Amobee at SemEval-2018 Task 1: GRU Neural Network with a CNN Attention Mechanism for Sentiment Classification

Alon Rozental*, Daniel Fleischer*
Amobee, Tel Aviv, Israel
alon.rozental@amobee.com
daniel.fleischer@amobee.com

Abstract

This paper describes the participation of Amobee in the shared sentiment analysis task at SemEval 2018. We participated in all the English sub-tasks and the Spanish valence tasks. Our system consists of three parts: training task-specific word embeddings, training a model consisting of gated-recurrent-units (GRU) with a convolution neural network (CNN) attention mechanism and training stacking-based ensembles for each of the sub-tasks. Our algorithm reached 3rd and 1st places in the valence ordinal classification sub-tasks in English and Spanish, respectively.

1 Introduction

Sentiment analysis is a collection of methods and algorithms used to infer and measure affection expressed by a writer. The main motivation is enabling computers to better understand human language, particularly sentiment carried by the speaker. Among the popular sources of textual data for NLP is Twitter, a social network service where users communicate by posting short messages, no longer than 280 characters long—called tweets. Tweets can carry sentimental information when talking about events, public figures, brands or products. Unique linguistic features, such as the use of slang, emojis, misspelling and sarcasm, make Twitter a challenging source for NLP research, attracting the interest of both academia and the industry.

Semeval is a yearly event in which international teams of researchers work on tasks in a competition format where they tackle open research questions in the field of semantic analysis. We participated in Semeval 2018 task 1, which focuses on sentiment and emotions evaluation in tweets. There were three main problems: identifying the presence of a given emotion in a tweet (sub-tasks EI-reg, EI-oc), identifying the general sentiment (valence) in a tweet (sub-tasks V-reg, V-oc) and identifying which emotions are expressed in a tweet (sub-task E-c). For a complete description of Semeval 2018 task 1, see the official task description (Mohammad et al., 2018).

We developed an architecture based on gated-recurrent-units (GRU, Cho et al. (2014)). We used a bi-directional GRU layer, together with a convolutional neural network (CNN) attention-mechanism, where its input is the hidden states of the GRU layer; lastly there were two fully connected layers. We will refer to this architecture as the Amobee sentiment classifier (ASC). We used ASC to train word embeddings to incorporate sentiment information and to classify sentiment using annotated tweets. We participated in all the English sub-tasks and in the valence Spanish sub-tasks, achieving competitive results.

The paper is organized as follows: section 2 describes our data sources, section 3 describes the data pre-processing pipeline. A description of the main architecture is in section 4. Section 5 describes the word embeddings generation; section 6 describes the extraction of features. In section 7 we describe the performance of our models; finally, in section 8 we review and summarize the results.

2 Data Sources

We used four sources of data:

1. Twitter Firehose: we randomly sampled 200 million tweets using the Twitter Firehose service. They were used for training word embeddings and for distant supervision learning.
2. Semeval 2017 task 4 datasets of tweets, annotated according to their general sentiment

*These authors contributed equally to this work.

Proceedings of the 12th International Workshop on Semantic Evaluation (SemEval-2018), pages 218–225
New Orleans, Louisiana, June 5–6, 2018.

on 3 and 5 level scales; used to train the ASC model.

3. Annotated tweets from an external source[1], annotated on a 3-level scale; used to train the ASC model.
4. Official Semeval 2018 task 1 datasets: used to train task specific models.

Datasets of Semeval 2017 and the external source were combined with compression[2]; the resulting dataset contained 88,623 tweets with the following distribution: positive: 30097 sentences (34%), neutral: 35818 (40%), negative: 22708 (26%). Description of the official Semeval 2018 task 1 datasets can be found in Mohammad et al. (2018); Mohammad and Kiritchenko (2018).

3 Preprocessing

We started by defining a cleaning pipeline that produces two cleaned version of an original text; we refer to them as "simple" and "complex" versions. Both versions share the same initial cleaning steps:

1. Word tokenization using the CoreNLP library (Manning et al., 2014).
2. Parts of speech (POS) tagging using the Tweet NLP tagger, trained on Twitter data (Owoputi et al., 2013).
3. Grouping similar emojis and replacing them with representative keywords.
4. Regex: replacing URLs with a special keyword, removing duplications, breaking #CamelCasingHashtags into individual words.

