DataStories at SemEval-2017 Task 4: Bidirectional LSTM with Attention for Message-level and Topic-based Sentiment Analysis

Christos Baziotis, Nikos Pelekis, Christos Doulkeridis
University of Piraeus - Data Science Lab
Piraeus, Greece
mpsp14057@unipi.gr, npelekis@unipi.gr, cdoulk@unipi.gr

Abstract

In this paper we present two deep-learning systems that competed at SemEval-2017 Task 4 "Sentiment Analysis in Twitter". We participated in all subtasks for English tweets, involving message-level and topic-based sentiment polarity classification and quantification. We use Long Short-Term Memory (LSTM) networks augmented with two kinds of attention mechanisms, on top of word embeddings pre-trained on a big collection of Twitter messages. Also, we present a text processing tool suitable for social network messages, which performs tokenization, word normalization, segmentation and spell correction. Moreover, our approach uses no hand-crafted features or sentiment lexicons. We ranked 1st (tie) in Subtask A, and achieved very competitive results in the rest of the Subtasks. Both the word embeddings and our text processing tool[1] are available to the research community.

1 Introduction

Sentiment analysis is an area in Natural Language Processing (NLP), studying the identification and quantification of the sentiment expressed in text. Sentiment analysis in Twitter is a particularly challenging task, because of the informal and "creative" writing style, with improper use of grammar, figurative language, misspellings and slang.

In previous runs of the Task, sentiment analysis was usually tackled using hand-crafted features and/or sentiment lexicons (Mohammad et al., 2013; Kiritchenko et al., 2014; Palogiannidi et al., 2016), feeding them to classifiers such as Naive Bayes or Support Vector Machines (SVM). These approaches require a laborious feature-engineering process, which may also need domain-specific knowledge, usually resulting both in redundant and missing features. Whereas, artificial neural networks (Deriu et al., 2016; Rouvier and Favre, 2016) which perform feature learning, last year (Nakov et al., 2016) achieved very good results, outperforming the competition.

In this paper, we present two deep-learning systems that competed at SemEval-2017 Task 4 (Rosenthal et al., 2017). Our first model is designed for addressing the problem of message-level sentiment analysis (Subtask A). We employ a 2-layer Bidirectional LSTM, equipped with an attention mechanism (Rocktäschel et al., 2015). For the topic-based sentiment analysis tasks, we propose a Siamese Bidirectional LSTM with a context-aware attention mechanism (Yang et al., 2016). In contrast to top-performing systems of previous years, we do not rely on hand-crafted features, sentiment lexicons and we do not use model ensembles. We make the following contributions:

- A text processing tool for text tokenization, word normalization, word segmentation and spell correction, geared towards Twitter.
- A deep learning system for short-text sentiment analysis using an attention mechanism, in order to enforce the contribution of words that determine the sentiment of a message.
- A deep learning system for topic-based sentiment analysis, with a context-aware attention mechanism utilizing the topic information.

2 Overview

Figure 1 provides a high-level overview of our approach, which consists of two main steps and an optional task-dependent third step: (1) the *text processing*, where we use our own text processing tool for preparing the data for our neural network, (2) the *learning* step, where we train the neural

[1] github.com/cbaziotis/ekphrasis

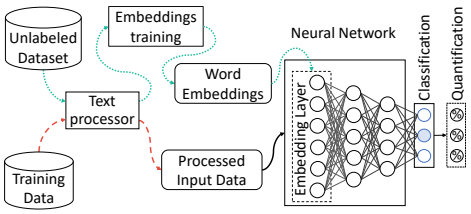


Figure 1: High-level overview of our approach

networks and (3) the *quantification* step for estimating the sentiment distribution for each topic.

Task definitions. In Subtask A, given a message we must classify whether the message expresses positive, negative, or neutral sentiment (3-point scale). In Subtasks B, C (topic-based sentiment polarity classification) we are given a message and a topic and must classify the message on 2-point scale (Subtask B) and a 5-point scale (Subtask C). In Subtasks D, E (quantification) we are given a set of messages about a set of topics and must estimate the distribution of the tweets across 2-point scale (Subtask D) and a 5-point scale (Subtask E).

