UWB at IEST 2018: Emotion Prediction in Tweets with Bidirectional Long Short-Term Memory Neural Network

Pavel Přibáň[1,2] and **Jiří Martínek**[1,2]

[1]NTIS – New Technologies for the Information Society,
Faculty of Applied Sciences, University of West Bohemia, Czech Republic
[2]Department of Computer Science and Engineering,
Faculty of Applied Sciences, University of West Bohemia, Czech Republic
{pribanp,jimar}@kiv.zcu.cz
http://nlp.kiv.zcu.cz

Abstract

This paper describes our system created for the WASSA 2018 Implicit Emotion Shared Task. The goal of this task is to predict the emotion of a given tweet, from which a certain emotion word is removed. The removed word can be *sad*, *happy*, *disgusted*, *angry*, *afraid* or a synonym of one of them. Our proposed system is based on deep-learning methods. We use Bidirectional Long Short-Term Memory (BiLSTM) with word embeddings as an input. Pre-trained DeepMoji model and pre-trained emoji2vec emoji embeddings are also used as additional inputs. Our System achieves 0.657 macro F_1 score and our rank is 13^{th} out of 30.

1 Introduction

Emotions, especially on the social media and social networks, as an immediate response to a specific object or a situation, are a significant part of the communication between people. Even for a human, it is sometimes challenging to describe or recognize an emotion without imminent contact with a subject (e.g. idioms or sarcasm). One of the most important ways to express an emotion in a text is an emoji. Emojis are small ideograms depicting objects, people and scenes (Barbieri et al., 2018). Emojis try to capture a facial expression of a subject, which is determining for emotion detection.

This paper describes our system created for the WASSA 2018 Implicit Emotion Shared Task (Klinger et al., 2018). The goal of this task is to predict the emotion of a given tweet, from which a certain emotion word is removed, for example:

```
It is [#TARGETWORD#] when you feel
like you are invisible to others.
```

The removed word can be *sad*, *happy*, *disgusted*, *surprised*, *angry*, *afraid* or a synonym of one of them. The possible emotions are Sadness, Joy, Disgust, Surprise, Anger, and Fear. The `[#TARGETWORD#]` token in the example indicates a position of the removed word in the given tweet.

1.1 Related Work

As we mentioned before emojis are an important part of expressing emotions. Barbieri et al. (2017) investigated the relationship between words and emojis. They also proposed an approach to predict the most probable emoji that is associated with a tweet. The mentioned approach uses a Bidirectional Long Short–Term Memory networks (BiLSTM) (Graves and Schmidhuber, 2005).

Pre-trained word embeddings (word representations) such as (Mikolov et al., 2013; Pennington et al., 2014) are currently standard part in most of the state-of-the-art solutions for key NLP tasks.

Tang et al. (2014) propose a method that can learn sentiment–specific word embeddings, which are able to improve performance by combining with other existing feature sets.

There are also some previously submitted systems in similar SemEval shared tasks using deep learning models. Cliche (2017) uses a CNN and LSTM for Sentiment Analysis SemEval–2017 task 4 (Rosenthal et al., 2017). Another approach with a deep LSTM with Attention mechanism is used by Baziotis et al. (2017) for the same task. Most of the best performing submitted systems (Baziotis et al., 2018; Gee and Wang, 2018; Park et al., 2018) in SemEval-2018 Task 1: Affect in Tweets (Mohammad et al., 2018) also use deep learning models with LSTM or BiLSTM neural networks.

2 Overview

Our approach is based on the artificial neural network that combines word embeddings and emoji–

Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, pages 224–230

https://doi.org/10.18653/v1/P17

based features as input. We use Weka machine learning workbench (Hall et al., 2009) for preprocessing. Our submitted model combines BiLSTM layer for word embeddings input and dense layers for the other inputs (emoji2vec (Eisner et al., 2016) and DeepMoji (Felbo et al., 2017) features, see 2.2) connected to one dense layer, see the Figure 2 with a model architecture. Outputs of these three layers are concatenated and then a dropout (Srivastava et al., 2014) technique is applied. After the concatenating a next dense layer is employed. An output from the previous dense layer is then passed to a fully-connected softmax layer. An output of the softmax layer is a probability distribution over all six possible classes.

