DeepMiner at SemEval-2018 Task 1: Emotion Intensity Recognition Using Deep Representation Learning

Habibeh Naderi
Dalhousie University, Canada
habibeh.naderi@dal.ca

Behrouz H. Soleimani
Dalhousie University, Canada
behrouz.hajisoleimani@dal.ca

Svetlana Kiritchenko, Saif M. Mohammad
National Research Council Canada
{svetlana.kiritchenko, saif.mohammad}@nrc-cnrc.gc.ca

Stan Matwin
Dalhousie University, Canada
stan@cs.dal.ca

Abstract

In this paper, we propose a regression system to infer the emotion intensity of a tweet. We develop a multi-aspect feature learning mechanism to capture the most discriminative semantic features of a tweet as well as the emotion information conveyed by each word in it. We combine six types of feature groups: (1) a tweet representation learned by an LSTM deep neural network on the training data, (2) a tweet representation learned by an LSTM network on a large corpus of tweets that contain emotion words (a distant supervision corpus), (3) word embeddings trained on the distant supervision corpus and averaged over all words in a tweet, (4) word and character n-grams, (5) features derived from various sentiment and emotion lexicons, and (6) other hand-crafted features. As part of the word embedding training, we also learn the distributed representations of multi-word expressions (MWEs) and negated forms of words. An SVR regressor is then trained over the full set of features. We evaluate the effectiveness of our ensemble feature sets on the SemEval-2018 Task 1 datasets and achieve a Pearson correlation of 72% on the task of tweet emotion intensity prediction.

1 Introduction

The widespread use of micro-blogging and social networking websites such as Twitter for conveying information, sharing opinions, and expressing feelings, makes the sentiment analysis of tweets an attractive area of research. However, sentiment analysis is challenging because people often convey their emotions indirectly and creatively, rather than explicitly stating how they feel. Sentiment analysis of tweets is additionally challenging because of the frequent occurrences of non-standard language and poor grammatical structure. Tweets also often contain misspellings, abbreviations, hashtags, and emoticons.

Various machine learning approaches have been developed for Twitter sentiment classification. Most of these algorithms train a classifier over tweets with manually annotated sentiment intensity labels and learn the most discriminative features. Hence, designing an effective feature engineering algorithm can improve classification performance, greatly. Mohammad et al. (2013; 2014) used many different sentiment lexicons (manually created and automatically generated), as well as a variety of hand-crafted features to build the top-ranked system for Twitter sentiment classification tasks in SemEval-2013 and SemEval-2014. Sentiment lexicons, either hand-crafted or algorithmically generated, consist of words and their associated polarity scores. However, since feature engineering is labour intensive and usually needs domain-specific knowledge, sentiment classification algorithms with less dependency on feature engineering are attracting considerable interest.

Socher et al. (2013) proposed a feature learning algorithm to discover explanatory factors in sentiment classification. They consider the representation of a sentence (or document) as a composition of the representations of its constituent words or phrases. This way, the sentiment classification problem reduces to learning an effective word representation (or word embedding) that not only models the syntactic context of words but also captures sentiment information of the sentence. Tang et al. (2014) extended the traditional word embedding methods (Mikolov et al., 2013b; Collobert et al., 2011) by encoding sentiment information into the existing continuous representation of words. They built sentiment-specific word embedding (SSWE) by developing three neural networks wherein the sentiment polarity of the tweet is incorporated in the neural networks' loss functions. Teng et al. (2016) proposed a context-sensitive lexicon-based method using recurrent and simple

Emotion	Train	Dev.	Test	Total
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
Total	7,102	1,464	4,068	12,634

Table 1: Number of instances provided in the Tweet Emotion Intensity dataset (SemEval-2018 Task 1, EI-reg English). The data was divided into train, development, and test sets.

feed-forward neural networks to extract sentiment lexicons and produce a new polarity weight, respectively.