Unlabeled Dataset. We collected a big dataset of 330M English Twitter messages, gathered from 12/2012 to 07/2016, which is used (1) for calculating words statistics needed by our text processor and (2) for training our word embeddings.

Pre-trained Word Embeddings. Word embeddings are dense vector representations of words (Collobert and Weston, 2008; Mikolov et al., 2013), capturing their semantic and syntactic information. We leverage our big collection of Twitter messages to generate word embeddings, with vocabulary size of 660K words, using GloVe (Pennington et al., 2014). The pre-trained word embeddings are used for initializing the first layer (embedding layer) of our neural networks.

2.1 Text Processor

We developed our own text processing tool, in order to preserve most of the information in text, performing sentiment-aware tokenization, spell correction, word normalization, word segmentation (for splitting hashtags) and word annotation.

Tokenizer. The difficulty in tokenization is to avoid splitting expressions or words that should be kept intact (as one token). Although there are some tokenizers geared towards Twitter (Potts, 2011; Gimpel et al., 2011) that recognize the Twitter markup and some basic sentiment expressions or simple emoticons, our tokenizer is able to identify most emoticons, emojis, expressions such as dates (e.g. 07/11/2011, April 23rd), times (e.g. 4:30pm, 11:00 am), currencies (e.g. $10, 25mil, 50€), acronyms, censored words (e.g. s**t), words with emphasis (e.g. *very*) and more.

Text Postprocessing. After the tokenization we add an extra postprocessing step, performing modifications on the extracted tokens. This is where we perform spell correction, word normalization and segmentation and decide which tokens to omit, normalize or annotate (surround or replace with special tags). For the tasks of spell correction (Jurafsky and Martin, 2000) and word segmentation (Segaran and Hammerbacher, 2009) we used the Viterbi algorithm, utilizing word statistics (unigrams and bigrams) from our unlabeled dataset, to obtain word probabilities. Moreover, we lowercase all words, and normalize URLs, emails and user handles (@user).

After performing the aforementioned steps we decrease the vocabulary size, while keeping information that is usually lost during the tokenization phase. Table 1 shows an example of the text processing tool.

2.2 Neural Networks

Last year, most of the top scoring systems used Convolutional Neural Networks (CNN) (LeCun et al., 1998). Even though CNNs are designed for computer vision, the fact that they are fast and easy to train, makes them a popular choice for NLP problems. However CNNs have no notion of order, thus when applying them to NLP tasks the crucial information of the word order is lost.

2.2.1 Recurrent Neural Networks

A more natural choice is to use Recurrent Neural Networks (RNN). An RNN processes an input sequentially in a way that resembles how humans do it. It performs the same operation, $h_t = f_W(x_t, h_{t-1})$, on every element of a sequence, where h_t is the hidden state a timestep t, and

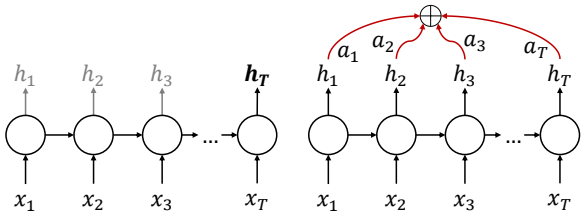
original	The *new* season of #TwinPeaks is coming on May 21, 2017. CANT WAIT \o/ !!! #tvseries #davidlynch :D
processed	the new <emphasis> season of <hashtag> twin peaks </hashtag> is coming on <date> . cant <allcaps> wait <allcaps> <happy> ! <repeated> <hashtag> tv series </hashtag> <hashtag> david lynch </hashtag> <laugh>

Table 1: Example of our text processor

W the weights of the network. The hidden state at each timestep depends on the previous hidden states. This is why the order of the elements (words) is important. This process also enables RNNs to handle inputs of variable length.