The complex version contains these additional steps:

1. Word lemmatization, using CoreNLP.
2. Named entity recognition (NER) using CoreNLP and replacing the entities with representative keywords, e.g. `_date_`, `_number_`, `_brand_`, etc.
3. Synonym replacement, based on a manually-created dictionary.
4. Word replacement using a Wikipedia dictionary, created by crawling and extracting lists of places, brands and names.

[1] https://github.com/monkeylearn/sentiment-analysis-benchmark

[2] Transformed 5 labels to 3: $\{-2, -1\} \rightarrow \{-1\}$, $\{1, 2\} \rightarrow \{1\}$, $\{0\} \rightarrow \{0\}$.

As an example, table 1 shows a fictitious tweet and the results after the simple and complex cleaning stages.

4 ASC Architecture

Our main contribution is an RNN network, based on GRU units with a CNN-based attention mechanism; we will refer to it as the Amobee sentiment classifier (ASC). It is comprised of four identical sub-models, which differ by the input data each of them receives. Sub-model inputs are composed of word embeddings and embeddings of the POS tags—see section 5 for a description of our embedding procedure. The words were embedded in a 200 or 150 dimensional vector spaces and the POS tags were embedded in a 8 dimensional vector space. We pruned the tweets to have 40 words, padding shorter sentences with a zero vector. The embeddings form the input layer.

Next we describe the sub-model architecture; the embeddings were fed to a bi-directional GRU layer of dimension 200. Inspired by the attention mechanism introduced in Bahdanau et al. (2014), we extracted the hidden states of the GRU layer; each state corresponds to a decoded word in the GRU as it reads each tweet word by word. The hidden states were arranged in a matrix of dimension 40×400 for each tweet (bi-directionality of the GRU layer contributes a factor of 2). We fed the hidden states to a CNN layer, instead of a weighted sum as in the original paper. We used 6 filter sizes $[1, 2, 3, 4, 5, 6]$, with 100 filters for each size. After a max-pooling layer we concatenated all outputs, creating a 600 dimensional vector. Next was a fully connected layer of size 30 with $\tanh$ activation, and finally a fully connected layer of size 3 with a softmax activation function.

We defined 4 such sub-models with embedding inputs of the following settings: w2v-200, w2v-150, ft-200, ft-150 (ft=FastText, w2v=Word2Vec, see discussion in the next section). We combined the four sub-models by extracting their hidden $d = 30$ layer and concatenating them. Next we added a fully connected $d = 25$ layer with $\tanh$ activation and a final fully connected layer of size 3. See figure 1 for an illustration of the entire architecture. We used the AdaGrad optimizer (Duchi et al., 2011) and a cross-entropy loss function. We used the Keras library (Chollet et al., 2015) and the TensorFlow framework (Abadi et al., 2016).

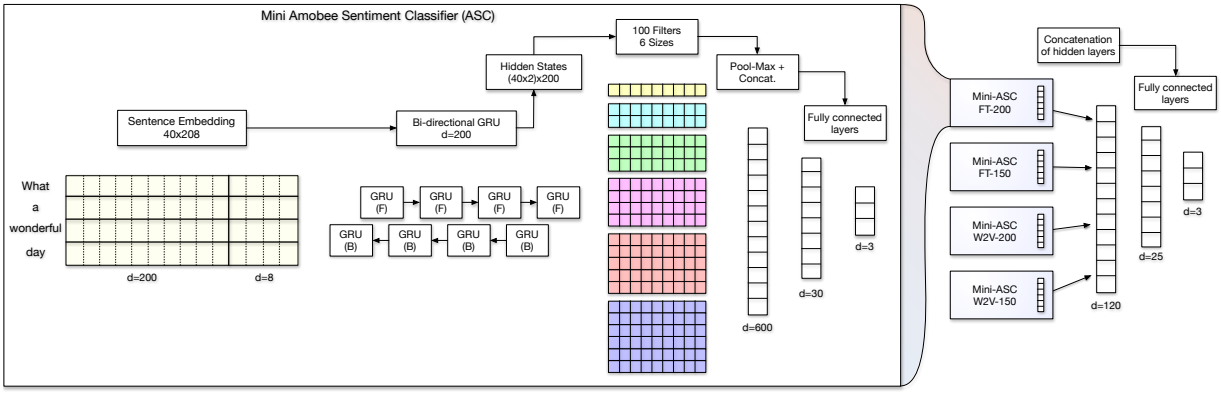

Figure 1: Architecture of the ASC nework. Each of the four sub-models on the right has the same structure as depicted in the central region.