Unlike lexicon-based sentiment analysis, deep learning approaches are effective in exploring both linguistic and semantic relations between words (Liu et al., 2015). However, due to the limited amount of high-quality labeled data, it is difficult to train deep models with a large number of hyper-parameters for sentiment analysis tasks. Additionally, manual labeling of data is costly and requires domain expert knowledge, which is not always available.

In this paper, we describe two systems: System I, our official submission to the competition, and System II, our best model. In both systems, we combine deep learning and lexicon-based approaches to extract the most informative semantic and emotion representations of tweets. We train two LSTM models, one on the provided training data and another one on a large corpus of tweets that contain emotion words, to obtain emotion-specific tweet representations. We augment this feature space with word and character n-grams, features derived from several sentiment and emotion lexicons as well as other hand-crafted features. Our best model achieves an average Pearson correlation of 71.96% on the official EI-reg test dataset.

2 Data

The English training, development, and test datasets used in our experiments were provided as part of the SemEval-2018 Task 1, EI-reg subtask (Mohammad et al., 2018).[1] The data files include tweet id, tweet text, emotion of the tweet, and the emotion intensity. An overview of the data is provided in Table 1.

[1]A detailed description of the English datasets and the analysis of various affect dimensions is available in Mohammad and Kiritchenko (2018).

2.1 Data preparation

The following pre-processing steps were applied to each of the training and test tweets:

- Remove URLs and usernames.
- Lower-case all the tweet text.
- Substitute abbreviated phrases such as *I've, don't, I'd, etc.* with their long forms.
- Replace tweet-specific acronyms such as *gr8, lol, rotfl, etc.* with their expanded forms.
- Substitute the elongated words with the same words but keeping at most two consecutive occurrences of repeated letters.
- Standardize all the emojis in data to their explanatory phrases using emoji Python package[2].
- Remove all the HTML character codes.
- Replace all occurrences of a multi-word expression (MWE) by a unique identifier. We use WikiMWE (Hartmann et al., 2012), which contains all multi-word expressions from Wikipedia.
- Generate the negated form of all the tokens that occur between any of the negation words, such as *no, not, never, etc.*, and a punctuation mark.
- Remove special characters, numbers, non-English words or phrases.
- Normalize all adjectives and adverbs in test data that do not exist in train or development data sets with adjective or adverb in the training data which shares the most common Synsets of WordNet with it (if we find more than one candidate, we replace the adjective with the most frequent one in the training data).
- Applying WordNet lemmatizer to have the simple singular form of tokens with part-of-speech tags of adjective, adverb, verb or noun.

The tweets are now fed to the system.

[2]https://pypi.org/project/emoji/

3 System Description

We created two models: System I, our official submission to the competition, and System II, our best model. Both models address the task of emotion intensity prediction (EI-reg): given a tweet T and an emotion e, predict a real-valued intensity score (in the range $[0, 1]$) of e that represents the emotional state of the author of the tweet T.

3.1 System I

Our first model takes advantage of both embedding-based and lexicon-based features. In particular, the following feature sets are generated:

- Embedding-based features:
  - Average word embedding vector;
  - Representation of a tweet learned by an LSTM neural network on the provided training data;
- Lexicon-based and n-gram features:
  - Word and character n-gram features;
  - Vector of 43 lexicon-derived features, compiled using the AffectiveTweets package (Mohammad and Bravo-Marquez, 2017).[3] The lexicons used include those created by Nielsen (2011); Mohammad and Turney (2013); Kiritchenko et al. (2014); Hu and Liu (2004); Bravo-Marquez et al. (2016); Thelwall et al. (2012); Wilson et al. (2005).