RNNs are difficult to train (Pascanu et al., 2013), because gradients may grow or decay exponentially over long sequences (Bengio et al., 1994; Hochreiter et al., 2001). A way to overcome these problems is by using one of the more sophisticated variants of the regular RNN, the Long Short-Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) or the recently proposed Gated Recurrent Units (GRU) (Cho et al., 2014). Both variants introduce a gating mechanism, ensuring proper gradient propagation through the network. We use the LSTM, as it performed slightly better than GRU in our experiments.

Attention Mechanism. An RNN updates its hidden state h_i as it processes a sequence and at the end, the hidden state holds a summary of all the processed information. In order to amplify the contribution of important elements in the final representation we use an attention mechanism (Rocktäschel et al., 2015; Raffel and Ellis, 2015), that aggregates all the hidden states using their relative importance (see Figure 2).



(a) Regular RNN (b) Attention RNN

Figure 2: Comparison between the regular RNN and the RNN with attention.

2.3 Quantification

For the quantification tasks an obvious approach is the Classify & Count (Forman, 2008), where we simply compute the fraction of a topic's messages that a classifier predicts to belong to a class c. Another approach is the Probabilistic Classify & Count (PCC) (Gao and Sebastiani, 2016), in which first we train a classifier that produces a probability distribution over the classes and then we average the estimated probabilities for each class to obtain the final distribution. Let T be the set of topics in the training set and $p(c|tweet)$ the (posterior) probability that a tweet belongs to class c as estimated by the classifier. Then we estimate the expected fraction of a topic's tweets that belong to class c as follows:

$$\hat{p}_T(c) = \frac{1}{|T|} \sum_{tweet \in T} p(c|tweet) \quad (1)$$

3 Models Description

We propose two different models, a Message-level Sentiment Analysis (MSA) model for Subtask A (3.1) and a Topic-based Sentiment Analysis (TSA) (3.2) model for Subtasks B,C,D,E.

3.1 MSA Model (message-level)

Our message-level sentiment analysis model (MSA) consists of a 2-layer bidirectional LSTM (BiLSTM) with an attention mechanism, for identifying the most informative words.

Embedding Layer. The input to the network is a Twitter message, treated as a sequence of words. We use an Embedding layer to project the words $X = (x_1, x_2, ..., x_T)$ to a low-dimensional vector space R^E, where E the size of the Embedding layer and T the number of words in a tweet. We initialize the weights of the embedding layer with our pre-trained word embeddings.

BiLSTM Layers. An LSTM takes as input the words of a tweet and produces the word annotations $H = (h_1, h_2, ..., h_T)$, where h_i is the hidden state of the LSTM at time-step i, summarizing all the information of the sentence up to x_i. We use bidirectional LSTM (BiLSTM) in order to get word annotations that summarize the information from both directions. A bidirectional LSTM consists of a forward LSTM $\overrightarrow{f}$ that reads the sentence from x_1 to x_T and a backward LSTM $\overleftarrow{f}$ that reads the sentence from x_T to x_1. We obtain the final annotation for a given word x_i, by concatenating the annotations from both directions:

$$h_i = \overrightarrow{h_i} \parallel \overleftarrow{h_i}, \quad h_i \in R^{2L} \quad (2)$$

where $\parallel$ denotes the concatenation operation and L the size of each LSTM. We stack two layers of BiLSTMs in order to learn more abstract features.

Attention Layer. Not all words contribute equally to the expression of the sentiment in a message. We use an attention mechanism to find the relative contribution (importance) of each word. The attention mechanism assigns a weight a_i to each word annotation. We compute the fixed representation r of the whole message as the weighted sum

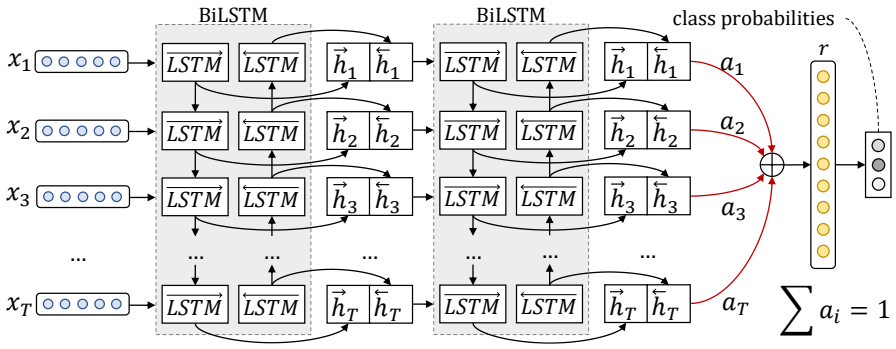


Figure 3: The MSA model: A 2-layer bidirectional LSTM with attention over that last layer.

