]Visual Computing Center, KAUST, KSA
[1]*mah.nabil@cu.edu.eg*
[2]*mohamed@mohamedaly.info*
[3]*amir@alumni.caltech.edu*

Abstract

In this paper we describe a deep learning system that has been built for SemEval 2016 Task4 (Subtask A and B). In this work we trained a Gated Recurrent Unit (GRU) neural network model on top of two sets of word embeddings: (a) general word embeddings generated from unsupervised neural language model; and (b) task specific word embeddings generated from supervised neural language model that was trained to classify tweets into positive and negative categories. We also added a method for analyzing and splitting multi-words hashtags and appending them to the tweet body before feeding it to our model. Our models achieved 0.58 F1-measure for Subtask A (ranked 12/34) and 0.679 Recall for Subtask B (ranked 12/19).

1 Introduction

Twitter is a huge microbloging service with more than 500 million tweets per day[1] from different locations in the world and in different languages. This large, continuous, and dynamically updated content is considered a valuable resource for researchers. However many issues should be taken into account while dealing with tweets, namely: (1) informal language used by the users; (2) spelling errors; (3) text in the tweet may be referring to images, videos, or external URLs; (4) emoticons; (5) hashtags used (combining more than one word as a single word); (6) usernames used to call or notify other users; (7) spam or irrelevant tweets; and (8) character limit for a tweet to 140 characters. This poses many challenges when analyzing tweets for natural language processing tasks. In this paper we describe our system used for SemEval 2016 (Nakov et al., 2016b) Subtasks A and B. Subtask A (Message Polarity Classification) requires classifying a tweet's sentiment as positive; negative; or neutral,. Subtask B (Tweet classification according to a two-point scale) requires classifying a tweet's sentiment given a topic as positive or negative. Our system uses a GRU neural network model (Bahdanau et al., 2014) with one hidden layer on top of two sets of word embeddings that are slightly fine-tuned on each training set (see Fig. 1). The first set of word embeddings is considered as general purpose embeddings and was obtained by training *word2vec* (Mikolov et al., 2013) on 20.5 million tweets that we crawled for this purpose. The second set of word embeddings is considered as task specific set, and was obtained by training on a supervised sentiment analysis dataset using another GRU model. We also added a method for analyzing multi-words hashtags by splitting them and appending them to the body of the tweet before feeding it to the GRU model. In our experiments we tried both keeping the word embeddings static during the training or fine-tuning them and reported the result for each experiment. We achieved 0.58 F1-measure for Subtask A (ranked 12/34) and 0.679 Recall for Subtask B (ranked 12/19).

2 Related Work

A considerable amount of research has been done to address the problem of sentiment analysis for