We use bag-of-word (BOW) (Pang et al., 2002) and term frequency-inverse document frequency (tf-idf) methods to extract different word and character n-grams. We train word embeddings on a large corpus of tweets that contain emotion words. Then, we refine our learned word embeddings to build emotion-specific word embeddings for every emotion. Specifically, we assign emotion-specific weights to every word in our learned word embeddings and multiply each word vector by weights. These emotion-specific weights are obtained by calculating the Pearson correlation between the extracted unigram features and intensity labels of the training and development datasets of each emotion.

[3] https://affectivetweets.cms.waikato.ac.nz/

We concatenate two learned embedding-based tweet representations, word and character n-grams, and the lexicon features in a multimodal feature layer. We train a Random Forest (RF) over this heterogeneous multimodal feature layer to predict emotion intensity of a tweet.

This approach was evaluated on the datasets of SemEval-2018 Task 1, EI-reg (an emotion intensity regression task) and EI-oc (an emotion intensity ordinal classification task), for which it obtained Pearson correlations of 57.5% and 48.5% on the test sets, respectively.

Further investigation revealed that our system I was overfitted to the training data and lost its generalization ability over new unseen data. The cause of this problem was the use of development dataset labels in our feature engineering algorithm. So, we modify our model to overcome overfitting and propose system II.

3.2 System II

Similarly to System I, our second model incorporates both embedding-based representations and linguistic knowledge in a unified architecture (see Figure 1). We train a Support Vector Regressor (SVR) over the following two categories of features:

- Embedding-based features:
  - Average word embedding vector;
  - Representation of a tweet learned by an LSTM neural network on the provided training data;
  - Emotion-polarized representation of a tweet learned by an LSTM neural network on a distant supervision corpus;
- Lexicon-based and hand-crafted features:
  - Word and character n-gram features;
  - Vector of 43 lexicon-derived features, compiled using the AffectiveTweets package (Mohammad and Bravo-Marquez, 2017);
  - Hand-crafted features based on either word similarities in learned word embeddings or emotion intensity similarities in accordance to train and development labels.

Below, different components of the two systems are explained in detail.

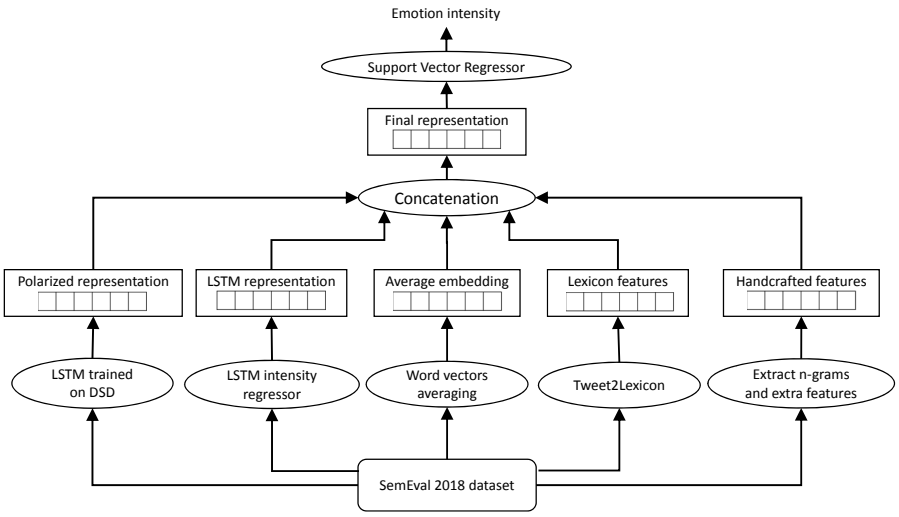

Figure 1: The architecture of our model (System II).