of all the word annotations. Formally:

$$e_i = tanh(W_h h_i + b_h), \quad e_i \in [-1, 1] \tag{3}$$

$$a_i = \frac{exp(e_i)}{\sum_{t=1}^{T} exp(e_t)}, \quad \sum_{i=1}^{T} a_i = 1 \tag{4}$$

$$r = \sum_{i=1}^{T} a_i h_i, \quad r \in R^{2L} \tag{5}$$

where W_h and b_h are the attention layer's weights, optimized during training to assign bigger weights to the most important words of a sentence.

Output Layer. We use the representation r as feature vector for classification and we feed it to a final fully-connected softmax layer which outputs a probability distribution over all classes.

3.2 TSA Model (topic-based)

For the topic-based sentiment analysis tasks, we propose a Siamese[2] Bidirectional LSTM network with a different attention mechanism than in MSA. Our model is comparable to the work of (Wang et al., 2016; Tang et al., 2015). However our model differs in the way it incorporates topic information and in the attention mechanism.

Embedding Layer. The network has two inputs, the sequence of words in the tweet $X^{tw} = (x_1^{tw}, x_2^{tw}, ..., x_{T_{tw}}^{tw})$, where T_{tw} the number of words in the tweet, and the sequence of words in the topic $X^{to} = (x_1^{to}, x_2^{to}, ..., x_{T_{to}}^{to})$, where T_{to} the number of words in the topic. We project all words to R^E, where E the size of the Embedding layer.

Siamese BiLSTM. We use a bidirectional LSTM with *shared weights* to map the words of the tweet and the topic to the *same* vector space, in order to be able to make meaningful comparison between the two. The BiLSTM produces the annotations for the words of the tweet $H^{tw} = (h_1^{tw}, h_2^{tw}, ..., h_{T_{tw}}^{tw})$ and the topic $H^{to} = (h_1^{to}, h_2^{to}, ..., h_{T_{to}}^{to})$, where each word annotation consists of the concatenation of its forward and backward annotations:

$$h_i^j = \overrightarrow{h_i^j} \parallel \overleftarrow{h_i^j}, \quad h_i^j \in R^{2L}, \quad j \in \{tw, to\} \tag{6}$$

where $\parallel$ denotes the concatenation operation and L the size of each LSTM.

Mean-Pooling Layer. We use a *Mean-Pooling* layer over the word annotations of the topic H^{to} that aggregates them to produce a single annotation. The layer computes the *mean over time* to produce the topic annotation, $\overline{h^{to}} = \frac{1}{T_{to}} \sum_1^{T_{to}} h_i^{to}$, that we *append* to all individual message word annotations to create the final *context-aware* annotation for each word:

$$h_i = h_i^{tw} \parallel \overline{h^{to}}, \quad h_i^j \in R^{4L} \tag{7}$$

Context-Attention Layer. We use a context-aware attention mechanism as in (Yang et al., 2016), in order to strengthen the contribution of words that express sentiment towards a given topic. This is done by adding a context vector u_h that can be interpreted as a fixed query, like "which words express sentiment towards the given topic", over the words of the message. Concretely:

$$e_i = tanh(W_h h_i + b_h), \quad e_i \in [-1, 1] \tag{8}$$

$$a_i = \frac{exp(e_i^\top u_h)}{\sum_{t=1}^{T_{tw}} exp(e_t^\top u_h)}, \quad \sum_{i=1}^{T_{tw}} a_i = 1 \tag{9}$$

$$r = \sum_{i=1}^{T_{tw}} a_i h_i, \quad r \in R^{4L} \tag{10}$$

where W_h, b_h and u_h are jointly learned weights.

