uOttawa at SemEval-2018 Task 1: Self-Attentive Hybrid GRU-Based Network

Ahmed Husseini Orabi[1], Mahmoud Husseini Orabi[1], Diana Inkpen[1], David Van Bruwaene[2]
[1]EECS, University of Ottawa, 800 King Edward Avenue, Ottawa, Canada
[2]VISR inc., 10 Dundas St E. Suite 600, Toronto, Canada
{ahuss045, mhuss092, diana.inkpen}@uottawa.ca, d@visr.co

Abstract

We propose a novel attentive hybrid GRU-based network (SAHGN), which we used at SemEval-2018 Task 1: Affect in Tweets. Our network has two main characteristics, 1) has the ability to internally optimize its feature representation using attention mechanisms, and 2) provides a hybrid representation using a character-level Convolutional Neural Network (CNN), as well as a self-attentive word-level encoder. The key advantage of our model is its ability to signify the relevant and important information that enables self-optimization. Results are reported on the valence intensity regression task.

1 Introduction

Affect analysis is one of the main topics of natural language processing (NLP). It involves many sub-tasks such as sentiment and valence analyses expressed in text. We focus on the task of determining valence intensity.

Hand-crafted features and/or sentiment lexicons are commonly used for affect analysis (Mohammad, Kiritchenko, & Zhu, 2013; Taboada, Brooke, Tofiloski, Voll, & Stede, 2011) with classifiers such as random forest and support vector machines (SVM).

Affect in tweets (AIT) is a challenging task as it requires handling an informal writing style, which typically has many grammar mistakes, slangs, and misspellings.

In this paper, we present a self-attentive hybrid GRU-based network (SAHGN) that competed at SemEval-2018 Task 1 (Mohammad, Bravo-Marquez, Salameh, & Kiritchenko, 2018; Mohammad & Kiritchenko, 2018).

Our contributions can be summarized as below.

- **The implementation of a social media text processor:** A library to help process social media text such as short-forms, emoticons, emojis, misspellings, hash tags, and slangs, as well as tokenization, word normalization, and sentence encoding.
- **The implementation of a self-attentive deep learning system:** This system can predict valence and intensity with limited corpora and vocabulary, and yet can have acceptable performance.

2 High-Level Description of Our System

Our goal is to provide a system that can predict valence and intensity for short text. Figure 1 shows a high-level description of our solution, which consists of two main components, social media text processor (Section 3) and self-attentive hybrid GRU-based network (Section 4.2).

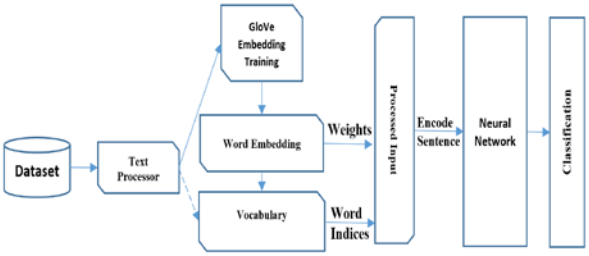

Figure 1: System architecture.

3 Social Media Text Processor

The social media text processor aims to provide a reliable and fast tokenization. It involves the following preprocessing steps:

- Use a named entity recognizer (NER) (Finkel, Grenager, & Manning, 2005) to identify entities such as persons, names, and places, and then replace them accordingly.
- Build a vocabulary using an NGram tokenizer.

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 181–185
New Orleans, Louisiana, June 5–6, 2018.