Original	@USAIRWAYS is right :-) ! Flying in September #NiceToFly
Simple Cleaning	twitter-entity is right happy-smily ! flying in september nice to fly
Complex Cleaning	twitter-entity be right happy-smily ! fly in _date_ pleasant to fly

Table 1: An example of a tweet processing, producing two cleaned versions.

5 Embeddings Training

Word embedding is a family of techniques in which words are encoded as real-valued vectors of lower dimensionality. These word representations have been used successfully in sentiment analysis tasks in recent years. Among the popular algorithms are Word2Vec (Mikolov et al., 2013) and FastText (Bojanowski et al., 2016).

Word embeddings are useful representations of words and can uncover hidden relationships. However, one disadvantage they have is the typical lack of sentiment information. For example, the word vector "good" can be very close to the word vector "bad" in some trained, off-the-shelf word embeddings. Our goal was to train word embeddings based on Twitter data and then re-learn them so they will contain emotion-specific sentiment.

We started with our 200 million tweets dataset; we cleaned them using the pre-processing pipeline (described in section 3) and then trained generic embeddings using the Gensim package (Řehůřek and Sojka, 2010); we created four embeddings for the words and two embeddings for the POS tags: for each sentence we created a list of corresponding POS tags (there are 25 tags offered by the tagger we used); treating the tags as words, we trained $d = 8$ embeddings using the word2vec algorithm on the simple and complex cleaned datasets. The embeddings parameters are specified in table 2.

Following Tang et al. (2014); Cliche (2017), who explored training word embeddings for sentiment classification, we employed a similar approach. We created distant supervision datasets, first, by manually compiling 4 lists of representative words for each emotion: anger, fear, joy and sadness; then, we built two datasets for each emotion: the first containing tweets with the representative words and the second does not. Each list contained about 40 words and each dataset contained roughly 2 million tweets. We used the ASC sub-model architecture (section 4) to train as following: training for one epoch with embeddings set to be untrainable (fixed). Then train for 6 epochs where the embeddings can change.

Overall we trained 16 word embeddings—4 embedding configurations for each emotion. In addition, we decided to use the trained models' final hidden layer ($d = 15$) as a feature vector in the task-specific architectures; our motivation was using them as emotion and intensity classifiers via transfer learning.

	Algorithm	**Dimension**	**Dataset**
Words	Word2Vec	200	Simple
	Word2Vec	150	Complex
	FastText	200	Simple
	FastText	150	Complex
Tags	Word2Vec	200	Simple
	Word2Vec	150	Complex

Table 2: Parameters for the word and POS tag embeddings.

6 Features Description

In addition to our ASC models, we extracted semantic and syntactic features, based on domain knowledge:

- Number of magnifier and diminisher words, e.g. “incredibly”, “hardly” in each tweet.
- Logarithm of length of sentences.
- Existence of elongated words, e.g. “wowww”.
- Fully capitalized words.
- The symbols #,@ appearing in the sentence.
- Predictions of external packages: Vader (part of the NLTK library, Hutto and Gilbert, 2014) and TextBlob (Loria et al., 2014).

Additionally, we compiled a list of 338 emojis and words in 16 categories of emotion, annotated with scores from the set $\{0.5, 1, 1.5, 2\}$. For each sentence, we summed up the scores in each category, up to a maximum value of 5, generating 16 features. The categories are: anger, disappointed, fear, hopeful, joy, lonely, love, negative, neutral, positive, sadness and surprise. Finally, we used the NRC Affect Intensity lexicon (Mohammad, 2017) containing 5814 entries; each entry is a word with a score between 0 and 1 for a given emotion out of the following: anger, fear, joy and sadness. We used the lexicon to produce 4 emotion features from hashtags in the tweets; each feature contained the largest score of all the hashtags in the tweet. For a summary of all features used, see table 6 in the appendix.