Maxout Layer. We pass the representation r to a Maxout (Goodfellow et al., 2013) layer to make the final comparison between the tweet aspects

[2] Siamese are called the networks that have identical configuration and their weights are linked during training.

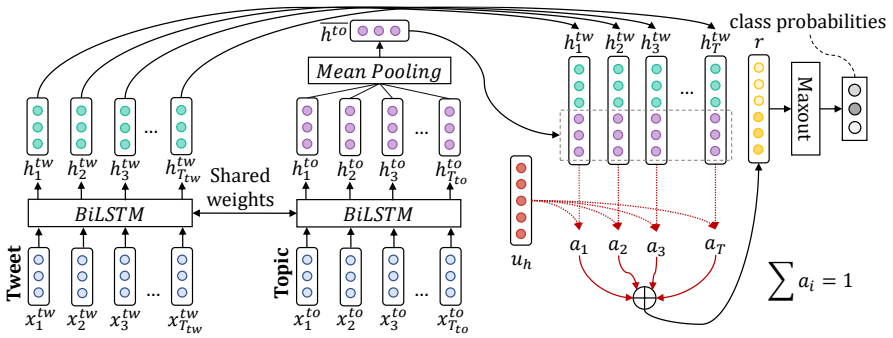


Figure 4: The TSA model: A Siamese Bidirectional LSTM with context-aware attention.

and the topic aspects. We selected Maxout as it amplifies the effects of Dropout (Section 3.3).

Output Layer. We pass the output of the Maxout layer to a final fully-connected softmax layer which outputs a probability distribution over all classes.

3.3 Regularization

In both of our models we add Gaussian noise at the embedding layer, which can be interpreted as a random data augmentation technique, making our models more robust to overfitting. In addition to that we use dropout (Srivastava et al., 2014) to randomly turn-off neurons in our network. Dropout prevents co-adaptation of neurons and can also be thought as a form of ensemble learning, because for each training example a subpart of the whole network is trained. Moreover we apply dropout to the recurrent connections of the LSTM as in (Gal and Ghahramani, 2016). Furthermore we add L_2 regularization penalty (weight decay) to the loss function to discourage large weights. Lastly, we stop training after the validation loss has stopped decreasing (early-stopping).

Finally, we do not fine-tune the embedding layers. Words occurring in the training set, will be moved in the embedding space and the classifier will correlate certain regions (in embedding space) to certain sentiments. However, words in the test set and not in the training set, will remain at their initial position which may no longer reflect their "true" sentiment, leading to miss-classifications.

3.4 Class Weights

In the training data some classes have more training examples than others, introducing bias in our models. In order to deal with this problem we apply class weights to the loss function of our models, penalizing more the misclassification of underrepresented classes. Moreover, we introduce a smoothing factor in order to smooth out the weights in cases where the imbalances are very strong, which would otherwise lead to extremely large class weights. Let x be the vector with the class counts and α the smoothing factor, we obtain class weights with $w_i = \frac{max(x)}{x_i + \alpha \times max(x)}$. In Subtasks A, B, D we use no smoothing ($\alpha = 0$) and in Subtasks C and E we set $\alpha = 0.1$.

3.5 Training

We train all of our networks to minimize the cross-entropy loss, using back-propagation with stochastic gradient descent and mini-batches of size 128. We use Adam (Kingma and Ba, 2014) for tuning the learning rate and we clip the norm of the gradients (Pascanu et al., 2013) at 5, as an extra safety measure against exploding gradients.

Dataset. For training we use the available data from prior years (only tweets). Table 2 shows that statistics of the data we used. Also, we do not use any user information from the tweets (only text).

3.5.1 Hyper-parameters

In order to find good hyper-parameter values in a relative short time, compared to grid or random search, we adopt the Bayesian optimization (Bergstra et al., 2013) approach, performing a "smart" search in the high dimensional space of all the possible values.

MSA Model. The size of the embedding layer is 300, and the LSTM layers 150 (300 for BiLSTM). We add Gaussian noise with $\sigma = 0.2$ and dropout of 0.3 at the embedding layer, dropout of 0.5 at the LSTM layers and dropout of 0.25 at the recurrent connections of the LSTM. Finally, we add L_2 regularization of 0.0001 at the loss function.