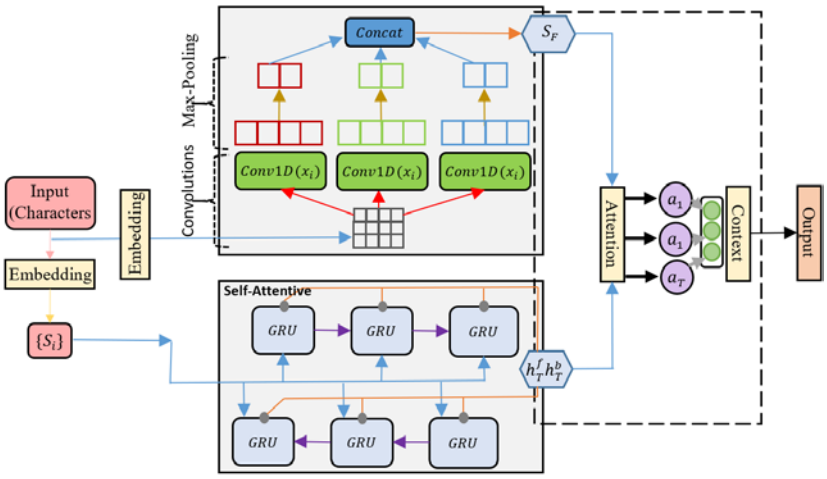

Figure 2: The architecture of Self-Attentive Hybrid GRU-Based Network.

- Tokenize sentences into a set of tokens, and then use them to encode text into a sequence of indices (Table 1), which are fed into the network.
- Clean text from accents, punctuations, and non-Latin characters.
- Identify emoticons and emojis, and then replace them with meaningful text; e.g., replace the happy face emoticon :) with <happy>.
- Recognize hashtags, URLs, and then briefly describe them; e.g. replace #depressed by <hashtag_start>depressed<hashtag_end>.
- Identify user reference mentions, and then replace them with a person entity; e.g. <person>.

Text	@name I am feeling under the weather after I met with Carl :'(#sick \\u0001F600
Processed	<SOS> <reference> I am feeling under the weather after I met with <person> <crying> <hashtag_start> sick <hashtag_end> <grinning_face> <EOS>

Table 1: Example of processed text.

4 Model Description

The overall architecture of our SAHGN model is shown in Figure 2. The main components include 1) a word sequence encoder, 2) a bidirectional GRU-based layer that applies a self-attentive mechanism on the word level, 3) a character-level CNN feature extractor, and 4) an attention with context-aware mechanism.

4.1 Word Sequence Encoder

A network input is described as a sequence (S) of tokens (such as words), where $S = [s_1, s_2, \ldots, s_t]$ and t denotes the timestep. S_i is a one-hot input (i) vector of a fixed length (T) of tokens. A sequence that exceeds this length is truncated.

Word encoding. We use a W word vocabulary to encode a sequence. W has fixed terms to determine the start and end of the sequence, as well as the out of vocabulary (OOV) words. We handle the variable length through padding for short sequences and truncating for long sequences.

Embedding layer. We apply a pretrained GloVe word embedding (Pennington, Socher, & Manning, 2014) on S_i. GloVe projects these words into a low-dimensional vector representation (x_i), where $x_i \in R^W$ and W is the word weight embedding matrix. W is used to initialize the word embedding layer.

We used the official training and development corpora to train the GloVe word embedding with a dimension of 100. The vocabulary size of this model is *8145* words, which is small and poses a major challenge to training, as well as to performance.

4.2 Self-attentive GRU-based Mechanism

Recurrent neural network (RNN) is commonly used for NLP problems (Yin, Kann, Yu, & Schütze, 2017; Young, Hazarika, Poria, & Cambria, 2017), as it enables remembering values over arbitrary time durations. RNN processes every element of an input embedding (x_i) sequentially, such that $h_t = \tanh(W_{x_i} + W_{h_{t-1}})$. W is the weight matrix between an input and hidden states, while h_t is the hidden state of the recurrent connection at timestep (t). The design of the RNN enables variable length processing while preserving the sequence order.

However, RNN has many limitations with long sequences, in particular the exponentially growing or decaying gradients. A common way to resolve these issues is by using gating mechanisms, such as LSTM and GRU (Gers, Schmidhuber, & Cummins, 2000; Hochreiter & Schmidhuber, 1997). We use GRU as it is faster to converge, in addition to being memory efficient.

Bidirectional GRU layer. In our model, we use bidirectional GRU layers. GRU receives a sequence of tokens as inputs, and then projects word information $H = (h_1, h_2, \dots, h_T)$, where h_t denotes the hidden state of GRU at a timestep (t). It captures the temporal and abstract information of sequences in a forward (h^f) or reverse (h^b) manner. After that, we concatenate forward and backward representations; e.g. $h_t = h_t^f || h_t^b$.