We trained several modified versions of our submitted model and we evaluated these models on the development data. The model with the highest macro F_1 score on the development data was then trained again on the training data extended by the development data. This model was used for test data predictions. All models were implemented by using Keras (Chollet et al., 2015) with TensorFlow backend (Abadi et al., 2015).

2.1 Tweets Preprocessing

Tweets often contain slang expressions, misspelled words, emoticons or abbreviations and it is needed to make some preprocessing steps before training and making predictions. We use a similar approach to Přibáň et al. (2018).

At first, we remove the `[#TARGETWORD#]` token, that represents a position of the removed word with a certain emotion and every tweet is tokenized using *TweetNLP twokenizer* (Gimpel et al., 2011). Then the following steps are applied on tokens:

1. Tokens are converted to lowercase
2. Tokens containing sequences of letters occurring more than two times in a row are replaced with two occurrences of them (e.g. *huuuungry* is reduced to *huungry*, *looooove* to *loove*)
3. From hashtags (tokens starting with #) the # character is removed.
4. Common sequences of words and emojis are separated by space (e.g. token "*nice:D:D*" is split into three tokens "*nice*", "*:D*" and "*:D*")
5. Characters `&_-` in tokens are replaced with space

Weka machine learning workbench is used to perform the mentioned steps. After tokenization and mentioned preprocessing the tweet is padded to 50 tokens. Tweets longer than 50 words are shortened, while to the shorter tweets padding tokens are added.

2.2 Features

We use three types of input features – word embeddings, emoji embeddings and an emotional representation of a sentence. **Word embeddings** are representations of words usually expressed as pre-trained dense real vectors (Mikolov et al., 2013; Pennington et al., 2014) with a fixed dimension size. We use pre-trained Ultradense Word Embeddings (Rothe et al., 2016) that were trained on Twitter domain corpus. The number of dimensions for this embedding is 400.

Pre-trained emoji2vec (Eisner et al., 2016) **emoji embeddings** (300-dimensional) are used as another input to our model. We average vectors for each emoji in a tweet and the resulting averaged vector is used as an input. The mentioned emoji2vec embeddings contain vectors for all Unicode emojis which were learned from their description in the Unicode emoji standard[1], see the (Eisner et al., 2016) for details. Emoji2vec embeddings can be used only for some tweets because not every tweet contains some emojis, but we suppose that using emoji2vec will lead to an overall performance improvement.

We also use DeepMoji (Felbo et al., 2017) as an **emotional sentence representation**. The DeepMoji model is able to predict emoji that is included with a given sentence and thus the model has also an understanding of the emotional content of that sentence. The model was trained on a dataset of 1.2 billion tweets. As an input for our model, we use the 2304-dimensional vector from the attention layer in the pre-trained DeepMoji model.

2.3 Recurrent Neural Network

The Recurrent Neural Network (RNN) extends the classic (feed–forward) neural network. An RNN is intended for sequential data. The actual hidden state h_t of the RNN depends on the previous hidden state h_{t-1} (see Figure 1). An RNN takes the input sequence $x_1, x_2 \dots x_T$ and for each element, at the time step t computes new hidden state h_t

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

from the input x_t and from the previous hidden state h_{t-1}. The new hidden state h_t is computed by hidden layer function $\mathcal{H}$.

$$h_t = \mathcal{H}(x_t, h_{t-1}) \tag{1}$$

In the simplest case, the hidden layer function $\mathcal{H}$ is defined as:

$$h_t = \sigma\left(W_{xh}x_t + W_{hh}h_{t-1} + b_h\right) \tag{2}$$

where the W terms correspond to weight matrices (e.g. W_{xh} is the input-hidden weight matrix) and b_h term is hidden bias vector. The concrete implementation of the $\mathcal{H}$ function depends on the type of the used RNN unit (Graves et al., 2013), for example Long Short-Term Memory (LSTM) unit (Hochreiter and Schmidhuber, 1997) or Gated Recurrent Unit (GRU) (Cho et al., 2014).

