Prayas at EmoInt 2017: An Ensemble of Deep Neural Architectures for Emotion Intensity Prediction in Tweets

Pranav Goel*, **Devang Kulshreshtha***, **Prayas Jain and K.K. Shukla**

Indian Institute of Technology (Banaras Hindu University) Varanasi, India
{pranav.goel.cse14, devang.kulshreshtha.cse14, prayas.jain.cse14, kkshkla.cse}@iitbu.ac.in
publication@emnlp2017.net

Abstract

The paper describes the best performing system for EmoInt - a shared task to predict the intensity of emotions in tweets. Intensity is a real valued score, between 0 and 1. The emotions are classified as - anger, fear, joy and sadness. We apply three different deep neural network based models, which approach the problem from essentially different directions. Our final performance quantified by an average pearson correlation score of 74.7 and an average spearman correlation score of 73.5 is obtained using an ensemble of the three models. We outperform the baseline model of the shared task by 9.9% and 9.4% pearson and spearman correlation scores respectively.

1 Introduction

EmoInt (Mohammad and Bravo-Marquez, 2017) is a shared task hosted by WASSA 2017, aiming to predict the emotion intensity in tweets. The emotion can be one out of anger, joy, fear and sadness. For each tweet, the emotion is known, and the task is to *predict the intensity of the corresponding emotion*, where intensity is a real valued score ranging from 0 to 1. This is different from most of the other tasks or systems in the domain of emotion detection/sentiment analysis which tend to focus on classifying the tweets or text into different categories.

For example, given the tweet - 'I hate my lawn mower. If it had a soul, I'd condemn it to the fiery pits of Hell.' and the corresponding emotion - 'anger', the system has to predict a value for how intensely this emotion is felt by the author of the tweet which is as close as possible to the gold label intensity (0.833 in this case).

The systems built for this task are useful across various NLP applications, but perhaps most obviously in complementing sentiment analysis systems. For example, the degree of anger expressed in a grievance can be used to decide its priority of being addressed, and the intensity of joy can help decide which reviews to project when publicizing a product.

Our submitted system is an ensemble of three broad sets of approaches combined using a weighted average of the separate predictions (section 3). All the approaches rely on representing the input tweet as a word vector using the word2vec approach (Mikolov et al., 2013), and using neural network based architectures to finally give the intensity score for the tweet of the given emotion X (please note that we already know the emotion of the tweet in this task).

The shared task organizers provided the training and a small development dataset for building our systems, and then a period of about 2 weeks was given for submitting our predictions on a blind test set.[1]

The rest of the paper is structured as follows. Section 2 discusses in brief the dataset for the task. Section 3 explains the various approaches used by our ensemble model, the kind of experiments we carried out along with the details of the parameters which gave optimal results on cross validation, and the way we combined the predictions. Section 4 explains how the system is evaluated and Section 5 states the results we achieved and discusses the various implications of those results. We conclude our work in Section 6.

* these authors have equal contributions to the paper

[1] http://saifmohammad.com/WebPages/EmotionIntensity-SharedTask.html

2 Data

We used the dataset provided within the shared task for training our system. No other external datasets were used in training. The data files include the tweet id, the tweet, the emotion of the tweet and the emotion intensity (for training and dev sets). Test set's gold labels were given only after the evaluation period.
There are around 800-1100 tweets in the training set, 70-110 in the development set, and around 700-1000 in the test set (across all the emotions). The complete details of the dataset can be found in (Mohammad and Bravo-Marquez, 2017).

3 Proposed System

Our system is an ensemble of three sets of approaches. We describe the individual approaches, followed by the ensemble process. We mention the parameters for the optimal variants of each approach and the architecture based decisions or parameters that were varied to provide an insight into the scope of our experiments. The parameters were chosen so that they maximize the Pearson-correlation between the predicted and actual scores on the K-fold cross-validation. The evaluation method used to select the optimal variants is explained in section 4.
A bird's eye view of the various architectures is shown in Figure 1.