Attention mechanism. Words do not have equal valence weights in sentences. Towards that, we use an attention mechanism to signify the relatively important words.

Attention is used to compute the compatibility between a given source (x_i) and query (q). It uses an alignment function $f(x_i, q)$ to measure the level of dependency of q to x_i. This function produces an attention weight $a = f(x_i, q)_{i=1}^{T}$. Then, a softmax function is applied to produce a probability distribution $p(z|x, q)$ for each word (t) of an input (x). Hence, a bigger weight of x_i indicates a higher importance than other words.

The attention alignment approaches have the same implementation, but they mainly differ on how they compute weights. This can be either in an additive manner $f(x_i, q) = \tanh(W^T B(W_{x_i} + W_q))$ (Bahdanau, Cho, & Bengio, 2014), or a multiplicative manner $f(x_i, q) = \tanh((W_{x_i}.W_q))$ (Vaswani et al., 2017). In our model training, we use an additive attention mechanism, as it helped improve the prediction performance.

Self-Attention mechanism. In our model training, we have a small number of corpora, which are not sufficient to train an efficient word embedding or alleviate well-known problems such as polysemy. In an effort to overcome such limitations, we use a self-attention mechanism. This approach measures the dependency of different tokens in the same input embedding (x_i). It mainly computes attention for each word by replacing q and x_i with a set of token pairs (x_i, x_j).

4.3 Character-level CNN

The CNN encoding layer (Figure 3) takes an input of a sequence (S) of characters, where $S = [s_1, s_2, \dots, s_t]$ such that t denotes the timestep. S_i is a one-hot input (i) vector of a fixed length (T) of characters.

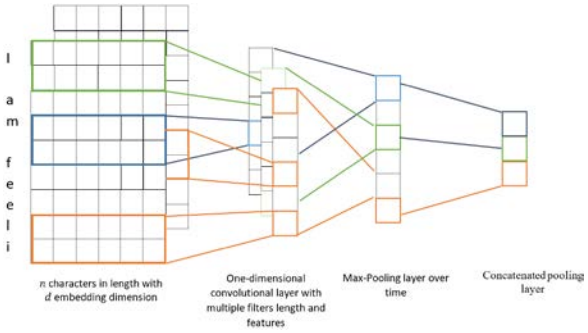

Figure 3: Character-level CNN.

CNN usually uses temporal convolutions (timestep-based) rather than spatial convolutions with text analysis.

We mainly use convolutions to extract low-level character information such as misspellings, slangs, and so on.

Character encoding. We define a charset of the size 95, including the upper and lower cases of the English alphabet, special characters, padding, and the start and end of a given input sequence. We need this charset to build a vocabulary, which is used to encode a character sequence. Similarly to the word embedding, we handle the variable length through padding and truncating (Section 4.1).

Character embedding layer. We build a character embedding of 32 dimensions. We use a uniform distribution scheme of a range (-0.5 to +0.5) to initialize its weight matrix.

We apply 3 convolutions of 100 features, as well as different filter lengths 2, 3, and 4. Each one-dimensional operation is used, where $C_i^n = Conv1d(S_i)$, and n is the filter length. After that, a max-pooling layer is applied on the feature map to extract abstract information, $\widehat{C_i^n} = \max(C_i^n)$. Then, we concatenate these feature representations into one output.

As opposed to recurrent layers (Section 4.2), convolutional operations with max-pooling are helpful to extract word features without paying attention to their sequence order (Kalchbrenner, Grefenstette, & Blunsom, 2014). These features are combined with recurrent features to improve the performance of our model.

4.4 Attention with Context

Output vectors received from previous steps are concatenated, and then fed into an attention with context.

We use a context-aware attention mechanism (Yang et al., 2016) to compute a fixed representation ($r = \sum_{i=1}^{T} a_i h_i$) of a sequence as the weighted sum of all tokens in that sequence. This representation is used as a classification feature vector to be fed to the final fully-connected sigmoid layer. This layer outputs a continuous value representing the valence of a given sentence.

4.5 Training

In our training, we use mini batch stochastic gradient of the size 32, to minimize the mean-squared error using back-propagation. We use Adam optimizer with a learning rate of 0.001 (Kingma & Ba, 2014). For training, we use 80% of the training set and 20% for validation. We test and report our results on both development and test sets.