In our case, the input x_t denotes the word embedding vector for each word in the tweet and T is a length of the tweet. Every tweet is also padded to the length T. As mentioned, the new hidden state h_t depends on the previous hidden state and hence the word order is also taken into account in the RNN.

2.4 Long Short-Term Memory

The Long Short-Term Memory (Hochreiter and Schmidhuber, 1997) allows learning (remember) long-term dependencies from the input sequence. The LSTM unit consists of cell state (cell activation vector) input, forget and output gates. These gates control how the cell state is updated. The $\mathcal{H}$ function of the LSTM unit is defined as:

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + b_i\right) \tag{3}$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + b_f\right) \tag{4}$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + b_o\right) \tag{5}$$

$$c_t = f_t * c_{t-1} + i_t * \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \tag{6}$$

$$h_t = o_t * \tanh\left(c_t\right) \tag{7}$$

where the W terms correspond to weight matrices and b terms are bias vectors, i, f, o are the input, forget and output gates, c denotes cell state (activation vector), σ is sigmoid function and $*$ character means element-wise multiplication.

It is a common practice to use Bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005). The BiLSTM consists of two LSTMs, one LSTM process the input sequence from the first element x_1 to x_T and produces output vector $\overrightarrow{h_t}$. The second LSTM process the input sequence in reverse order e.g. from the last element x_T to x_1 and produces output vector $\overleftarrow{h_t}$. Both output vectors have dimension D. The final output vector h_t from BiLSTM with dimension $2D$ is then created by concatenating two vectors $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$.

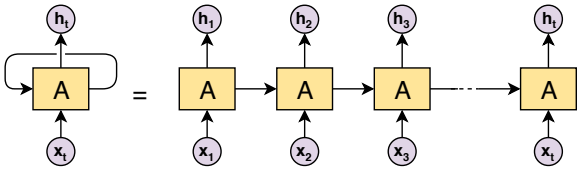

Figure 1: Basic RNN architecture[2]

Dropout (Srivastava et al., 2014) is a technique for improving neural networks by reducing overfitting. The dropout technique randomly drops out units (hidden and visible) during training and thus prevents co-adaption of neurons from training data.

3 Model Description

The proposed model has three inputs. Figure 2 shows the model architecture. The first input (**word embeddings**) represents tweet as a sequence of $t = 50$ tokens. We use the Ultradense Word Embeddings (Rothe et al., 2016) to obtain a vector of dimension $d = 400$ for each token from the tweet. The whole tweet is then represented as a matrix $M \in \mathbb{R}^{t \times d}$. The vectors are obtained only for 50,000 most frequent words in the training dataset. If the tweet word is not present in a vocabulary of 50,000 most frequent words, the randomly initialized vector of the same dimension is used. The **word embeddings** input is followed by a BiLSTM layer with 1200 units, respectively every single LSTM has 600 units. We also use a dropout to recurrent connections in BiLSTM layer.

The **emoji embeddings** input is based on emoji2vec (Eisner et al., 2016). For each emoji in the tweet, a 300-dimensional vector is produced by the pre-trained model. All emoji vectors for the tweet are then averaged to a single vector. If the tweet does not contain any emoji a zero vector is used. The resulting averaged **emoji embeddings** vector $E \in \mathbb{R}^{300}$ is used as an input to a dense layer with 300 units.

[2]Image is based on: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Our last input uses a pre-trained DeepMoji (Felbo et al., 2017) model for an **emotional sentence representation**. The DeepMoji model generates for each tweet vector $D \in \mathbb{R}^{2304}$ which represents the emotional content of the tweet. The **emotional sentence representation** input is followed by a dense layer with 2304 units.