[1]`http://internetlivestats.com/twitter-statistics/`

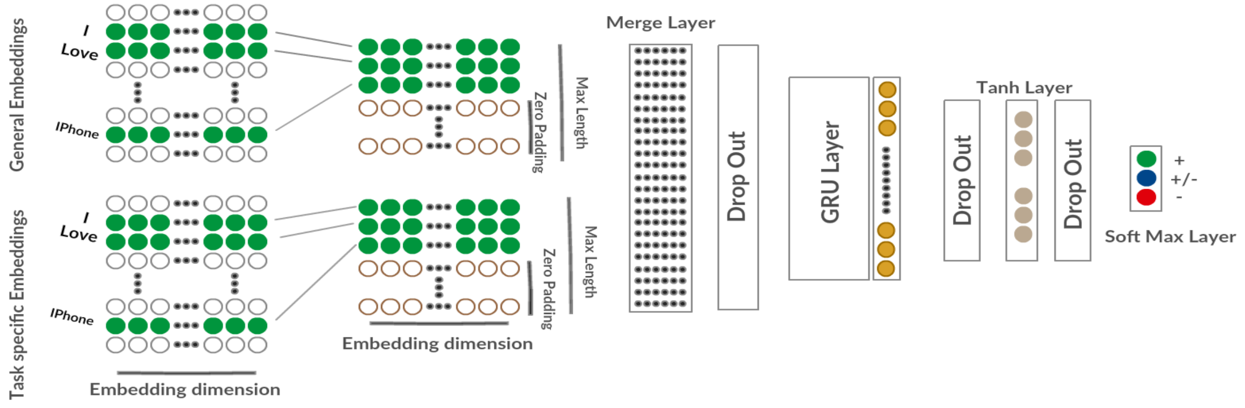

Figure 1: The architecture of the GRU deep Learning model

social content. Nevertheless, most of the state-of-the-art systems still extensively depends on feature engineering, hand coded features, and linguistic resources. Recently, deep learning model gained much attention in sentence text classification inspired from computer vision and speech recognition tasks. Indeed, two of the top four performing systems from SemEval 2015 used deep learning models. (Severyn and Moschitti, 2015) used a Convolution Neural Network (CNN) on top of skip-gram model word embeddings trained on 50 million unsupervised tweets. In (Astudillo et al., 2015) the author built a model that uses skip-gram word embeddings trained on 52 million unsupervised tweets then they project these embeddings into a small subspace, finally they used a non-linear model that maps the embedding subspace to the classification space. In (Kim, 2014) the author presented a series of CNN experiments for sentence classification where static and fine-tuned word embeddings were used. Also the author proposed an architecture modification that allow the use of both task-specific and static vectors. In (Lai et al., 2015) the author proposed a recurrent convolutional neural network for text classification. Finally regarding feature engineering methods, (Büchner and Stein, 2015) the top performing team in SemEval 2015, used an ensemble learning approach that averages the confidence scores of four classifiers. The model uses a large set of linguistic resources and hand coded features.

3 System Description

Fig 1 shows the architecture of our deep learning model. The core of our network is a GRU layer, which we chose because (1) it is more computational efficient than Convolutional Neural Network (CNN) models (Lai et al., 2015) that we experimented with but were much slower; (2) it can capture long semantic patterns without tuning the model parameter, unlike CNN models where the model depends on the length of the convolutional feature maps for capturing long patterns; (3) it achieved superior performance to CNNs in our experiments.

Our network architecture is composed of a word embeddings layer, a merge layer, dropout layers, a GRU layer, a hyperbolic tangent *tanh* layer, and a soft-max classification layer. In the following we give a brief description of the main components of the architecture.

3.1 Embedding Layer

This is the first layer in the network where each tweet is treated as a sequence of words $w_1, w_2...w_S$ of length S, where S is the maximum tweet length. We set S to 40 as the length of any tweet is limited to 140 character. We used zero padding while dealing with short tweets. Each word w_i is represented by two embedding vectors $w_{i_1}, w_{i_2} \in R^d$ where d is the embedding dimension, and according to (Astudillo et al., 2015) setting d to 200 is a good choice with respect to the performance and the computation efficiency. w_{i_1} is considered a general-purpose embedding vector while w_{i_2} is considered a task-specific embedding vector. We performed the following steps to initialize both types of word embeddings:

1. For the general word embeddings we collected about 40M tweets using twitter streaming API over a period of two month (Dec. 2015 and Jan. 2016). We used three criteria while collecting the tweets: (a) they contain at least one emoticon in a set of happy and sad emoticons like ':)' ,':(', ':D' ... etc. (Go et al., 2009); (b) hash tags collected from SemEval 2016 data set; (c) hash tags collected from SemEval 2013 data set. After preparing the tweets as described in Section 4 and removing retweets we ended up with about 19 million tweet. We also appended 1.5 million tweets from *Sentiment140* (Go et al., 2009) corpus after preparation so we end up with about 20.5 million tweet. To train the general embeddings we used *word2vec* (Mikolov et al., 2013) neural language model skipgram model with window size 5, negative sampling and filtered out words with frequency less than 5.

2. For the task specific word embeddings we used semi-supervised 1.5 million tweets from *sentiment140* corpus, where each tweet is tagged either positive or negative according to the tweet's sentiment . Then we applied another GRU model similar to Fig 1 with a modification to the soft-max layer for the purpose of the two classes classification and with random initialized embeddings that are fine-tuned during the training. We used the resulting fine-tuned embeddings as task-specific since they contain contextual semantic meaning from the training process.

3.2 Merge Layer

The purpose of this layer is to concatenate the two types of word embeddings used in the previous layer in order to form a sequence of length $2S$ that can be used in the following GRU layer.

3.3 Dropout Layers

The purpose of this layer is to prevent the previous layer from overfitting (Srivastava et al., 2014) where some units are randomly dropped during training so the regularization of these units is improved.