3.1 Approach 1: Feed-forward neural network

Feed forward neural networks have proven to be highly successful in classification and real value prediction based tasks across a variety of domains, including NLP applications ((Bengio et al., 2003), (Collobert et al., 2011)). (Deep) Neural networks have given state-of-the-art results in sentiment analysis (Tang et al., 2014) which is closely related to our task. Here we detail the architecture of our network -

Input features: Each tweet is represented as a 443 dimensional vector by concatenating two different feature vectors obtained as follows -

1. Word2Vec (Mikolov et al., 2013) representation of the tweet using publicly available embeddings (Godin et al., 2015) which were trained on 400 million tweets for the ACL W-NUT 2015 shared task (Baldwin et al., 2015). We chose it over other available pre-trained tweet based embeddings as it is trained on a large dataset and we also prefer its high dimensionality of 400. The vector for each word is *averaged* to get a 400 dimensional representation of the tweet.

2. TweetToLexiconFeatureVector is a filter in the AffectiveTweets[2] (Mohammad and Bravo-Marquez, 2017) package for converting tweets into numeric 43-dimensional vectors that can be used directly as features in our machine learning system. The filter calculates the features from the tweet using several lexicons:

   (a) MPQA Subjectivity Lexicon: Calculates the number of positive and negative words from the lexicon (Wilson et al., 2005)

   (b) Bing-Lui: Calculates the number of positive and negative words from the lexicon (Bauman et al., 2017)

   (c) AFINN: Wordlist-based approach for calculating positive and negative sentiment scores from the lexicon(Nielsen, 2011)

   (d) Sentiment140: Calculates positive and negative sentiment score provided by the lexicon in which tweets are annotated by lexicons (Mohammad and Turney, 2013)

   (e) NRC Hashtag Sentiment lexicon: Uses same lexicon as Sentiment 140 but here tweets with only emotional hashtags are considered during training.

   (f) NRC-10 Expanded: Emotional associations of words matching the Twitter specific expansion of the lexicon(Bravo-Marquez et al., 2016) are added to give the vale of this feature.

   (g) NRC Hashtag Emotion Association Lexicon: Emotional associations of words of the lexicon(Mohammad and Kiritchenko, 2015) are added to give the vale of this feature.

   (h) SentiWordNet: Calculates positive and negative sentiment score using SentiWordNet(Baccianella et al., 2010)