3.3 Word embedding layer

Since the input of our model is a sequence of tokens $\{w_1, w_2, ..., w_n\}$, it is crucial to learn an effective word representation for automatic emotion analysis. A word embedding is a dense, low-dimensional and real-valued vector associated with each word w_i. We used word2vec (Mikolov et al., 2013a) and SVD-NS (Soleimani and Matwin, 2018) to learn word embeddings and trained it on an unlabeled corpus of 21M tweets provided as part of the SemEval-2018 Affect in Tweets DIstant Supervision Corpus (SemEval-2018 AIT DISC) (Mohammad et al., 2018). SVD-NS works better for word and sentence similarity tasks and is much faster than word2vec. Such distributed word representations learned from massive text data make feature engineering less dependent on the task. However, unsupervised learning of word embeddings cannot thoroughly capture finer context-based semantic information of a specific task. Hence, to incorporate linguistic structure of tweets, we use the following two techniques to improve the word vectors:

1. Our model learns a unique distributed representation for every Multi-Word Expression (MWE). MWEs occur frequently in tweets, and their meanings are often not derivable from the meanings of the constituent words.

2. Additionally, our model learns an embedding vector to represent the negated form of every word occurring between a negation word (e.g., *no*, *shouldn't*) and the following punctuation mark. Due to the significant impact of negation words in changing the sentiment polarity of a sentence, we treat the negated tokens differently (Zhu et al., 2014; Kiritchenko et al., 2014). By adding a 'NEG' prefix to them, we consider the negated tokens as different entities and learn separate word representations for them.

To learn word embeddings, we applied two methods: the continuous skip-gram model (Mikolov et al., 2013a) with the window size of 5, and SVD-NS (Soleimani and Matwin, 2018) with the PMI threshold of $\alpha = -2.5$. The vector dimensionality was set to $d = 100$. We also filter words that occur less than 3 times in the corpus.

3.4 Average embedding layer

To capture the global context of a tweet, we build a tweet embedding by vertically concatenating embedding vectors of its n words. This yields a tweet embedding matrix $X \in R^{n \times d}$. Then, we take the mean of these word embeddings across the tweet length. Therefore, an average embedding will add d features (equal to the number of our word embedding dimensions) to our multimodal feature layer.

3.5 Tweet embedding vector learned by LSTM layer

To learn a semantic representation of a tweet, we use an LSTM neural network, which we found effective in detecting salient words of a sentence while automatically attenuating unimportant

words. The LSTM model sequentially takes each word in a sentence, extracts its information, and embeds it into a semantic vector. Due to its ability to capture long-term memory, LSTM accumulates increasingly richer information as it goes through the sentence, and when it reaches the last word, the hidden layer of the network provides a semantic representation of the whole sentence (Palangi et al., 2016). To be able to train sequential neural networks in batches, we normalize tweet length by zero padding and then feed the zero-padded tweet embedding matrix to an LSTM layer. We apply dropout (Srivastava et al., 2014) on the LSTM layer to prevent network parameters from overfitting and control the co-adaptation of features. Our LSTM layer is then followed by two fully connected hidden layers, and one output layer. Each of these layers computes the transformation $f(W_i * x_i + b_i)$ for $i = \{1, 2, 3\}$, where W is the weight matrix, b is the bias vector and f is a *Relu* non-linear activation function for hidden layers and a *Sigmoid* neuron for output layer. The full network is trained on the provided training data to predict the intensity score of the input tweet. We consider the representation obtained from the first hidden layer as a sentence embedding vector of an input tweet.

The network parameters are learned by minimizing the mean squared error (MSE) between the actual and predicted values of emotion intensity on the training data. We optimize this loss function by back-propagating through layers via mini-batch gradient descent, with batch size of 32, 40 training epochs, and Adam optimization algorithm (Kingma and Ba, 2014) with learning rate of $\alpha = 0.001$. We use one LSTM layer with 64 neurons, followed by a dropout of 0.2, and two hidden layers of sizes 32 and 16, respectively. We use the same network parameters for an LSTM model trained on the distant supervision data (see Section 3.6).