3.4 GRU Layer

This is the core layer in our model which takes an input sequence of length $2S$ words each having dimension d (i.e. input dimension is $2Sd$) . The gated recurrent network proposed in (Bahdanau et al., 2014) is a recurrent neural network (a neural network with feedback connection, see (Atiya and Parlos, 2000)) where the activation h_t^j of the neural unit j at time t is a linear interpolation between the previous activation h_{t-1}^j at time $t-1$ and the candidate activation $\tilde{h}_t^j$ (Chung et al., 2014):

$$h_t^j = (1 - z_t^j) h_{t\text{-}1}^j + z_t^j \tilde{h}_t^j$$

where z_t^j is the *update gate* that determines how much the unit updates its content, and $\tilde{h}_t^j$ is the newly computed candidate state.

3.5 Tanh Layer

The purpose of this layer is to allow the neural network to make complex decisions by learning non-linear classification boundaries. Although the *tanh* function takes more training time than the Rectified Linear Units (ReLU), *tanh* gives more accurate results in our experiments.

3.6 Soft-Max Layer

This is last layer in our network where the output of the tanh layer is fed to a fully connected soft-max layer. This layer calculates the classes probability distribution.

$$P(y = c \,|\, x, b) = \frac{\exp\left(w_c^T x + b_c\right)}{\sum_{k=1}^{K} \exp\left(w_k^T x + b_k\right)}$$

where c is the target class, x is the output from the previous layer, w_k and b_k are the weight and the bias of class k, and K is the total number of classes. The difference between the architecture used for Subtask A and Subtask B is in this layer, where for Subtask A three neurons were used (i.e. $K = 3$) while for Subtask B only two neurons were used (i.e. $K = 2$).

4 Data Preparation

All the data used either for training the word embeddings or for training the sentiment classification model undergoes the following preprocessing steps:

Pattern	Examples	Normalization
Usernames	@user1,@user2	_UserName_
Happy emotions	:), :-), :=)	:)
Sad emotions	:(, :-(, :=(	:(
Laugh emotions	:D, :-D, :=D	:D
Kiss emotions	:-*, :*, :-)*	_KISS_
Surprise emotions	:O, :-o	:o
Tongue emotions	:P, :p	:p
Numbers	123	_NUM_
URLs	www.google.com	_URL_
Topic (Subtask B only)	Microsoft	_Entity_

Table 1: Normalization Patterns

1. Using NLTK twitter tokenizer[2] to tokenize each tweet.

2. Using hand-coded tokenization regex to split the following suffixes: 's, 've, 't , 're, 'd, 'll.

3. Using the patterns described in Table 1 to normalize each tweet.

4. Adding `_StartToken_` and `_EndToken_` at the beginning and the ending of each tweet.

5. Splitting multi-word hashtags as explained below.

Consider the following tweet *"Thinking of reverting back to 8.1 or 7. #Windows10Fail*". The sentiment of the tweet is clearly negative and the simplest way to give the correct tag is by looking at the word "Fail" in the hashtag "#Windows10Fail". For this reason we added a depth first search dictionary method in order to infer the location of spaces inside each hashtag in the tweet and append the result tokens to the tweet's end. We used 125k words dictionary[3] collected from Wikipedia. In the given example, we first lower the hashtag case, remove numbers and underscores from the hashtag then we apply our method to split the hashtag this results in two tokens "windows" and "fail". Hence, we append these two tokens to the end of the tweet and the normal preparation steps continue. After the preparation the tweet will look like "*_StartToken_ Thinking of reverting back to _NUM_ or _NUM_. #Windows10Fail. windows fail _EndToken_*".

[2]http://nltk.org/api/nltk.tokenize.html

[3]http://pasted.co/c1666a6b

Dataset	all	pos.	neg.	neut.
train-A	12886	5651	1967	5268
dev-A	3222	1395	462	1365
train-B	6324	5059	1265	-
dev-B	1265	1059	206	-

Table 2: Tweets distribution for Subtask A and B

Dataset	Subtask A	Subtask B
GRU-static	0.635	0.826
GRU-fine-tuned	0.639	0.829
GRU-fine-tuned + Split Hashtag	**0.642**	**0.830**

Table 3: Development results for Subtask A and B. **Note**: average F1-mesure for positive and negative classes is used for Subtask A, while the average recall is used for Subtask B.