TSA Model. The size of the embedding layer is 300, and the LSTM layers 64 (128 for BiLSTM). We insert Gaussian noise with $\sigma = 0.2$ at the embedding layer of both inputs and dropout of 0.3 at

Dataset	Task	Positive		Neutral	Negative		Total
		2	**1**	**0**	**-1**	**-2**	**Total**
Train	A	19652 (39.64%)		22195 (44.78%)	7723 (15.58%)		49570
	B,D	14897 (78.85%)		-	3997 (21.15%)		18894
	C,E	1016 (3.34%)	12852 (42.23%)	12888 (42.35%)	3380 (11.11%)	296 (0.97%)	30432
Test	A	2375 (19.33%)		5937 (48.33%)	3972 (32.33%)		12284
	B,D	2463 (39.82%)		-	3722 (60.18%)		6185
	C,E	131 (1.06%)	2332 (18.84%)	6194 (50.04%)	3545 (28.64%)	177 (1.43%)	12379

Table 2: Dataset statistics. Notice the difference in the ratio of positive-negative classes this year.

the embedding layer of the message, dropout of 0.2 at the LSTM layer and the recurrent connection of the LSTM layer and dropout of 0.3 at the attention layer and the Maxout layer. Finally, we add L_2 regularization of 0.001 at the loss function.

4 Experiments and Results

Semeval Results. Our official ranking (Rosenthal et al., 2017) is 1/38 (tie) in Subtask A, 2/24 in Subtask B, 2/16 in Subtask C, 2/16 in Subtask D and 11/12 in Subtask E. All of our models performed very good, with the exception of Subtask E. Since the quantification was performed on top of the classifier of Subtask C, which came in 2nd place, we conclude that our quantification approach was the reason for the bad results (for Subtask E).

Attention Mechanism. In order to assess the impact of the attention mechanisms, we evaluated the performance of each model, with and without attention. We report (Table 3) the average scores of 10 runs for each system, on the official test set. The attention-based models performed better, but only by a small margin.

RNN	Subtask A (MSA)		Subtask B (TSA)	
	ρ	$F1^{pn}$	ρ	$F1^{pn}$
Regular	0.678	0.673	0.856	0.817
Attention	**0.682**	**0.675**	**0.863**	**0.82**

Table 3: Results of the impact of attention[3].

Quantification. To get a better insight into the quantification approaches, we compare the performance of CC and PCC. It is inconclusive as to which quantification approach is better. PCC outperformed CC in (Bella et al., 2010) but underperformed CC in (Esuli and Sebastiani, 2015). Following the results from (Gao and Sebastiani, 2016), which are reported on sentiment analysis in twitter, we decided to use PCC for both of our quantification submissions. Table 4 shows the performance of our models. PCC performed better than CC for Subtask D but far worse than CC for Subtask E. We hypothesize that two possible reasons for the difference in performance between D and E, might be 1) the difference in the number of classes and 2) the big change in the ratio of *pos*-to-*neg* classes between the training and test sets.

Method	Subtask D			Subtask E
	KLD	AE	RAE	EMD
CC	0.060	**0.093**	**0.608**	**0.359**
PCC	**0.048**	0.095	0.848	0.595

Table 4: Results of the quantification approaches[4].

Experimental setup. For developing our models we used Keras (Chollet, 2015) with Theano (Theano Dev Team, 2016) as backend and Scikit-learn (Pedregosa et al., 2011). We trained our neural networks on a GTX750Ti (4GB). Lastly, we share the source code of our models [5].

5 Conclusion

In this paper, we present two deep-learning systems for short text sentiment analysis developed for SemEval-2017 Task 4 "Sentiment Analysis in Twitter". We use RNNs, utilizing well established methods in the literature. Additionally, we empower our networks with two different kinds of attention mechanisms in order to amplify the contribution of the most important words.

Our models achieved excellent results in the classification tasks, but mixed results in the quantification tasks. We would like to work more in this area and explore more quantification techniques in the future. Another interesting approach would be to design models operating on the character-level.