3.6 Emotion-polarized tweet representation learned by LSTM

We leverage large amount of Twitter data with distant supervision to polarize our word embeddings for each emotion. Hence, we use SemEval-2018 AIT DISC distant supervision corpus of tweets released by the competition organizers, which includes around 100M English tweet ids associated with tweets that contain emotion-related query terms such as '*#angry*', '*annoyed*', '*panic*', '*happy*', etc. We collected 21M tweets by polling the Twitter API with these tweet ids. Based on the query terms, one or more emotion labels of {'anger', 'fear', 'joy', 'sadness'} have been assigned to every tweet in this dataset. For each emotion, we randomly select $200,000$ tweets labeled with that emotion (e.g., 'anger') and $200,000$ tweets labeled with other emotions ('not anger') to build the emotion-specific word embeddings. Since the four basic emotions are not independent and may be correlated, we build these emotion-polarized word embeddings in two ways: *(i)* one against all strategy: for example, 'not anger' tweets are selected from tweets labeled with any of the other three emotions, i.e., 'fear', 'joy', or 'sadness'; *(ii)* considering emotions with similar valence as one group of labels: tweets labeled with 'anger', 'fear', and 'sadness' are treated as they have the same label. So, here 'not anger' tweets are selected from tweets that are labeled only as 'joy'. Then, we train an LSTM neural network using these emotion-specific word embeddings to build emotion-specific representations of tweets. Our final emotion-specific tweet representation obtained by concatenating two hidden state layers learned by the same LSTM neural network trained twice on the same data but with different emotion labeling according to the above two labeling strategies.

3.7 Hand-Crafted Features

In addition to the two kinds of tweet representations described above, we use bag-of-word (BOW) representation to extract most and least frequent word n-grams (unigrams, bigrams, and trigrams) as well as character n-grams (three, four, and five consecutive characters) from the training, development, and test datasets. BOW represents each word as a one-hot vector which has the same length as the size of the vocabulary, and only one dimension is 1, with all others being 0. However, the one-hot word representation cannot sufficiently capture the complex linguistic characteristics of words. We augment our feature space by generating additional hand-crafted features. We define a set of binary features by adding n adjectives with highest and lowest intensities for each emotion according to the emotion's training data. The intensity of a word (unigram) is obtained as an average emotion intensity of tweets in the train-

	Experiment	Anger	Fear	Joy	Sadness	Average
System I (EI-reg)	WE + TE + lex	58.15	57.06	57.51	57.36	57.54
System I (EI-oc)	WE + TE + lex	49.07	41.09	55.62	48.45	48.56
	WE	62.37	60.07	56.46	60.69	60.12
	WE + MWEs	63.26	61.55	57.91	61.89	61.15
	WE + MWEs + negation	62.80	62.66	58.21	63.32	61.75
	ngram	48.30	52.65	52.44	52.01	51.35
	polTE	30.12	36.81	33.95	48.86	37.44
	TE	68.91	68.93	69.21	70.14	69.30
System II (EI-reg)	WE + lex	67.74	68.74	66.20	67.21	67.48
	WE + ngram	63.70	66.38	60.87	64.55	63.87
	WE + ngram + lex	66.99	70.27	67.46	67.67	68.10
	WE + lex + handcrafted	68.82	71.63	67.74	69.27	69.37
	WE + ngram + TE	69.69	69.54	68.49	69.66	69.35
	WE + ngram + TE + lex	69.98	**73.44**	69.14	**73.33**	71.47
	all features	**72.30**	70.46	**71.55**	73.13	**71.96**

Table 2: Pearson correlation (r) % obtained on the test sets. The highest score in each emotion is shown in bold. System I indicates the results of our first overfitted model and System II shows the results of our modified model. In every experiment on system II, we train SVR regressor with linear kernel to predict emotion intensity of a tweet while in system I experiments, we use RF regressor and SVM classifier for SemEval-2018 Task 1 and 2, respectively. The all-features experiment represents the model built on concatenation of all six groups of features including WE, ngram, TE, polTE, lex, and handcrft.

ing data that contain that unigram. We also add the weighted average intensity of all extracted unigrams and the intensity of their k nearest neighbors in learned word embeddings (sorted based on cosine similarity) to our feature set.

