RIDDL at SemEval-2018 Task 1: Rage Intensity Detection with Deep Learning

Venkatesh Elango, Karan Uppal
Bloomberg
New York, NY, USA
{velango,kuppal8}@bloomberg.net

Abstract

We present our methods and results for affect analysis in Twitter developed as a part of SemEval-2018 Task 1, where the sub-tasks involve predicting the intensity of emotion, the intensity of sentiment, and valence for tweets. For modeling, though we use a traditional LSTM network, we combine our model with several state-of-the-art techniques to improve its performance in a low-resource setting. For example, we use an encoder-decoder network to initialize the LSTM weights. Without any task specific optimization we achieve competitive results (macro-average Pearson correlation coefficient 0.696) in the El-reg task. In this paper, we describe our development strategy in detail along with an exposition of our results.

1 Introduction

Sentiment analysis is a technique to classify documents based on the polarity of opinion expressed by the author of the document (Pang et al., 2002). Traditionally this involved extracting coarse sentiment (positive, negative, or neutral) from documents such as news articles, product or movie reviews (Wiebe et al., 2005; Hu and Liu, 2004). In order to get a fine grained view of the opinion, sentiment analysis was applied at the sentence and phrase level (Yu and Hatzivassiloglou, 2003; Wilson et al., 2005). With the advent of social media, Twitter in particular, sentiment towards a wide range of topics could be extracted at a much larger scale than before. This however came with its own set of problems, viz., a lack of proper grammatical structure, prevalence of slang, acronyms, and misspellings (Jansen et al., 2009; Barbosa and Feng, 2010).

SemEval tasks have provided a curated testing environment for analysis on Twitter data with tasks to quantify sentiment on two-point, three-point, and five-point scales (Nakov et al., 2016; Rosenthal et al., 2017). While a finer gradation in polarity of the text could be inferred by introducing more nuanced categories, it faces the problem of needing to collect more labeled data as the number of classes increase. Therefore, this approach requires that the intensity of the sentiment expressed be measured on a continuous scale rather than through discrete categories. SemEval 2018 Task 1 takes a novel step in this direction by introducing tasks for predicting intensity of emotion, or sentiment expressed (Mohammad et al., 2018).

We participated in SemEval-2018 Task-1 (El-Reg, V-reg, V-oc) and our contributions through this paper are as follows:

- We present an LSTM network combined with known state-of-the-art techniques to improve performance on low-resource setting tasks.
- The proposed model requires no task specific hyper-parameter tuning.
- We perform error analysis of our model to obtain a better understanding of strengths and weaknesses of a deep learning-based approach for these tasks and propose improvements.

2 Methods

2.1 Datasets

For each subtask, the organizers provide training and development datasets for model training and hyperparameter selection. The details on how much data and how it was labeled can be found here (Mohammad and Kiritchenko, 2018). We concatenate the training and development datasets and sample 10% of this combined dataset, to use as validation data. Our model training involves an

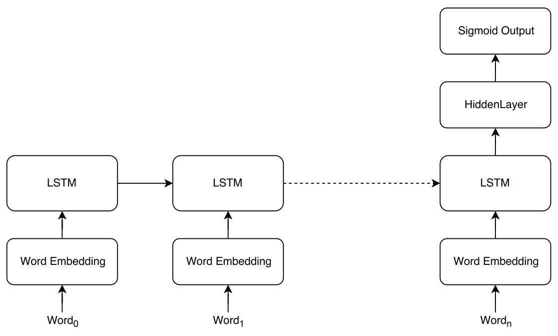


Figure 1: Network architecture.

unsupervised phase and a supervised phase, which is described in detail in Section 2.3. For the unsupervised learning phase, we use the concatenated training data from all the tasks, and for the supervised learning phase, we use the task specific training data.

2.2 Model

We start by pre-processing and tokenizing the tweets by adapting the pre-processing used in training GloVe word embeddings for Twitter (Pennington et al., 2014). Following the pre-processing techniques used in GloVe, we retain punctuations, normalize mentions, numbers, URLs, smileys (happy, neutral, and sad separately), and include tags for hashtags, repeating, and all upper case characters. In addition, we also pre-process emojis by replacing them with their Unicode text description[1].

After a tweet is broken down into a sequence of tokens, it is then converted to a sequence of vectors using the 200-dimensional GloVe word embeddings for Twitter which was trained on 2 billion tweets with a vocabulary of size 1.2 million (Pennington et al., 2014).