7 Experiments

Our general workflow for the tasks is as follows: for each sub-task, we started by cleaning the datasets, obtaining two cleaned versions. We ran a pipeline that produced all the features we designed: the ASC predictions and the features described in section 6. We removed sparse features (less than 8 samples). Next, we defined a shallow neural network with a soft-voting ensemble. We chose the best features and meta-parameters—such as learning rate, batch size and number of epochs—based on the dev dataset. Finally, we generated predictions for the regression tasks. For the classification tasks, we used a grid search method on the regression predictions to optimize the loss. Most model trainings were conducted on a local machine equipped with a Nvidia GTX 1080 Ti GPU. Our official results are summarized in table 3.

Task	Metric	Score	Ranking
V-oc-Spanish	Pearson	0.765	1/14
V-reg-Spanish		0.770	2/14
V-oc		0.813	3/37
EI-oc Average		0.646	4/39
V-reg		0.843	5/38
E-c	Jaccard	0.566	6/35
EI-reg Average	Pearson	0.721	13/48

Table 3: Summary of results.

7.1 Valence Prediction

In the valence sub-tasks, we identified how intense a general sentiment (valence) is; the score is either in a continuous scale between 0 and 1 or classified into 7 ordinal classes $\{-3, -2, -1, 0, 1, 2, 3\}$, and is evaluated using the Pearson correlation coefficient.

We started with the regression task and defined the following model: first, we normalized the features to have zero mean and SD $= 1$. Then, we inserted 300 instances of fully connected layers of size 3, with a softmax activation and no bias term. For each copy, we applied the function $f(x) = (x_0 - x_2)/2 + 0.5$ where x_0, x_2 are the 1st and 3rd component of each hidden layer. Our aim was transforming the label predictions of the ASCs (trained on 3-label based sentiment annotation) into a regression score such that high certainty in either label (negative, neutral or positive) would produce scores close to 0, 0.5 or 1, respectively. Finally, we calculated the mean of all 300 prediction to get the final node; this is also known as a soft-voting ensemble. We used the Adam optimizer (Kingma and Ba, 2014) with default values, mean-square-error loss function, batch size of 400 and 65 epochs of training. For an illustration of the network, see figure 2. We experimented with the dev dataset, testing different subsets of the features. Finally we produced predictions for the regression sub-task V-reg.

We analyzed the relative contribution of each feature by measuring variable importance using Pratt (1987) approach. We calculated scores d_i for each feature using the following formula: $d_i = \hat{\beta}_i \, \hat{\rho}_i / R^2$ where $\hat{\beta}_i$ denotes the sample estimation

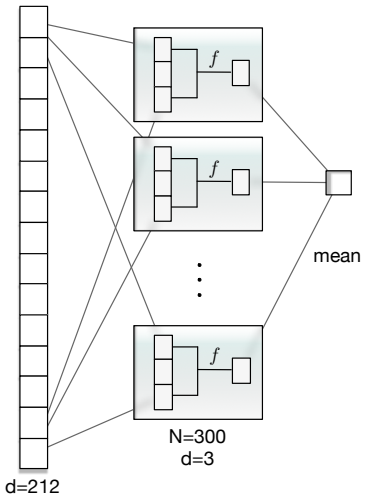

Figure 2: Architecture of the final classifier in the valence sub-tasks, where $f = (x_0 - x_2)/2 + 0.5$ and the input dimension is 212 for the V-reg sub-task.

of the feature, $\hat{\rho}_i$ is the simple correlation between the labels and the ith feature and R^2 is the coefficient of determination (see Thomas et al. 1998). We present the relative contribution of each feature in figure 3 and the top 10 features in table 4. We can see that the ASC models, both general and emotion-specific, contributed about 72% of the total contribution made by all features, in this sub-task.

For the ordinal classification task, we used the predictions of the regression task on the sentences, which were the same in both tasks. Using a grid search method, we partitioned the regression scores into 7 categories such that the Pearson correlation coefficient was maximized. We submitted the classes predictions as sub-task V-oc. Our final scores were 0.843, 0.813 in the regression and classification sub-tasks, respectively.

Name	Dim.	%
ASC_anger	25	31.38%
ASC	25	18.92%
ASC_fear	25	10.63%
ASC_joy	25	8.13%
W2V_200_sadness	15	7.10%
W2V_200_fear	15	3.82%
ASC_sadness	25	3.46%
W2V_200_joy	15	1.74%
Blob	1	1.64%
Joy	1	1.60%

Table 4: Relative contribution of features in the valence regression sub-task.