[3] ρ is the average recall and $F1^{pn}$ the macro-average F1 score of the positive and negative classes

[4] KLD is Kullback-Leibler Divergence, EMD is Earth Mover's Distance, AE is Absolute Error and RAE is Relative Absolute Error. For all metrics lower is better.

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

References

Antonio Bella, Cesar Ferri, José Hernández-Orallo, and Maria Jose Ramirez-Quintana. 2010. Quantification via probability estimators. In *Proceedings of ICDM*. pages 737–742.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

James Bergstra, Daniel Yamins, and David D. Cox. 2013. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *Proceedings of ICML*. pages 115–123.

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

François Chollet. 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 ICML*. pages 160–167.

Jan Deriu, Maurice Gonzenbach, Fatih Uzdilli, Aurélien Lucchi, Valeria De Luca, and Martin Jaggi. 2016. SwissCheese at SemEval-2016 Task 4: Sentiment classification using an ensemble of convolutional neural networks with distant supervision. In *Proceedings of SemEval*. pages 1124–1128.

Andrea Esuli and Fabrizio Sebastiani. 2015. Optimizing Text Quantifiers for Multivariate Loss Functions. *ACM Trans. Knowl. Discov. Data* 9(4):27:1–27:27.

George Forman. 2008. Quantifying counts and costs via classification. *Data Mining and Knowledge Discovery* 17(2):164–206.

Yarin Gal and Zoubin Ghahramani. 2016. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of NIPS*. pages 1019–1027.

Wei Gao and Fabrizio Sebastiani. 2016. From classification to quantification in tweet sentiment analysis. *Social Network Analysis and Mining* 6(1).

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of ACL*. pages 42–47.

Ian J. Goodfellow, David Warde-Farley, Mehdi Mirza, Aaron C. Courville, and Yoshua Bengio. 2013. Maxout networks. In *Proceedings of ICML*. pages 1319–1327.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, and Jürgen Schmidhuber. 2001. *Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-Term Dependencies*. A field guide to dynamical recurrent neural networks. IEEE Press.

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

Daniel Jurafsky and James H. Martin. 2000. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. Prentice Hall PTR, 1st edition.

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

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

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE* 86(11):2278–2324.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*. pages 3111–3119.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242* .

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. Semeval-2016 Task 4: Sentiment analysis in Twitter. In *Proceedings of SemEval*. pages 1–18.

Elisavet Palogiannidi, Athanasia Kolovou, Fenia Christopoulou, Filippos Kokkinos, Elias Iosif, Nikolaos Malandrakis, Haris Papageorgiou, Shrikanth Narayanan, and Alexandros Potamianos. 2016. Tweester at SemEval-2016 Task 4: Sentiment analysis in Twitter using semantic-affective model adaptation. In *Proceedings of SemEval*. pages 155–163.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *Proceedings of ICML*. pages 1310–1318.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, and others. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of EMNLP*. pages 1532–1543.

Christopher Potts. 2011. Sentiment Symposium Tutorial: Tokenizing. http://sentiment.christopherpotts.net/tokenizing.html.

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

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

Sara Rosenthal, Noura Farra, and Preslav Nakov. 2017. SemEval-2017 Task 4: Sentiment Analysis in Twitter. In *Proceedings of SemEval*. Vancouver, Canada.

Mickael Rouvier and Benoît Favre. 2016. SENSEI-LIF at SemEval-2016 Task 4: Polarity embedding fusion for robust sentiment analysis. In *Proceedings of SemEval*. pages 202–208.

Toby Segaran and Jeff Hammerbacher. 2009. *Beautiful Data: The Stories Behind Elegant Data Solutions*. "O'Reilly Media, Inc.".

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15(1):1929–1958.

Duyu Tang, Bing Qin, Xiaocheng Feng, and Ting Liu. 2015. Target-dependent sentiment classification with long short term memory. *CoRR, abs/1512.01100* .

Theano Dev Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints* abs/1605.02688.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of EMNLP*. pages 606–615.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of NAACL-HLT*. pages 1480–1489.