   (i) Emoticons: Calculates sentiment scores using word associations provided by

[2] https://github.com/felipebravom/AffectiveTweets

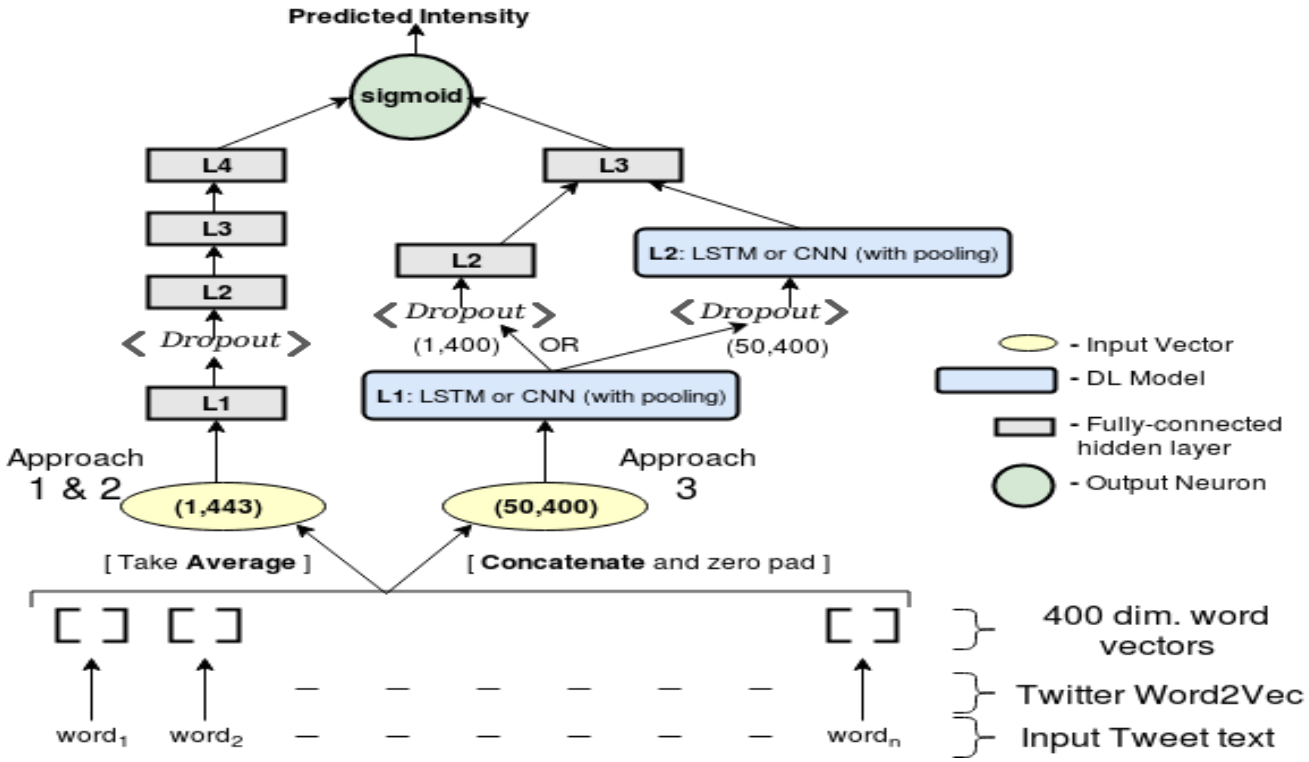

Figure 1: The architecture of our various approaches

emoticons from the lexicon(Nielsen, 2011)

(j) Negations: This feature simply count the number of negating words in the tweet.

Network Architecture: The input layer passes the 443 dimensional vector into 4 subsequent hidden layers ($L1, L2, L3, L4$) (the left half of Figure 1). We use Rectified Linear Unit ('relu') (Maas et al., 2013) as an activation function for each of the hidden layers (chosen as per the cross validation performance described in section 4). $L1$ is followed by $dropout$ (Srivastava et al., 2014) to avoid over-fitting and co-adaption of features. The number of hidden units in $L1 - L4$ and value of dropout (p) was varied, and the optimal settings were decided as per the cross validation performance for each emotion separately. The chosen values are mentioned in Table 1. $L4$ is followed by a single sigmoid neuron which predicts the intensity of the emotion between 0 to 1.

Training: The network parameters are learned by directly minimizing the negative of the Pearson-correlation (as it is a differentiable function) between actual and predicted intensities. We optimize the above function by back-propagating through layers via Mini-batch Gradient Descent.

Parameter/ Emotion	L1	p	L2	L3	L4
Anger	300	0.5	125	50	25
Fear	300	0.5	150	50	25
Joy	300	0.5	100	50	25
Sadness	300	0.5	125	50	25

Table 1: Network parameters for Approach 1

We use a batch size of 8, 30 training epochs and Adam optimization algorithm (Kingma and Ba, 2014) with the parameters set as $\alpha = 0.001, \beta_1 = 0.9, \beta_2 = 0.999\ and\ \epsilon = 10^{-9}$.

3.2 Approach 2: Multitask Deep Learning

Multitask learning using deep neural network via shared layers has become quite popular and successful as exploited in, for example (Collobert and Weston, 2008), and has been the focus of many cross lingual models like (Huang et al., 2013). (Collobert and Weston, 2008) described a single unified architecture for performing a variety of NLP tasks: named entity recognition, semantic similarity, part-of-speech tagging, etc. In this approach, we attempt to use the idea of multitask learning to explore the notion of generalized or shared learning across the different emotions.

Parameter/ Emotion	L1 (shared)	p	L2 (shared)	L3	L4
Anger(a)	300	0.3	150	50	20
Fear(b)	300	0.3	150	75	25
Joy(c)	300	0.3	150	50	15
Sadness(d)	300	0.3	150	50	20

Table 2: Network parameters for Approach 2

Input features: The input features are same as Approach 1 and same for all the 4 subtasks. We treat the 4 emotions as different subtasks to apply deep multi-task learning.

