psyML at SemEval-2018 Task 1: Transfer Learning for Sentiment and Emotion Analysis

Grace Gee, Eugene Wang
psyML, Inc.
grace.eugene@psyml.co

Abstract

In this paper, we describe the first attempt to perform transfer learning from sentiment to emotions. Our system employs Long Short-Term Memory (LSTM) networks, including bidirectional LSTM (biLSTM) and LSTM with attention mechanism. We perform transfer learning by first pre-training the LSTM networks on sentiment data before concatenating the penultimate layers of these networks into a single vector as input to new dense layers. For the E-c subtask, we utilize a novel approach to train models for correlated emotion classes. Our system performs 4/48, 3/39, 8/38, 4/37, 4/35 on all English subtasks EI-reg, EI-oc, V-reg, V-oc, E-c of SemEval 2018 Task 1: Affect in Tweets.

1 Introduction

SemEval-2018 Task 1: Affect in Tweets (Mohammad et al., 2018) is a shared task expanding on previous SemEval sentiment tasks and the WASSA-2017 Shared Task on Emotion Intensity (Mohammad and Bravo-Marquez, 2017). It presents 5 tasks:

1. Emotion intensity regression (EI-reg): given tweet and emotion (fear, anger, joy or sadness), predict real-valued emotion intensity from 0 to 1.

2. Emotion intensity ordinal classification (EI-oc): given tweet and emotion (fear, anger, joy or sadness), predict emotion intensity ordinal class from 0 (no emotion) to 3 (high emotion).

3. Sentiment intensity regression (V-reg): given tweet, predict real-valued sentiment intensity from 0 (no sentiment) to 1 (high sentiment). In this subtask, the directionality of the tweet sentiment is ignored. A negative tweet will be given the same score as a positive tweet with the same valence.

4. Sentiment intensity ordinal classification (V-oc): given tweet, predict sentiment ordinal intensity class from -3 (very negative) to 3 (very positive).

5. Emotion classification (E-c): given tweet, predict for each one of 11 emotions (anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise, trust) whether the emotion is neutral (0) or present (1).

The task is particularly challenging since E-c and EI-oc are completely new subtasks. Thus, no prior data or working models are available for comparison. The leaderboard is also not public during the competition. As shown in Table 1, taken from (Mohammad and Kiritchenko, 2018), the development sets are particularly small compared to the test sets, and the test sets are comparable in size to the training sets, so the model must generalize.

For EI-oc and EI-reg, the development and test sets are also annotated separately from the training sets. This impacts performance as our system would have placed 1st with average pearson score 0.755 on the WASSA 2017 task, in which the EI-reg train, development, and test data are annotated in the same format. Furthermore, tweets are difficult to analyze due to the unstructuredness of its language (hashtags, emoticons, slang, misspellings, poor grammar).

Previously submitted systems in SemEval sentiment analysis use deep learning models such as CNN, RNN and LSTMs (Baziotis et al., 2017; Cliche, 2017; Rouvier, 2017). In a previous run of EI-reg in WASSA-2017 Shared Task on Emotion Intensity, top performing teams use deep learning models (Goel et al., 2017; Köper et al., 2017) and

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 369–376
New Orleans, Louisiana, June 5–6, 2018.

Dataset	train	dev	test	Total
EI-reg, EI-oc				
anger	1,701	388	1,002	3,091
fear	2,252	389	986	3,627
joy	1,616	290	1,105	3,011
sadness	1,533	397	975	2,905
V-reg, V-oc	1,181	449	937	2,567
E-c	6,838	886	3,259	10,983

Table 1: Number of tweets in SemEval-2018: Affect in Tweets Dataset.

classifiers such as Support Vector Regressors or Random Forest Regressors (Duppada and Hiray, 2017; Köper et al., 2017). In both tasks, some participants use an ensemble approach (Goel et al., 2017; Duppada and Hiray, 2017; Rouvier, 2017).

To extract linguistic features, some systems employ pre-trained word embeddings (Baziotis et al., 2017; Cliche, 2017) or a combination of manually created features and/or lexicons (Köper et al., 2017; Duppada and Hiray, 2017). However, exclusively relying on hand-crafted features for EI-reg may result in a model that fails to encompass unforeseen linguistic relationships. Similarly, relying exclusively on deep learning models without lexicon inputs can lead to simple misclassifications due to the small training data.