Regularization. We use dropout to randomly drop neurons off the network, which helps preventing co-adaptation of neurons (Srivastava, Hinton, Krizhevsky, Sutskever, & Salakhutdinov, 2014). Dropout is also applied on the recurrent connection of our GRU-based layers. Additionally, we apply a weight decay approach through setting an L2 regularization penalty (Cortes, Mohri, & Rostamizadeh, 2012).

Hyperparameters. The size of the embedding layer is 200, and of the GRU layers is 150, which becomes 300 for bidirectional GRU. We apply a dropout of 0.4, and a dropout of 0.2 on the recurrent connections. Finally, an L2 regularization of 0.00001 is applied at the loss function.

5 Results

We report our results using the Pearson correlation between the prediction and gold rating sets on the test set (all instances). The other one (gold in 0.5-1 shown in Table 2) differs in including tweets only with intensity greater than or equal to 0.5.

Our model performed well on the development set scoring 0.869, while on the testing set, the performance degraded to 0.752. This degradation could be related to the size of the corpus we used to train our word embedding. We also trained only on 80% of the training set.

Dataset	**Valence task**	
	Pearson correlation (all instances)	Pearson correlation (gold in 0.5-1)
Development	0.869	0.692
Testing	0.752	0.559

Table 2: Results of valence intensity regression (English).

6 Conclusion

In this paper, we presented a self-attentive hybrid GRU-based network for predicting valence intensity for short text.

We used a hybrid approach combining low-character-level features with self-attentive word embedding. Our network uses two different attention mechanisms to signify the relevant and important words, and hence optimize feature representation.

With limited corpora and vocabulary of the size 8152, our model still managed to achieve an optimized feature representation, which achieved excellent results on the development set. However, our model failed to maintain the same performance on the testing set.

For future work, we will explore the performance of our model with larger corpora against the testing set. It would also be interesting to see if the model performs well on other long-text NLP tasks such as topic classification.

Acknowledgments

This research is funded by Natural Sciences and Engineering Research Council of Canada (NSERC), Ontario Centres of Excellence (OCE) and VISR.co.

References

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. Retrieved from http://arxiv.org/abs/1409.0473

Cortes, C., Mohri, M., & Rostamizadeh, A. (2012). L2 Regularization for Learning Kernels. Retrieved from http://arxiv.org/abs/1205.2653

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by Gibbs sampling. In Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics - ACL '05 (pp. 363–370). Morristown, NJ, USA: Association for Computational Linguistics. https://doi.org/10.3115/1219840.1219885

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. Neural Computation, 12(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. Neural Computation, 9(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). A Convolutional Neural Network for Modelling Sentences. In Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) (pp. 655–665). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/v1/P14-1062

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. Retrieved from http://arxiv.org/abs/1412.6980

Mohammad, S. M., Bravo-Marquez, F., Salameh, M., & Kiritchenko, S. (2018). Semeval-2018 Task 1: Affect in tweets. In International Workshop on Semantic Evaluation (SemEval-2018), New Orleans, LA, USA.

Mohammad, S. M., & Kiritchenko, S. (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 (LREC-2018).

Mohammad, S. M., Kiritchenko, S., & Zhu, X. (2013). NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. Retrieved from http://arxiv.org/abs/1308.6242

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532–1543). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.3115/v1/D14-1162

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), 1929–1958.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. Computational Linguistics, 37(2), 267–307. https://doi.org/10.1162/COLI_a_00049

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention is All you Need. In Advances in Neural Information Processing Systems 30 (NIPS).

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical Attention Networks for Document Classification. In Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (pp. 1480–1489). Stroudsburg, PA, USA: Association for Computational Linguistics. https://doi.org/10.18653/v1/N16-1174

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative Study of CNN and RNN for Natural Language Processing. Retrieved from http://arxiv.org/abs/1702.01923

Young, T., Hazarika, D., Poria, S., & Cambria, E. (2017). Recent Trends in Deep Learning Based Natural Language Processing. Retrieved from http://arxiv.org/abs/1708.02709