All three output vectors of the BiLSTM and two dense layers are concatenated into one vector $C \in \mathbb{R}^{3804}$ that is passed to a next dense layer with 400 units. We also use a dropout after the concatenating. An output of the last dense layer is passed to a final fully-connected softmax layer. An output of the softmax layer is a probability distribution over all six possible classes. The class with the highest probability is predicted as a final output of our model.

3.1 Model Training & Hyper-Parameters

We trained our model using mini-batches of size 1024 for 5 epochs and we used the Adam (Kingma and Ba, 2014) optimizer with learning rate 0.001 the other parameters of the Adam optimizer follow those provided in the cited paper. As an activation function in the BiLSTM and in the dense layers, we used a Rectified Linear Unit (ReLu). Dropout of 0.2 is used for the recurrent connections in BiLSTM layer and in all dense layers.

We trained the model on the provided training dataset and we evaluated the trained model on the development dataset. We experimented with different settings of the hyper-parameters (learning rate, mini-batch size etc.) but the mentioned settings showed to be the best one on the development data. These hyper-parameters settings were also used for final submission.

4 Experiments & Results

All presented experiments were evaluated on the provided development and test datasets. Table 1 shows the results for the different model settings.

We performed ablation study to see which features are the most beneficial (see Table 2). Numbers represent the performance change when the given feature is removed [3].

We also modified our model and we experimented with an attention mechanism (Rocktäschel et al., 2015; Raffel and Ellis, 2015). The attention mechanism was added to our BiLSTM layer[4] (see Table 1 with results obtained by the modified model).

	Macro F_1	
Model Settings	**dev data**	**test data**
Ultradense + DeepMoji + emoji2vec†	0.657	0.657
Ultradense + DeepMoji	**0.661**	**0.660**
Ultradense + emoji2vec	0.653	0.658
Ultradense	0.648	0.650
DeepMoji + emoji2vec	0.556	0.547
DeepMoji	0.560	0.552
emoji2vec	0.151	0.154
Ultradense + DeepMoji + emoji2vec*	0.661	0.656
Ultradense + DeepMoji*	0.654	0.653

*Model with added attention mechanism to BiLSTM layer
†Model used for the final submission

Table 1: Results for individual model settings

Our results for the WASSA 2018 Implicit Emotion Shared Task are shown in Table 3 along with some other teams. Table 4 contains the confusion matrix obtained from the submitted predictions and Table 5 contains Recall, Precision an F_1 measures that are computed from the confusion matrix.

Feature	**dev data**	**test data**
Ultradense + DeepMoji + emoji2vec*	0.657	0.657
Ultradense	**-0.101**	**-0.110**
DeepMoji	-0.004	-0.001
emoji2vec	0.004	-0.003

*Values used to calculate ablation results

Table 2: Feature ablation study

Team	**Macro F_1**	**Rank**
Amobee	0.714	1
IIIDYT	0.710	2
NTUA-SLP	0.703	3
hgsgnlp	0.658	12
UWB	**0.657**	**13**
NL-FIIT	0.655	14
BASELINE	0.599	20

Table 3: WASSA 2018 Implicit Emotion Shared Task official results

4.1 Discussion

Thanks to the ablation study (see Table 2) and results from Table 1 we can observe that the **Ultradense Word Embeddings** are the most important features for our model. The **DeepMoji** and the

[3]The lowest number denotes the most beneficial feature

[4]We experimented with an attention mechanism after the submission deadline and therefore the modified model cannot be used to make predictions for the final submission