4 Results

We train the SVR regressor on the combined set of tweets in the training and development sets and apply the model on the test set. The Pearson correlation between the predictions and the gold labels was used by the competition organizers as the official evaluation measure. The percentage of Pearson correlation scores obtained by all of our individual and combined models on the test set are shown in Table 2. To make the result table easier to understand, we shortened the feature groups' names as follows: 1) average word embedding vectors $\rightarrow$ WE, 2) tweet embedding vectors learned by LSTM $\rightarrow$ TE, 3) emotion-polarized tweet embeddings learned by LSTM $\rightarrow$ polTE, 4) word and character n-gram features $\rightarrow$ ngram, 5) AffectiveTweets lexicon features $\rightarrow$ lex, 6) hand-crafted features based on word similarities in emotion intensity $\rightarrow$ handcrft. All the results reported in the table use word embeddings that are obtained by SVD-NS (Soleimani and Matwin, 2018) method which was slightly better than word2vec (Mikolov et al., 2013b).

The 'all-features' row shows the results obtained by the model that concatenates all six groups of features including WE, ngram, TE, polTE, lex, and handcrft. This model achieves the highest Pearson correlation score among all of our proposed models. The tweet representation learned by LSTM is the best learned unimodal feature. Considering MWEs as independent semantic units improves the average embedding model's performance by 1.03 percentage points. Learning independent embedding vectors for negated form of words further improves the score by 0.6 percentage points.

5 Conclusion

We described a deep learning framework to predict emotion intensity in tweets. We implemented an ensemble of embedding-based feature representations and sentiment lexicon-based feature learning approaches. Our best model obtained a Pearson correlation of 71.96% on Task 1 of SemEval-2018 competition (EI-reg: an emotion intensity regression task). The tweet representation feature vector learned by LSTM was the most effective feature group amongst those that we used. Various sentiment and emotion lexicon features, our handcrafted features and word n-grams features also helped improve prediction quality.

Acknowledgments

We thank Xiang Jiang for helping us build attentive deep neural networks and fruitful discussions.

References

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*, pages 536–539. IEEE.

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.

Silvana Hartmann, György Szarvas, and Iryna Gurevych. 2012. Mining multiword terms from wikipedia. In *Semi-Automatic Ontology Development: Processes and Resources*, pages 226–258. IGI Global.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. 2014. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762.

Pengfei Liu, Shafiq Joty, and Helen Meng. 2015. Fine-grained opinion mining with recurrent neural networks and word embeddings. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1433–1443.

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

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

Saif M. Mohammad and Felipe Bravo-Marquez. 2017. Emotion intensities in tweets. In *Proceedings of the sixth joint conference on lexical and computational semantics (*Sem)*, Vancouver, Canada.

Saif M. Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. Semeval-2018 Task 1: Affect in tweets. In *Proceedings of International Workshop on Semantic Evaluation (SemEval-2018)*, New Orleans, LA, USA.

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

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. In *Proceedings of the International Workshop on Semantic Evaluation*, Atlanta, GA, USA.

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

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. In *Workshop on'Making Sense of Microposts: Big things come in small packages*, pages 93–98.

Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 24(4):694–707.

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

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Behrouz H Soleimani and Stan Matwin. 2018. Spectral word embedding with negative sampling. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*.

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.

Zhiyang Teng, Duy Tin Vo, and Yue Zhang. 2016. Context-sensitive lexicon features for neural sentiment analysis. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1629–1638.

Mike Thelwall, Kevan Buckley, and Georgios Paltoglou. 2012. Sentiment strength detection for the social web. *Journal of the Association for Information Science and Technology*, 63(1):163–173.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354.

Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. 2014. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 304–313.