Network Architecture: The overall architecture can still be realized using the left side of figure 1. The network's initial layers are shared across multiple emotions with an objective to increase the generalization whereas the individual top layers can be seen as learning emotion specific features. Specifically, the system consists of two hidden layers ($L1$ & $L2$) shared between 4 regressors, while the last two layers ($L3$ & $L4$) are allowed to be different across the different subtasks (L3a, L3b, L3c, L3d and the same for L4). The model can be thought of as an input vector for the tweet going into the exact same two hidden layers regardless of the subtask, but then going into different layers (at the 3rd and 4th level) with the output from L4 going into their respective output neurons. The parameters (number of neurons in the shared as well as the non shared layers along with the dropout rate p) for each emotion are given in Table 2. Note that these parameters are optimized using cross validation (section 4).

Training: We use the same settings as in Approach 1 with respect to the cost function, optimization algorithm, update rule, learning rate, epochs, etc.

We train the network for 4 cycles at every epoch. During the 1^{st} cycle, we train the model for anger, where the input will pass through L1, L2, L3a, L4a and finally the corresponding output neuron. The network is similarly trained for fear, joy and sadness during the 2^{nd},3^{rd} and 4^{th} cycles respectively. Learning parameters this way ensures additional training examples for the initial layers (L1, L2) so that they may generalize well to learn task-independent representations while the higher layers (L3, L4) put pressure on the parameters to learn more task-specific representations.

3.3 Approach 3: Sequence Modeling using CNNs and LSTMs

Using Recurrent Neural Networks (RNN) has become a very common technique for various NLP based tasks like language modeling (Mikolov et al., 2010). Their time step based sequentially connected structure is intuitive to use for sequential data such as sentences. Long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) architecture is an advanced version of RNN that uses various gates to control the vanishing gradient problem (among other obstacles) that arise during the training of RNNs, and has found resounding success in a host of applications ((Graves and Jaitly, 2014), (Graves and Schmidhuber, 2005)). Convolutional Neural Network (CNN) is also a popular neural network based architecture, and has been successful in the NLP domain in various tasks ((Lee and Dernoncourt, 2016), (Kim, 2014)). Combining these architectures has also been found to be quite successful as in (Zhou et al., 2015) Both these architecture expect a sequence of vectors as input to operate on.

We describe how we use these deep learning models, which play a dominant role in our final ensemble system -

Input features: We again use the word2vec embeddings trained on twitter tweets ((Godin et al., 2015)) to represent the words in a tweet as 400 dimensional vectors, ignoring the words not found. These embeddings are ideal for representing tweets as they have been trained on a very large amount of tweets. Instead of averaging the word vectors as in our first two approaches, we *concatenate* them. Since length of different tweets can vary, we fix the length of each concatenated representation as 50 (since the maximum tweet length across the training and development data is 46 according to our analysis and we do not want to miss out on any information in the already short tweet) by performing zero padding. For datasets where a tweet may have length greater than 50, the number has to be tuned accordingly. Padding of zero vectors is done to make the representation of every tweet as a (50,400) vector. These representations are then fed to a host of architectures, whose general representation is given in the figure 1.

Parameter/ Emotion	L1	p	L2	L3
Anger (1)	CNN (250,Max)	0	125	50
Anger (2)	CNN (256,Avg)	0	100	-
Anger (3)	LSTM (300)	0	CNN (200,Avg)	100
Fear (1)	LSTM (256)	.2	CNN (150,Avg)	100
Fear (2)	CNN (250,Max)	0	125	50
Fear (3)	LSTM (250)	.2	CNN (120,Avg)	50
Joy (1)	CNN (256,Max)	0	100	-
Joy (2)	LSTM (300)	0	CNN (200,Avg)	100
Joy (3)	LSTM (300)	.2	CNN (200,Avg)	100
Sadness(1)	CNN (250,Max)	0	125	50
Sadness(2)	CNN (250,Max)	.2	125	50
Sadness(3)	CNN (256,Max)	0	100	-

Table 3: Network Parameters for the 3 best models built according to Approach 3 (Ranked as per the cross validation scores ; The numbers in the Layer (L) columns represent the output dimensionality of that layer ; Max and Avg refer to the type of pooling)