To combine the best of both worlds, previous systems collapse high-dimensional word embeddings into a single dimension arithmetically, before combining it with hand-crafted features (usually one-dimensional). Goel et al. for instance averaged the word embeddings for each word in a tweet in order to concatenate it with a 43-dimensional vector. Duppada and Hiray simply averaged the two top performing model outputs.

In this paper, we present a deep learning system whose variants competed competitively in all English subtasks in SemEval-2018 Task 1: Affect in Tweets, specifically EI-reg, EI-oc, V-reg, V-oc, and E-c. We make the following contributions:

- A deep learning system that can take in a combination of one-dimensional hand-crafted and multi-dimensional word embedding inputs.
- A deep learning system that uses transfer learning from sentiment tasks to overcome the lack of training data compared to test data. To the best of our knowledge, this is the first instance of transferring knowledge from sentiment to emotion.
- Specifically for Task E-c, procedures for training correlated target classes.

2 Overview

Fig 1 shows an overview of our system, which consists of three steps: (1) preprocessing input using a text processor and the Weka AffectiveTweets package[1] (Mohammad and Bravo-Marquez, 2017) (2) pre-training Components A to C using sentiment data (3) training the entire system, including Components A, B, C, E, using subtask-specific dataset.

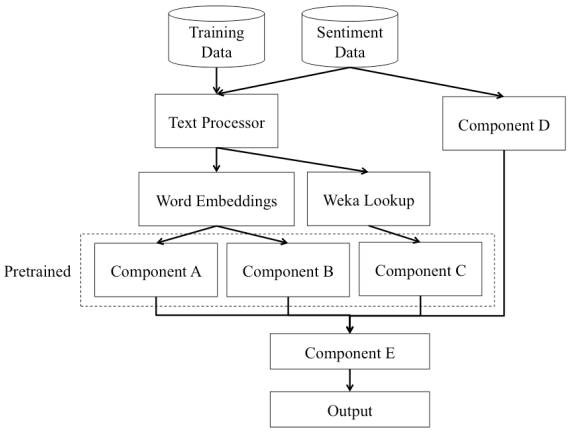

Figure 1: System Overview.

2.1 Preprocessing

We use the ekphrasis text processor[2] and word embeddings[3] built by Baziotis et al. Ekphrasis corrects for spelling, emoticons, emojis, splits hashtags and recognizes emphasized words. Its 300-dimension word embeddings are trained on 330M Twitter messages using GloVe. Other embeddings such as Stanford's GloVe (Pennington et al., 2014) do not incorporate newer popular unicode emojis. To build the Weka Lookup, we pass all 658,114 tokens in Baziotis et al. embedding dictionary into the TweetToLexiconFeatureVector filter in the Weka AffectiveTweets package. Of the 658,114, only 59,235 tokens returned nonzero

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

[2] https://github.com/cbaziotis/ekphrasis

[3] https://github.com/cbaziotis/datastories-semeval2017-task4

vectors. The TweetToLexiconFeatureVector returns a 43-dimension feature vector using sentiment and emotion lexicons such as Bing-Liu, AFINN, Sentiment140, and NRC-10 Expanded.

2.2 Transfer Learning

Transfer learning is the process of using knowledge from solving a source task to help performance in a target task. In particular, transfer learning is useful when the target task training set is small.

Another common way to deal with small data is distant supervision (Mintz et al., 2009), a process for generating labelled data from an unlabelled set according to a set of rules. For instance, for a sentiment analysis task, distant supervision can involve labelling tweets with smileys as positive and those with sad emojis as negative (Read, 2005).

Transfer learning has historically performed well on computer vision problems (Yosinski et al., 2014; Razavian et al., 2014). Traditionally, the CNN layer weights are frozen, its output treated as a feature vector input to a fully-connected layer, which will learn the new target task. Intuitively, the CNN will learn low-level image features on the source task while the dense layers will use these low-level features to predict a new target task.

Another strategy is to unfreeze the later layers weights of the pre-trained network and instead backpropagate all the way to the pre-trained network. In this case, the later layers of the pre-trained network can be fine-tuned. We choose to leave all weights from the pre-trained network unfrozen.

Transfer learning in natural language processing applications has been largely successful only within the same task such as POS tagging or sentiment (Blitzer et al., 2006, 2007). For different domains, good results are only achieved in semantically equivalent transfer (in which a source task and target task have the same objective but different data) but not for semantically different transfer (in which a source and target task have different objectives) (Mou et al., 2016).