This sequence of word vectors is next input to an LSTM network (Hochreiter and Schmidhuber, 1997), with h_1 hidden units. The output from the last time step of the LSTM cell is passed through a layer of h_2 hidden units with ReLU activation. A final sigmoid layer then produces the output. Since all of the tasks, including the ordinal classification task, have an innate sense of ordering, we cast them as regression problems. The network architecture is shown in Figure 1.

[1]http://unicode.org/emoji/charts/emoji-list.html

2.3 Training

We tried two different training strategies. The results submitted before the official deadline used the first strategy. Since then, we identified a few key areas of improvement and used the second training strategy to get much better results. Both strategies are described below and the results are discussed in Section 3.

2.3.1 Strategy 1

We divide the training of our model into two phases, an unsupervised phase and a supervised phase.

Unsupervised Phase

Since the amount of training data is small (approximately 2000 labeled samples on average across all tasks) in comparison to the number of parameters (approximately 500,000) of the model, the training of the model could be unstable and prone to over-fitting. To counter this problem, the weights of the LSTM are initialized using a modified sequence auto-encoder (Dai and Le, 2015). This modified sequence auto-encoder uses separate encoder and decoder networks, attempting to reconstruct the input to the encoder at the output of the decoder by minimizing the mean squared error between them (Elango et al., 2017; Srivastava et al., 2015). For this unsupervised learning phase, we pool the training data from all the tasks and use 10% of it as a validation set. The validation set is used to tune the number of epochs. The validation loss is minimized for 5 epochs. Then, we fine-tune these weights with task specific training + validation data and this fine-tuning is run for another 10 epochs.

The unsupervised learning procedure is crucial for the good initialization of weights in the supervised task. As the model optimizes its weights for reconstruction of the sequence, it is able to learn the structure of the data.

Supervised Phase

For the supervised phase, the weights from the encoder network are used as initialization. To learn a generalizable model, instead of optimizing the hyper-parameters for each task separately, we optimize the hyper-parameters only for the anger intensity regression sub-task (EI-reg anger) which was picked arbitrarily. For tuning the hyper-parameters we combine the training and development set provided for the task and randomly sam-

Hyperparameter	Value
h_1	256
h_2	32
n_{mb}	32
e	5
p_{do}	0.5

Table 1: Hyperparameters used for network training.

ple 10% of the data from this task as a validation set. The set of hyper-parameters optimized are the number of hidden units (h_1, h_2), mini-batch size (n_{mb}), number of epochs to train the network (e) and dropout probability (p_{do}). For all the other tasks, the same optimized set of hyper-parameter values are used in training the model with all of the task specific training data.

Also, per recommendation from (Gers et al., 2000; Jozefowicz et al., 2015), to enable gradient flow, the bias term in the forget gate of the LSTM is initialized to 1. We apply dropout to the recurrent states (Gal and Ghahramani, 2016) and the hidden nodes with a probability of p_{do}, to prevent over-fitting to the training data. The model optimization is carried out by back-propagation using Adam optimizer (Kingma and Ba, 2014).

Using the above approach, the set of hyper-parameters that provide the best performance are reported in Table 1.

2.3.2 Strategy 2

After observing the results for all the subtasks we noticed that variance is fairly high in the predicted intensities. This is also visible in the scatter plots of true and predicted intensity, shown in Figures 2 and 3. We also noticed during the hyper parameter optimization on the anger task that the gap between the validation loss and training loss was high.

Simpler Model and Regularization

The above observations led us to believe that we might be over-fitting to the training data. Hence we tried the following steps to reduce over-fitting:

1. Reduced number of parameters: Number of hidden units of LSTM was reduced to 100.

2. Increased dropout rate: Increased the dropout in the hidden layer to 0.75.

Additional Unlabeled Data

Furthermore, we believed that the unsupervised phase of our training could benefit from more unlabeled data. So we pooled in all the development and test data across all the tasks along with the training data, and used it in the unsupervised learning of weights of the LSTM. To tune the number of epochs, we set aside 10% of the combined training, development, and test as validation set. It is important to note that at no point were gold labels of test data used in any phase of training. This was simply an inexpensive way to get more data for trainining the unsupervised phase.