Network Architecture: As shown in figure 1, the concatenated vector representation of the tweet is first fed to a **LSTM** or **CNN** and then some fully connected (dense) hidden layers. The representation learned in the last hidden layer is fed to a single sigmoid neuron which gives us the intensity of the emotion (as in the previous 2 approaches). We tried many variations of the different parameters involved in constructing this model (keeping all others fixed while one is varied) to come up with several architectures but show the parameters for only the three top performing ones (as per cross validation) for each emotion in Table 3. The variations we tried include -

i) using only LSTM/CNN plus fully connected layers, and also the combination of these architectures with the initial LSTM's output for each word fed to a CNN, or vice versa.

ii) Using Simple RNN, Bidirectional LSTM ((Schuster and Paliwal, 1997), (Godin et al., 2015)), Gated Recurrent Units (GRU) (Cho et al., 2014) instead of LSTM.

iii) Using (global) max pooling versus (global) average pooling for CNNs.

iv) Using dropout (Srivastava et al., 2014). Note that a dropout layer was added after pooling layer for a CNN, while the same dropout rate was set for both matrices involved in the standard definition in case of LSTM (Zaremba et al., 2014).

v) Using different number of neurons for CNN/LSTM/fully connected hidden layers. (usually starting from 300 or 256, and halving the number of neurons as we went deeper)

vi) Using different number of fully connected hidden layers (0,1 or 2 in between the LSTM/CNN layer and sigmoid neuron).

In every case, 'relu' activation function was used in the hidden dense layers (except the last neuron which uses sigmoid). Dropout, if applied was always set to 0.2 (we also experimented with 0.1,0.3,0.4 and 0.5 as the dropout rate). Also, the filter height used for CNNs was always set to 3, and striding length for convolution was always 1.

Training: The network parameters are learned by minimizing the Mean Absolute Error between the actual and predicted values of emotion intensity. We optimize this loss function by back-propagating through layers via Mini-batch Gradient descent, with batch size of 8, 15 training epochs and Adam optimization algorithm (Kingma and Ba, 2014) with the same parameters as mentioned in Approach 1.

The deep learning based models in all the above approaches were implemented in Python using Keras library (Chollet et al., 2015).

3.4 Bringing it all together: The submitted ensemble system

As described above, we now have 5 models to combine - 1 each out of Approach 1 and 2, and 3 from Approach 3. We take a weighted average of the predictions from each of the system to form our final submission. The weights are informed from the results from cross validation (the CV score as explained in section 4), and are as follows - 1 for Approach 1, 3 for Approach 2, 3 each for the two best systems from approach 3 (which

Approach	Average		Anger		Fear		Joy		Sadness	
	CV	**Test**	**CV**	**Test**	**CV**	**Test**	**CV**	**Test**	**CV**	**Test**
Feed Forward NN	69.75	69.58	66.22	67.88	72.71	72.42	72.08	68.26	67.99	69.77
Multitask DL	66.30	66.20	63.73	64.49	68.07	67.74	66.80	65.37	66.65	67.22
CNN+LSTM Seq. Modeling	70.70	71.79	69.22	70.15	72.08	72.95	73.22	69.14	68.29	74.93
CNN+LSTM Seq. Modeling	70.25	72.15	69.08	69.86	70.95	73.27	72.93	69.86	68.04	75.6
CNN+LSTM Seq. Modeling	70.03	71.81	68.90	69.71	70.67	72.92	72.81	69.57	67.74	75.06
Ensemble Model	**75.26**	**74.70**	**72.94**	**73.2**	**76.78**	**76.20**	**74.42**	**73.20**	**76.90**	**76.50**
Baseline	61.10	64.8	60.50	63.9	57.40	65.2	70.30	65.4	56.20	64.8

Table 4: Results

are very close in performance as can be seen in Table 4), and 2 for the 3rd best system in approach 3. Our ensemble model improves the performance by at least 2% over any of our individual models (Table 4).

4 Evaluation

Cross Validation (CV): We combined the training and development sets, trained on 80% of this set while predicting on the remaining 20%, and repeated this seven times (for each emotion separately). The average of these was used as the CV score to evaluate our models. The metric used for evaluating performance was Pearson Correlation.

Test: The optimal setting for each model was decided using the CV score (Table 4). Then these chosen models (as described in Table 1,2 and 3) were used to generate predicted intensities on the test set, by training on the *full training and development sets combined*. Again an average of seven runs was taken. The predictions for the final ensemble model are generated using a weighted average of the individual predictions as described in section 3.4.