For all subtasks, we will use transfer learning to pre-train our models on sentiment data. The source task objective is to predict sentiment categorical classes ('positive', 'negative', or 'neutral') given a tweet. Since the source task is not equivalent to any of the target tasks, we'd expect lower performance than those experiments on domain adaptation.

There are two main ways to perform transfer learning, the parameter initialization approach, in which a model is trained on a source task and the weights are transferred to a target task, and multi-task learning, in which a model is trained to learn multiple tasks simultaneously. We choose to implement the parameter initialization approach as Mou et al. has shown both approaches to be comparable.

2.3 Neural Network

The Recurrent Neural Network (RNN) is an extension of the traditional neural network that allows sequential data. Fig 2 shows the architecture of a standard RNN which takes in a sequence of word embeddings $x_1, x_2, \ldots, x_N$ and outputs hidden states $h_1, h_2, \ldots, h_N$, where N is the length of the tweet.[4]

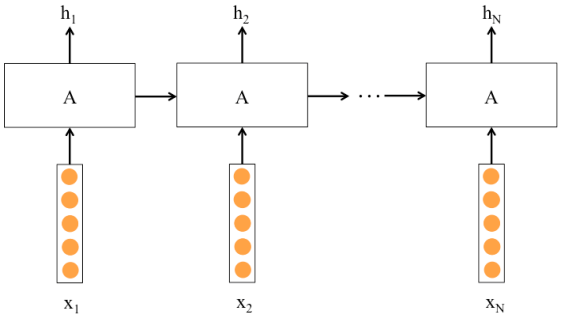

Figure 2: RNN basic structure.

In its simplest form, the hidden state $h_t \in \mathbb{R}^d$ (where d is the size of the RNN at time step t) is a function f of the current word embedding x_t, the past hidden state h_{t-1}, and the learned θ parameters.

$$h_t = f(x_t, h_{t-1}; \theta) \quad (1)$$

Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997) is a special form of RNN with a memory cell and input, forget, and output gates that allow it to take into account long-term dependencies. It is more widely used than RNN since it overcomes the vanishing or exploding gradient problem common in RNNs. The architecture of a standard LSTM will be the same as that of an

[4]Figure recreated from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN as shown in Fig 2, but with a different repeating A module. Formally, each LSTM cell is computed as follows:

$$X = \begin{bmatrix} h_{t-1} \\ x_t \end{bmatrix} \quad (2)$$

$$f_t = \sigma(W_f \cdot X + b_f) \quad (3)$$

$$i_t = \sigma(W_i \cdot X + b_i) \quad (4)$$

$$o_t = \sigma(W_o \cdot X + b_o) \quad (5)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c \cdot X + b_c) \quad (6)$$

$$h_t = \sigma_t \odot tanh(c_t) \quad (7)$$

where f_t is the forget gate, i_t the input gate, c_t the cell state, h_t the hidden state, σ the sigmoid function and $\odot$ element-wise multiplication.

We use bidirectional LSTM (biLSTM) to incorporate both past and future context. A biLSTM employs two LSTMs, one reading forward, $\overrightarrow{LSTM}$, on the input sequence and another backward, $\overleftarrow{LSTM}$.

Whereas in a traditional LSTM, output h_i only incorporates information up till time step i, or $x_1, x_2, \ldots, x_i$, a bidirectional LSTM would incorporate information from both past input $x_1, x_2, \ldots, x_i$ and future input $x_i, x_{i+1}, \ldots, x_N$. Fig 3 shows the architecture of a standard biLSTM.

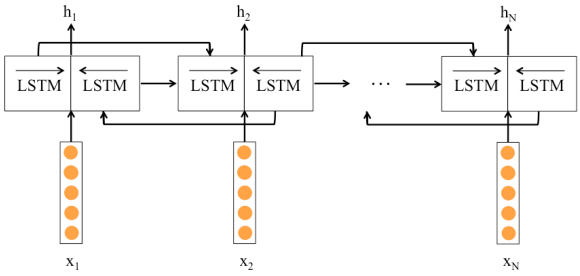

Figure 3: Bidirectional LSTM.

We use an attention mechanism (Rocktäschel et al., 2015) to learn which words in a tweet contribute more to a target task. Fig 4 shows the architecture of a standard LSTM with attention mechanism.

The attention layer is usually a 1 or 2-layer neural net that takes the output of an LSTM or RNN as input. It assigns "attention weight" α_i to each hidden state h_i and outputs a weighted representation r of the hidden states.

$$e_i = tanh(W_h h_i + b_h) \quad (8)$$

$$\alpha_i = softmax(e_i) \quad (9)$$

$$r = \sum_N \alpha_i h_i \quad (10)$$

where α is an attention weight vector, r is a weighted representation of the hidden states, and W_h and b_h are learned during backpropagation.