EI-reg	Anger	Fear	Joy	Sadness
Features	204	274	150	181
Learning rate	10^{-4}	10^{-5}	10^{-5}	$3 \cdot 10^{-5}$
Epochs	330	700	700	1000

Table 5: Summary of training parameters for the emotion intensity regression tasks.

7.2 Emotion Intensity

In the emotion intensity sub-tasks, we identified how intense a given emotion is in the given tweets. The four emotions were: anger, fear, joy and sadness; the score is either in a scale between 0 and 1 or classified into 4 ordinal classes $\{0, 1, 2, 3\}$. Performance was evaluated using the Pearson correlation coefficient. Our approach was similar to the valence tasks; first we generated features, then we used the same architecture as in the valence sub-tasks, depicted in figure 2. However, in these sub-tasks we used the emotion-specific embeddings for each emotion sub-task. We generated regression predictions and submitted them as the EI-reg sub-tasks; finally we carried a grid search for the best partition, maximizing the Pearson correlation and submitted the classes predictions as sub-tasks EI-oc. For a summary of the training parameters used in the regression sub-tasks, see table 5.

Our system performed as following: in the regression tasks, the scores were: 0.748, 0.670, 0.748, 0.721 for the anger, fear, joy and sadness, respectively, with a macro-average of 0.721. In the classification tasks, the scores were: 0.667, 0.536, 0.705, 0.673 for the anger, fear, joy and sadness, respectively, with a macro-average of 0.646.

7.3 Multi-label Classification

In the multi-label classification sub-task, we had to label tweets with respect to 11 emotions: anger, anticipation, disgust, fear, joy, love, optimism, pessimism, sadness, surprise and trust. The score was evaluated using the Jaccard similarity coefficient. We started with the same cleaning and feature-generation pipelines as before, creating an input layer of size 217. We added a fully connected layer of size 100 with $\tanh$ activation. Next there were 300 instances of fully connected layers of size 11 with sigmoid activation function. We calculated the mean of all $d = 11$ vectors, producing the final $d = 11$ vector. For an illustration, see figure 4 for an illustration. We used

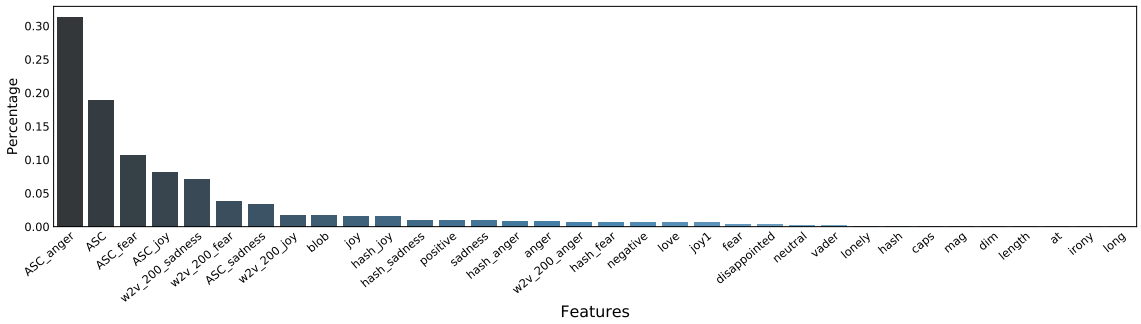

Figure 3: Relative contribution of features in the valence regression sub-task.

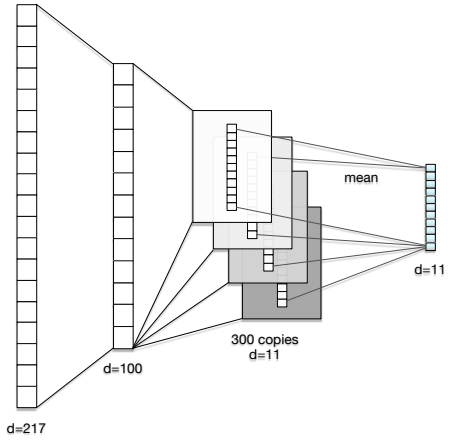

Figure 4: Architecture of the multi-label sub-task E-c.

the following loss function, based on Tanimoto distance: $L(y, \tilde{y}) = 1 - \frac{y \cdot \tilde{y}}{\|y+\tilde{y}\|_1 - y \cdot \tilde{y} + \epsilon}$, where $\|\cdot\|_1$ is an L^1 norm and $\epsilon = 10^{-7}$ is used for numerical stability. We trained with a batch size of 10, for 40 epochs with Adam optimization with default parameters. Our final score was 0.566.