5 Results and Discussion

We compare the results achieved by our individual approaches, the submitted ensemble system and the WEKA Baseline system which is the official baseline model for this task (Mohammad and Bravo-Marquez, 2017) in Table 4. For brevity, we only show the Pearson Correlation scores on the test set (although the Spearman correlation scores show similar trends). We discuss the major takeaways from these results -

1. Our submitted ensemble model achieves an average (or overall) score of 75.26% and 74.70%, which beats the baseline model by about 14% and 10% on cross validation and test sets respectively. The improvement points to the potential of deep learning based models over the simpler lexicon based approaches. These are also the best scores among all participating systems in the shared task (according to the public leaderboard [3]).

2. The ensemble model achieves about 3-5% improvement over the average scores, and offers significant improvement in performance across all the emotions, which indicates that the approaches do complement each other quite well.

3. Approach 2 (Multitask DL) achieves the lowest scores among the three sets of approaches. Among Approach 1 (Feed Forward NN) and Approach 3 (CNN+LSTM Seq. Modeling), approach 3 has a best test score of 72.15 compared to approach 1's 69.58, which is a significant improvement and points to sequential models like LSTMs and CNNs being a better choice over feed forward neural networks.

[3] https://competitions.codalab.org/competitions/16380#results

4. Among the individual emotions, our ensemble model gives the best performance for 'Sadness', followed very closely by 'Fear', then 'Joy' and finally 'Anger'.

6 Conclusion and Future Work

In this paper, we propose a deep learning framework to predict the intensity of the emotion in tweets exhibiting that emotion. The proposed approach is based on an ensemble of Feed-Forward Neural Networks, Multi-Task Deep Learning and Sequence Modeling using CNNs and LSTMs, allowing us to explore the different directions a neural network based methodology can take. Each individual approach is described in detail with a view of making our experiments replicable. The optimal parameters are mentioned, along with our method of bringing the approaches together. Our submitted system beats the baseline system by about 10% on the test set.

Although our model achieves state-of-the-art results, there is definite room for improvement. In the future, we would like to experiment with handcrafted features in addition to word-vectors and lexicon features. We would also experiment with other filters provided in AffectiveTweets package (Mohammad and Bravo-Marquez, 2017) such as TweetToSentiStrengthFeatureVector, TweetNLPTokenizer etc. Another very interesting idea would be to try better ways of 'ensembling' the different models and analyze how each system or approach complements the other.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *LREC*. volume 10, pages 2200–2204.

Timothy Baldwin, Marie Catherine De Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text (WNUT 2015), Beijing, China*.

Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews .

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. 2003. A neural probabilistic language model. *Journal of machine learning research* 3(Feb):1137–1155.

Felipe Bravo-Marquez, Eibe Frank, Saif M Mohammad, and Bernhard Pfahringer. 2016. Determining word-emotion associations from tweets by multi-label classification. In *Web Intelligence (WI), 2016 IEEE/WIC/ACM International Conference on*. IEEE, pages 536–539.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* .

François Chollet et al. 2015. Keras. https://github.com/fchollet/keras.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*. ACM, pages 160–167.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. 2015. Multimedia lab@ acl w-nut ner shared task: named entity recognition for twitter microposts using distributed word representations. *ACL-IJCNLP* 2015:146–153.

Alex Graves and Navdeep Jaitly. 2014. Towards end-to-end speech recognition with recurrent neural networks. In *ICML*. volume 14, pages 1764–1772.

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

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

Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. 2013. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*. IEEE, pages 7304–7308.

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* .

Ji Young Lee and Franck Dernoncourt. 2016. Sequential short-text classification with recurrent and convolutional neural networks. *arXiv preprint arXiv:1603.03827* .

Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*. volume 30.

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

Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*. volume 2, page 3.

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. WASSA-2017 shared task on emotion intensity. In *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA)*. Copenhagen, Denmark.

Saif M Mohammad and Svetlana Kiritchenko. 2015. Using hashtags to capture fine emotion categories from tweets. *Computational Intelligence* 31(2):301–326.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a word-emotion association lexicon 29(3):436–465.

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

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

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, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*. pages 208–212.

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*. Association for Computational Linguistics, pages 347–354.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329* .

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630* .