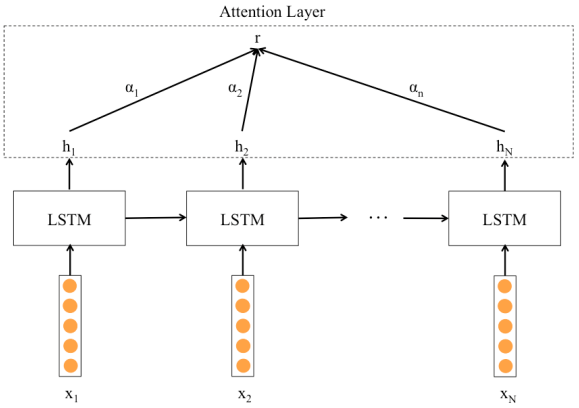

Figure 4: LSTM with Attention.

Components A and C are bidirectional LSTMs. Component B is a LSTM with attention mechanism.

3 Data

We used the dataset provided by the SemEval challenge.

For transfer learning, we pre-train Components A to C using train, development and test data from 2013 to 2017 SemEval Task 4A sentiment analysis classification tasks[5]. In total, this provided 7,723 negative, 22,195 neutral and 19,652 positive tweets.

4 Model

Component A is a 1-layer biLSTM. The input is a matrix $A \in \mathbb{R}^{n \times d}$, where n is the number of words in the tweet and d is the dimension of the word embedding. Component B is an LSTM with attention and takes in the same input as Component A. The attention layer is a 1 unit dense layer. Component C is also a 1-layer biLSTM like Component A, except the input is a matrix

[5] http://alt.qcri.org/semeval2017/task4/

$C \in \mathbb{R}^{n \times 43}$, since Weka's TweetToLexiconFeatureVector returns a 43-dimension vector and C is just a concatenated sequence of 43-dimension vectors of words passed into the TweetToLexiconFeatureVector. The tweet input for Components A through C are all zero padded. Component D just takes in an entire tweet and returns the 43-dimension TweetToLexiconFeatureVector vector. Component E is a 5-layer fully-connected neural net.

4.1 Regularization

In Components A through C, we apply dropout to both input and recurrent connections. Dropout (Srivastava et al., 2014) is a technique that involves randomly dropping units during training to prevent overfitting and co-adaptation of neurons. By randomly dropping units, neighboring neurons make up for the dropped units and learn representations for the target, resulting in a more robust network.

In Components A and C we apply global max pooling for the final layer.

We incorporate mini-batch gradient descent in all our models. Mini-batch gradient descent is calculated over small batches of data instead of the entire dataset as in traditional gradient descent. Compared to stochastic gradient descent, where gradient descent is calculated after every example, it is not as computationally intensive.

In the E-c dataset and sentiment classification dataset, some classes are overrepresented. This class imbalance can lead to bias in model output. To overcome this, for E-c we apply class weight to the loss function to boost recall of the minority class.

4.2 Hyper-parameters

We train all our models using mini-batches of size 8, and Adam (Kingma and Ba, 2014) optimization. For the E-c subtask, we minimize binary cross entropy loss. For EI-oc, EI-reg, V-oc and V-reg, we minimize mean squared error.

Models Components A through C all have dropout of 0.2 for their input layer and recurrent connections.

Components A and C are both biLSTM with 256 units. Component A takes in $A \in \mathbb{R}^{50 \times 300}$, while Component C takes in $C \in \mathbb{R}^{50 \times 43}$. We chose 50 since none of the tweets are longer than 50 words. Finally, we add a global max pooling layer.

Component B is a LSTM of 256 units, and takes in $B \in \mathbb{R}^{50 \times 300}$.

For source task learning, we apply a dense layer of 3 hidden units with sigmoid activation function to Components A through C.

Component E is 5 dense layers with 300, 125, 50, 25, and 1 hidden units. We use Rectified linear unit (ReLU) as the activation function for the former four layers.

4.3 Training

Source Task Learning During source task learning, we train each of Components A through C individually on the sentiment dataset with 10% hold out validation. The source task objective is to predict sentiment categorical class ('positive', 'negative', or 'neutral'). For each of the models, we train it for 1, 2, 3, 4, and 5 epochs and save the best performing one.