7.4 Spanish Valence Tasks

We participated in the Spanish valence tasks to examine the current state of neural machine translation (NMT) algorithms. We used the Google Cloud Translation API to translate the Spanish training, development and test datasets for the two valence tasks from Spanish to English. We then treated the tasks the same way as the English valence tasks, using the same cleaning and feature extraction pipelines and the same architecture described in section 7.1 to generate regression and classification predictions. We reached 1st and 2nd places in the classification and regression sub-tasks, with scores of 0.765, 0.770, respectively.

8 Review and Conclusions

In this paper we described the system developed to participate in the Semeval 2018 task 1 workshop. We reached 3rd place in the valence ordinal classification sub-task and 5th place in the valence regression sub-task. In the Spanish valence tasks, we reached 1st and 2nd places in the classification and regression sub-tasks, respectively. In the emotions intensity sub-tasks we reached 4th and 13th places in the classification and regression sub-tasks, respectively.

Summarizing the methods used: training of word embeddings based on a Twitter corpus (200M tweets), developing and using Amobee sentiment classifier (ASC) architecture—a bi-directional GRU layer with a CNN-based attention mechanism and an additional hidden layer—used to adjust the embeddings to include emotional context, and finally a shallow feed-forward NN with a stack-based ensemble of final hidden layers from all previous classifiers we trained. This form of transfer learning proved to be important, as the hidden layers features achieved a significant contribution to minimizing the loss.

Overall, we had better performance in the valence tasks, both in English and Spanish. We posit this is due to the fact our annotated supervised training dataset (non task-specific) was based on Semeval 2017 task 4, which focused on valence classification. In addition, the annotations in Semeval 2017 were label-based, lending themselves more easily to the ordinal classification tasks. In the Spanish tasks, we used external translation (Google API) and achieved good results without the use of Spanish-specific features.

Acknowledgment

We thank Zohar Kelrich for assisting in translating the Spanish datasets to English.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. 2016. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259.

François Chollet et al. 2015. Keras. `https://github.com/keras-team/keras`.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

C.J. Hutto and E.E. Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth International Conference on Weblogs and Social Media (ICWSM-14)*, Ann Arbor, MI.

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

Steven Loria, P Keen, M Honnibal, R Yankovsky, D Karesh, E Dempsey, et al. 2014. Textblob: simplified text processing. *Secondary TextBlob: Simplified Text Processing*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Saif M Mohammad. 2017. Word affect intensities. *arXiv preprint arXiv:1704.08798*.

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*, Miyazaki, Japan.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics.

John W Pratt. 1987. Dividing the indivisible: Using simple symmetry to partition variance explained. In *Proceedings of the second international Tampere conference in statistics, 1987*, pages 245–260. Department of Mathematical Sciences, University of Tampere.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

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.

D Roland Thomas, Edward Hughes, and Bruno D Zumbo. 1998. On variable importance in linear regression. *Social Indicators Research*, 45(1-3):253–275.

A Features List

List of features used as inputs for the task-specific models.

Name	Description	Dim.
ASC	ASC model hidden layer.	25
ASC x {anger,fear,joy,sadness}	Emotion specific ASC hidden layers.	4×25
at	'@' symbol in tweet.	1
blob	TextBlob sentiment library.	1
caps	Occurrence of all capitalized words.	1
dim	Diminisher words.	1
{ft,w2v} x {150,200} x {anger,fear,joy,sadness}	Hidden layers of models used to re-train the embeddings.	$4 \times 4 \times 15$
hash	'#' symbol in tweet.	1
hash x {anger,fear,joy,sadness}	Affection lexicon of hashtags.	4
irony	Occurrence of #irony or #sarcasm hashtags.	1
length	Logarithm of sentence length.	1
long	Elongated words, 'wowwww'.	1
mag	Magnifiers.	1
vader	Vader sentiment library.	3
negative	Negative emojis.	1
neutral	Neutral emojis.	1
positive	Positive emojis.	1
anger/1	Detection of emojis and words related to the given emotion, taken from a manually annotated list.	2
fear/1		2
joy/1		2
sadness/1		2
love		1
surprise		1
disappointed		1
lonely		1
hopeful		1

Table 6: Complete list of features generated from datasets.