5 Experiments

In order to train and test our model for Subtask A, we used the dataset provided for SemEval-2016 Task 4 and SemEval-2013 Task 2. We obtained 8,978 from the first dataset and 7,130 from the second, the remaining tweets were not available. So, we ended up with a dataset of 16,108 tweets. Regarding Subtask B we obtained 6,324 from SemEval-2016 provided dataset. We partitioned both datasets into train and development portions of ratio 8:2. Table 2 shows the distribution of tweets for both Subtasks.

For optimizing our network weights we used Adam (Kingma and Ba, 2014), a new and computationally efficient stochastic optimization method. All the experiments have been developed using Keras[4] deep learning library with Theano[5] backend and with CUDA enabled. The model was trained using the default parameters for Adam optimizer, and we tried either to keep the weights of embedding layer static or slightly fine-tune them by using a dropout probability equal to 0.9. Table 3 shows our results on the development part of the data set for Subtask A and B where we report the official performance measure for both subtasks (Nakov et al., 2016a). From 3 the results it is shown that fine-tuning word embeddings with hashtags splitting gives the best results on the development set. All our experiments were performed on a machine with Intel Core i7-4770 CPU @ 3.40GHz (8 cores), 16GB

[4]http://keras.io/

[5]http://deeplearning.net/software/theano/

Dataset	Baseline	F-measure (Old)	F-measure (New)
Tweet-2013	0.292	0.642	0.665
SMS-2013	0.190	0.596	0.665
Tweet-2014	0.346	0.662	0.676
Tweet-sarcasm	0.277	0.466	0.477
Live-Journal	0.272	0.697	0.631
Tweet-2015	0.303	0.598	0.624
Tweet-2016	0.255	0.580	0.608

Table 4: Results for Subtask A on different SemEval datasets.

Dataset	Baseline	Recall (Old)	Recall (New)
Tweet-2016	0.389	0.679	0.767

Table 5: Result for Subtask B on SemEval 2016 dataset.

of RAM and GeForce GT 640 GPU. Table 4 shows our individual results on different SemEval datasets. Table 5 shows our results for Subtask B. From the results and our rank in both Subtasks, we noticed that our system was not satisfactory compared to other teams this was due to the following reasons:

1. We used the development set to validate our model in order to find the best learning parameters, However we mistakenly used the learning accuracy to find the optimal learning parameters especially the number of the training epochs. This significantly affected our rank based on the official performance measure. Table 4 and Table 5 show the old and the new results after fixing this bug.

2. Most of the participating teams in this year competition used deep learning models and they used huge datasets (more than 50M tweets) to train and refine word embeddings according to the emotions of the tweet. However, we only used 1.5M from *sentiment140* corpus to generate task-specific embeddings.

3. The model used for generating the task-specific embeddings for Subtask A should be trained on three classes not only two (positive, negative, and neutral) where if the tweet contains positive emotions like ":)" should be positive, if it contains negative emotions like ":(" should be negative, and if it contains both or none it should be neutral.

6 Conclusion

In this paper, we presented our deep learning model used for SemEval2016 Task4 (Subtasks A and B). The model uses a gated recurrent layer as a core layer on top of two types of word embeddings (general-purpose and task-specific). Also we described our steps in generating both types word embeddings and how we prepared the dataset used especially when dealing with multi-words hashtags. The system ranked 12th on Subtask A and 12th for Subtask B.

Acknowledgments

This work has been funded by ITIDA's ITAC project number CFP65.

References

Ramon F Astudillo, Silvio Amir, Wang Ling, Bruno Martins, Mário Silva, Isabel Trancoso, and Rua Alves Redol. 2015. Inesc-id: Sentiment analysis without hand-coded features or liguistic resources using embedding subspaces. *SemEval-2015*, page 652.

Amir F Atiya and Alexander G Parlos. 2000. New results on recurrent network training: unifying the algorithms and accelerating convergence. *Neural Networks, IEEE Transactions on*, 11(3):697–709.

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

Matthias Hagen Martin Potthast Michel Büchner and Benno Stein. 2015. Webis: An ensemble for twitter sentiment detection. *SemEval-2015*, page 582.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1:12.

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

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

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *AAAI*, pages 2267–2273.

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

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

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016b. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, California, June. 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), Association for Computational Linguistics, Denver, Colorado*, pages 464–469.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.