Ensemble

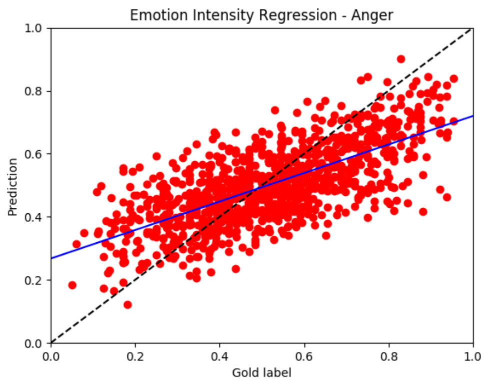
To further improve the prediction performance, we take an ensemble of 5 versions of our model and also optimize the number of epochs of training for each task for each of the 5 models. All the other hyper-parameters are kept the same across all tasks. For each of the 5 models, a random validation set with 10% of the labeled data is set aside to tune the number of epochs. Therefore, each model of the ensemble is being trained 90% of randomly sampled data. The supervised phase is trained, with random initialization for the dense layer, for 15 epochs, and the model state is saved at the end of every epoch. Finally the model corresponding to the epoch with the lowest validation loss is picked. All 5 models are used to predict on the test set, and the average of the 5 predictions is used as the final prediction of the ensemble.

3 Results

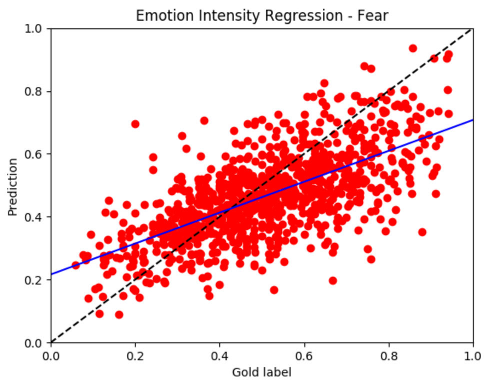
Models trained with both strategies are evaluated using Pearson correlation as the metric. We compare our performance with an unigram SVM baseline, as well as the best submission for each subtask. The official submitted results using Strategy 1 are reported in Tables 2 and 3 for subtask EI-reg and in Tables 4 and 5 for subtasks V-reg and V-oc respectively. The improved results using strategy 2 are reported in Table 6. The results in table 6 are averaged over 5 runs and report the standard deviation over the runs as well. We find that strategy 2 improves the macro-average Pearson correlation from 0.666 to 0.696.

To further analyze the results for the regression tasks, we created scatter plots shown in Figures 2 and 3. We plot the predicted intensity score against the gold label intensity score and also show the line of best fit.

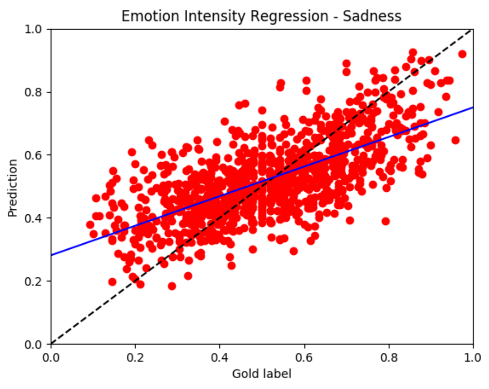
Analyzing the scatter plots, we note that our model consistently overestimates the intensity



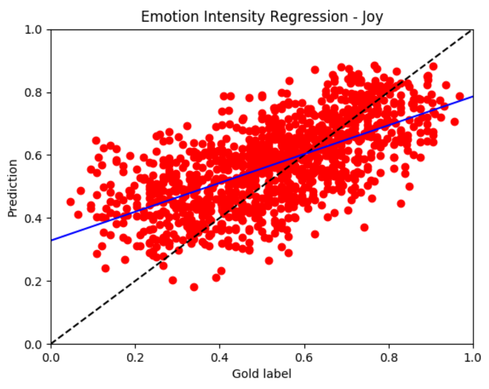
a



b



c



d

Figure 2: Plot of predicted against gold intensity score for emotion intensity regression using Strategy 1.

	Emotion			
Model	Anger	Fear	Joy	Sadness
Baseline	0.526	0.525	0.575	0.453
Ours	0.695	0.659	0.638	0.672
Best	0.827	0.779	0.792	0.798

Table 2: Pearson correlation on emotion intensity regression task (EI-reg) in English for each emotion using Strategy 1.