We experimented with RNN, CNN, LSTM, and biLSTM before settling on biLSTM and LSTM with attention.

Target Task Learning During target task learning, the final dense layers in Components A through C are removed and the penultimate layers are concatenated together with the output from Component D into a single vector as input to Component E. None of the weights are frozen and the entire system (Components A through E) is trained for between 5 to 10 epochs for subtasks EI-oc, EI-reg, V-oc, and V-reg. We choose the best performing model based on performance on the development set.

The final layer in Component E uses sigmoid as an activation function for EI-reg, EI-oc, V-reg, E-c and hyperbolic tangent for V-oc. The following functions are used to transform the ordinal classes of EI-oc, $d_{EI-oc} \in \{0, 1, 2, 3\}$, and V-oc, $d_{V-oc} \in \{-3, -2, -1, 0, 1, 2, 3\}$, to the output space of [0, 1] and [-1, 1] respectively.

$$d'_{EI-oc} = d_{EI-oc} * 0.25 + 0.125 \quad (11)$$

$$d'_{V-oc} = d_{V-oc} * 2/7 \quad (12)$$

For E-c, we notice the emotion classes are intercorrelated. The following Figure 5 is a dendrogram generated from E-c training data. The emotions are hierarchically clustered based on their correlations. The horizontal axis labels correspond to the 11 classes. The shorter the length of

the sideways n-shape, the more correlated the two classes.

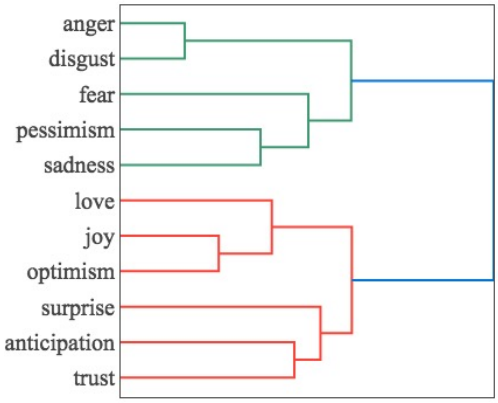

Figure 5: Dendrogram of 11 emotions, clustered using correlation and hierarchical clustering.

For E-c, we obtain the following emotion clusters from the dendrogram: [anger, disgust], [sadness, pessimism, fear], [joy, optimism, love], [anticipation, surprise, trust]. For each of the clusters, we obtain a fresh copy of the entire pre-trained system and train it for 2 epochs consecutively for each target class in the aforementioned order within the emotion cluster. The pseudocode is as follows. We call this method "cluster training":

Algorithm 1 Cluster training

```
for emotionCluster in emotionClusters do
    S ← System from Fig 1
    for emotion in emotionCluster do
        Train S for 2 epochs on emotion training
        Predict emotion test with S
```

4.4 Experimental Setup

We build all the models with the Keras library and train them on Google Datalab. The dendrogram diagram is built with Plotly.

5 Evaluation & Results

SemEval Results Our model ranks 4/48 in EI-reg, 3/39 in EI-oc, 8/38 in V-reg, 4/37 in V-oc, 4/35 in E-c (Mohammad et al., 2018). Our performance for V-reg is less than satisfactory because V-reg measures sentiment intensity without regards for directionality, whereas our source task takes into account directionality. This supports findings by Mou et al. that pre-training is less useful in a semantically different transfer.

System To evaluate our system, we assess the performance of each Component and various combinations of them. Table 2 shows the development set performance. In particular, we note that Component A+B performs better than Component A or Component B separately, as with Component C+D. Furthermore, Component A+B+C+D perform better overall compared to Component A+B and Component C+D.

Cluster training Subtask E-c classes are imbalanced, with 95% of "Trust" and "Surprise" training examples being negative. Table 3 shows the breakdown of negative and positive training examples for each of the E-c classes.

To assess our cluster training procedure, we evaluate the performance of independently training each of the E-c emotion classes (using a fresh copy of the pre-trained system for each of the 11 emotions) as well as with various class weighing schemes. Table 4 shows our experiment results.

Within independent training experiments, squared inverse weights performed best as measured by accuracy and micro-avg F1. Using squared inverse weights, cluster training performs better than independent training, attesting to the utility of cluster training.

6 Conclusion

In this paper, we present the first attempt to perform transfer learning from sentiment to emotions. Model weights are pre-trained with past SemEval sentiment categorization tasks and the penultimate layers of the models are concatenated into a single vector as input to new dense layers. The entire system is then trained for each subtask with the weights unfrozen.