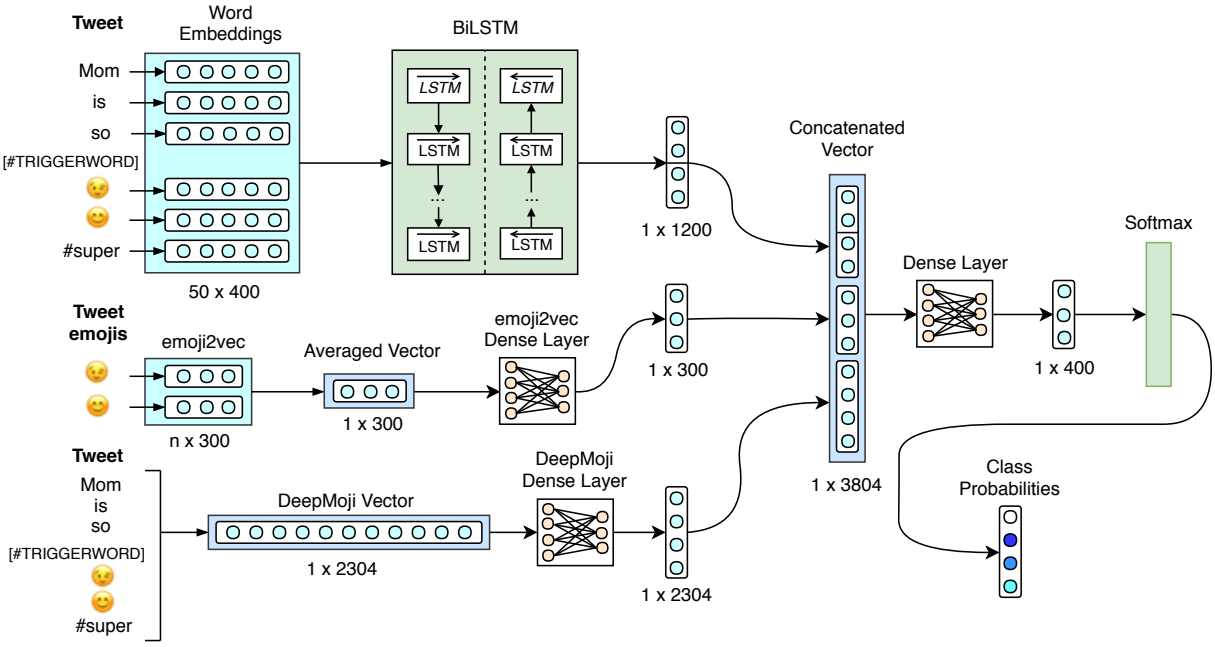

Figure 2: System architecture

		Predicted Labels					
		A	**D**	**F**	**J**	**Sa**	**Su**
Gold Labels	**A**	3011	250	441	326	431	335
	D	484	2807	274	204	596	429
	F	461	119	3484	243	210	274
	J	336	76	319	4021	304	190
	Sa	439	206	266	345	2939	145
	Su	530	331	569	315	345	2702

Table 4: Confusion Matrix on the test dataset (A, D, F, J, Sa, and Su are abbreviations for anger, disgust, fear, joy, sadness, and surprise respectively)

Emotion	**Recall**	**Precision**	**F_1 score**
Anger	0.628	0.572	0.599
Disgust	0.586	**0.741**	0.654
Fear	0.727	0.651	0.687
Joy	**0.766**	0.737	**0.752**
Sadness	0.677	0.609	0.641
Surprise	0.564	0.663	0.609

Table 5: Recall, Precision and F_1 score for each emotion of the test dataset

emoji2vec features also increase the performance of our model for the test dataset, but the contribution is insignificant and it is not so important as the word embeddings. So our assumption, that the emoji2vec feature will lead to a more significant overall performance improvement for the test dataset, is not correct. It would be more beneficial to use a simpler model without an emoji2vec feature.

The modified models with the attention mechanism did not improve the performance. The possible explanation is that it is caused by the missing emotion word in the classified tweet. The missing word carries probably the most information about the emotion. If the word was present in the classified tweet the attention mechanism would pay most attention to the missing word and thus the attention mechanism would improve the performance of our model.

Our model performs best for the joy and fear emotions (see Table 5). On the other hand, we obtained worst results for the anger emotion. Our model produces the most false positive predictions for the anger emotion (tweet is classified as anger but the true emotion is different). From the confusion matrix (Table 4), we can see that for our model it is difficult to distinguish especially between disgust and sadness, disgust and anger, fear and anger, surprise and anger, and between surprise and fear.