Model	Macro-average
Baseline	0.520
Ours	0.666
Best	0.799

Table 3: Macro average of Pearson correlation on emotion intensity regression task (EI-reg) in English using Strategy 1.

Model	Valence
Baseline	0.585
Ours	0.782
Best	0.873

Table 4: Pearson correlation on valence intensity regression task (V-reg) in English using Strategy 1.

Model	Valence
Baseline	0.509
Ours	0.593
Best	0.836

Table 5: Pearson correlation on valence ordinal classification task (V-oc) in English using Strategy 1.

	Pearson Correlation	
Emotion	Average	Std Dev
Anger	0.717	0.0021
Fear	0.695	0.0020
Joy	0.688	0.0054
Sadness	0.685	0.0020
Macro-average	0.696	0.0054

Table 6: Pearson correlation (average and standard deviation of 5 runs) on emotion intensity regression task (EI-reg) in English using Strategy 2.

when the gold label score is low and underestimates the intensity when the gold label score is high. The overestimation of low gold label score is most pronounced in the case of emotion intensity regression for *joy* and this is reflected in its low Pearson correlation in Table 2. Emotion intensity regression for *fear* has larger variance in predicted intensities for high gold label scores and

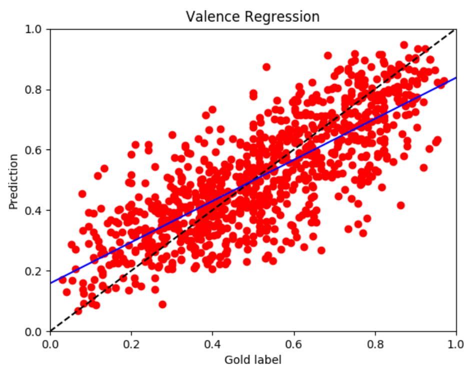


Figure 3: Plot of predicted against gold intensity score for valence regression using Strategy 1.

this too can be seen in the relatively low Pearson correlation reported in Table 2. While *anger* and *sadness* emotions also make under estimation error for high gold label scores, the line of best fit as well as the Pearson correlation are better than the corresponding ones for *joy* and *fear*.

The performance of the model in the valence regression task is markedly better in comparison to the performance in the emotion intensity regression tasks, as seen from the Pearson correlation, in Table 3, as well as the line of best fit, in Figure 3.

4 Conclusion

We presented an LSTM based approach for affect and emotion intensity regression and described our training strategy which did not involve any task specific hyper-parameter optimization. We did not employ any task specific hyper-parameter optimization to demonstrate that the training procedure is robust and that the model can be trained to achieve reasonable performance without being highly sensitive to values of hyper-parameters. We also show how the traditional LSTM network can be combined with known state-of-the-art techniques to get improvements in low resource settings. We use an encoder-decoder network to initialize the LSTM weights and use ensembles of our network to further improve performance. On the other hand, when the goal is to maximize performance, task specific hyper-parameter optimization could be employed, which is shown in strategy 2 where tuning the number of epochs on a per task basis helps the performance.

References

Luciano Barbosa and Junlan Feng. 2010. Robust sentiment detection on twitter from biased and noisy data. In *Proceedings of the 23rd international conference on computational linguistics: posters*, pages 36–44. Association for Computational Linguistics.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*, pages 3079–3087.

Venkatesh Elango, Aashish N Patel, Kai J Miller, and Vikash Gilja. 2017. Sequence transfer learning for neural decoding. *bioRxiv*, page 210732.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1019–1027.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10):2451–2471.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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.

Bernard J Jansen, Mimi Zhang, Kate Sobel, and Abdur Chowdury. 2009. Twitter power: Tweets as electronic word of mouth. *Journal of the Association for Information Science and Technology*, 60(11):2169–2188.

Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. 2015. An empirical exploration of recurrent network architectures. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2342–2350.

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

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 and Svetlana Kiritchenko. 2018. Understanding emotions: A dataset of tweets to study interactions between affect categories. In *Proceedings of the 11th Edition of the Language Resources and Evaluation Conference*, Miyazaki, Japan.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 task 4: Sentiment analysis in twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1–18.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

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-2017)*, pages 502–518.

Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*, pages 843–852.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.

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.

Hong Yu and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136. Association for Computational Linguistics.