Our deep learning system combines multi-dimensional word embeddings with single dimensional lexicon-based features. Specifically we combine features of $X \in \mathbb{R}^{50\times300}, \mathbb{R}^{50\times43}, \mathbb{R}^{1\times43}$, which results in better performance than systems using just one of the features.

For the E-c subtask, we utilize hierarchical clustering to group correlated emotions together and train the same model incrementally for emotions within the same cluster. This novel method outperforms a system which trains on each emotion independently.

Experiment	Anger	Sadness	Fear	Joy
Component A	0.729	0.706	0.677	0.719
Component B	0.710	0.719	0.646	0.689
Component C	0.584	0.561	0.567	0.564
Component D	0.602	0.597	0.546	0.585
Component A+B	0.722	0.714	0.701	0.732
Component C+D	0.659	0.619	0.631	0.601
Component A+B+C+D	**0.761**	**0.720**	**0.723**	**0.745**

Table 2: Results on development set comparing Component experiments.

Emotion	Percentage negative	Percentage positive
Anger	63%	37%
Disgust	62%	38%
Sadness	71%	29%
Pessimism	88%	12%
Fear	82%	18%
Joy	64%	36%
Optimism	71%	29%
Love	90%	10%
Anticipation	86%	14%
Surprise	95%	5%
Trust	95%	5%

Table 3: E-c training set, breakdown of percentage of positive and negative examples.

Experiment	Accuracy	Micro-avg F1	Macro-avg F1
Cluster training, squared inverse weights	**0.576**	**0.695**	0.522
Independent training, equal weights	0.513	0.639	0.439
Independent training, inverse weights	0.447	0.586	**0.545**
Independent training, squared inverse weights	0.558	0.684	0.527

Table 4: E-c experiment results on development set.

We participated in all of the English subtasks of SemEval 2018 Task 1: Affect in Tweets and obtained top 4 in 4 out of the 5 subtasks, testifying to our model robustness.

For future work, we would like to experiment with other training methods such as multi-task learning and distant supervision, as well as tune the hyper-parameters of our model to augment its performance across all subtasks (Mohammad and Kiritchenko, 2018).

Acknowledgments

This project was kindly financed by psyML Inc., a company co-founded by Galen Buckwalter and Dave Herman to bring psychology and AI together to solve real world problems. We thank Sebastian Ruder for his paper suggestions and the task organizers of SemEval 2018 Task 1 for the data and gracious support.

References

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. Association for Computational Linguistics.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 440–447. Association for Computational Linguistics.

John Blitzer, Ryan McDonald, and Fernando Pereira. 2006. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 120–128. Association for Computational Linguistics.

Mathieu Cliche. 2017. Bb_twtr at semeval-2017 task 4: Twitter sentiment analysis with cnns and lstms. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 573–580. Association for Computational Linguistics.

Venkatesh Duppada and Sushant Hiray. 2017. Seernet at emoint-2017: Tweet emotion intensity estimator. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 205–211. Association for Computational Linguistics.

Pranav Goel, Devang Kulshreshtha, Prayas Jain, and Kaushal Kumar Shukla. 2017. Prayas at emoint 2017: An ensemble of deep neural architectures for emotion intensity prediction in tweets. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 58–65. Association for Computational Linguistics.

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

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

Maximilian Köper, Evgeny Kim, and Roman Klinger. 2017. Ims at emoint-2017: Emotion intensity prediction with affective norms, automatically extended resources and deep learning. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 50–57. Association for Computational Linguistics.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1003–1011. Association for Computational Linguistics.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. In *Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 34–49. Association for Computational Linguistics.

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)*.

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 (LREC-2018)*.

Lili Mou, Zhao Meng, Rui Yan, Ge Li, Yan Xu, Lu Zhang, and Zhi Jin. 2016. How transferable are neural networks in nlp applications? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 479–489. 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. Association for Computational Linguistics.

Ali Sharif Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson. 2014. Cnn features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 806–813.

Jonathon Read. 2005. Using emoticons to reduce dependency in machine learning techniques for sentiment classification. In *Proceedings of the ACL Student Research Workshop*, pages 43–48. Association for Computational Linguistics.

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

Mickael Rouvier. 2017. Lia at semeval-2017 task 4: An ensemble of neural networks for sentiment classification. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 760–765. Association for Computational Linguistics.

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.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. 2014. How transferable are features in deep neural networks? *Advances in Neural Information Processing Systems*, 27:3320–3328.