Table 1 shows that there are no important differences between the development dataset and test dataset results. So our decision to select the second best model (evaluated on the development dataset) for the final submission based on the results for the development dataset was suitable.

5 Conclusion

In this paper we described our UWB deep-learning system created for the WASSA 2018 Implicit Emotion Shared Task. Our system uses Bidirectional Long Short-Term Memory (BiLSTM) with word embeddings as an input. Pre-trained DeepMoji model and pre-trained emoji2vec emoji embeddings are also used as additional inputs. The proposed system performs best for the joy emotion. Our System achieves 0.657 macro F_1 score and our rank is 13^{th} out of 30.

We performed ablation study and showed that the most beneficial features are word embeddings. The emotional sentence representation (DeepMoji feature) and the averaged emoji vectors (emoji2vec feature) did not much improve the performance of our model.

In the future work, we would like to try another approach employing a twitter specific language model to predict probabilities for each emotion class for the missing target emotion word in the provided data. These probabilities could be used as input features to our model.

Acknowledgments

This work was partly supported from ERDF "Research and Development of Intelligent Components of Advanced Technologies for the Pilsen Metropolitan Area (InteCom)" (no.: CZ.02.1.01/0.0/0.0/17_048/0007267) and by Grant No. SGS-2016-018 Data and Software Engineerig for Advanced Applications.

References

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. 2015. TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Francesco Barbieri, Miguel Ballesteros, and Horacio Saggion. 2017. Are emojis predictable? *arXiv preprint arXiv:1702.07285*.

Francesco Barbieri, Jose Camacho-Collados, Francesco Ronzano, Luis Espinosa Anke, Miguel Ballesteros, Valerio Basile, Viviana Patti, and Horacio Saggion. 2018. Semeval 2018 task 2: Multilingual emoji prediction. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 24–33.

Christos Baziotis, Athanasiou Nikolaos, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth Narayanan, and Alexandros Potamianos. 2018. Ntua-slp at semeval-2018 task 1: Predicting affective content in tweets with deep attentive rnns and transfer learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 245–255. Association for Computational Linguistics.

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 747–754.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

François Chollet et al. 2015. Keras. https://keras.io.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. *arXiv preprint arXiv:1704.06125*.

Ben Eisner, Tim Rocktäschel, Isabelle Augenstein, Matko Bošnjak, and Sebastian Riedel. 2016. emoji2vec: Learning emoji representations from their description. *arXiv preprint arXiv:1609.08359*.

Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. 2017. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Grace Gee and Eugene Wang. 2018. psyml at semeval-2018 task 1: Transfer learning for sentiment and emotion analysis. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 369–376. Association for Computational Linguistics.

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, HLT '11, pages 42–47, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5-6):602–610.

Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. 2009. The WEKA data mining software: An update. *SIGKDD Explorations*, 11(1):10–18.

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

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

Roman Klinger, Orphée de Clercq, Saif M. Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task. In *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, Brussels, Belgium. Association for Computational Linguistics.

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

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.

Ji Ho Park, Peng Xu, and Pascale Fung. 2018. Plusemo2vec at semeval-2018 task 1: Exploiting emotion knowledge from emoji and #hashtags. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 264–272. 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.

Pavel Přibáň, Tomáš Hercig, and Ladislav Lenc. 2018. Uwb at semeval-2018 task 1: Emotion intensity detection in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 133–140. Association for Computational Linguistics.

Colin Raffel and Daniel PW Ellis. 2015. Feed-forward networks with attention can solve some long-term memory problems. *arXiv preprint arXiv:1512.08756*.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2015. Reasoning about entailment with neural attention. *arXiv preprint arXiv:1509.06664*.

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. Association for Computational Linguistics.

Sascha Rothe, Sebastian Ebert, and Hinrich Schütze. 2016. Ultradense word embeddings by orthogonal transformation. *arXiv preprint arXiv:1602.07572*.

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.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. 2014. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1555–1565.