PotTS at SemEval-2016 Task 4: Sentiment Analysis of Twitter Using Character-level Convolutional Neural Networks.

Uladzimir Sidarenka
Applied Computational Linguistics/FSP Cognitive Science
University of Potsdam
Karl-Liebknecht Straße 24-25
14476 Potsdam
sidarenk@uni-potsdam.de

Abstract

This paper presents an alternative approach to polarity and intensity classification of sentiments in microblogs. In contrast to previous works, which either relied on carefully designed hand-crafted feature sets or automatically derived neural embeddings for words, our method harnesses character embeddings as its main input units. We obtain task-specific vector representations of characters by training a deep multi-layer convolutional neural network on the labeled dataset provided to the participants of the SemEval-2016 Shared Task 4 (Sentiment Analysis in Twitter; Nakov et al., 2016b) and subsequently evaluate our classifiers on subtasks B (two-way polarity classification) and C (joint five-way prediction of polarity and intensity) of this competition. Our first system, which uses three manifold convolution sets followed by four non-linear layers, ranks 16 in the former track; while our second network, which consists of a single convolutional filter set followed by a highway layer and three non-linearities with linear mappings in-between, attains the 10-th place on subtask C.[1]

1 Introduction

Sentiment analysis (SA) – a field of knowledge which deals with the analysis of people's opinions, sentiments, evaluations, appraisals, attitudes, and emotions towards particular entities mentioned in discourse (Liu, 2012) – is commonly considered to be one of the most challenging, competitive, but at the same time utmost necessary areas of research in modern computational linguistics.

Unfortunately, despite numerous notable advances in recent years (e.g., Breck et al., 2007; Yessenalina and Cardie, 2011; Socher et al., 2011), many of the challenges in the opinion mining field, such as domain adaptation or analysis of noisy texts, still pose considerable difficulties to researchers. In this respect, rapidly evaluating and comparing different approaches to solving these problems in a controlled environment – like the one provided for the SemEval task (Nakov et al., 2016b) – is of crucial importance for finding the best possible way of mastering them.

We also pursue this goal in the present paper by investigating whether one of the newest machine learning trends – the use of deep neural networks (DNNs) with small receptive fields – would be a viable solution for improving state-of-the-art results in sentiment analysis of Twitter.

After a brief summary of related work in Section 2, we present the architectures of our networks and describe the training procedure we used for them in Section 3. Since we applied two different DNN topologies to subtasks B and C, we make a cross-comparison of both systems and evaluate the role of the preprocessing steps in the next-to-last section. Finally, in Section 5, we draw conclusions from our experiments and make further suggestions for future research.

2 Related Work

Since its presumably first official mention by Nasukawa and Yi in 2003 (cf. Liu, 2012), sentiment analysis has constantly attracted the attention of re-

[1] The source code of our implementation is freely available online at https://github.com/WladimirSidorenko/SemEval-2016/

Proceedings of SemEval-2016, pages 230–237,
San Diego, California, June 16-17, 2016. ©2016 Association for Computational Linguistics

searchers. Though earlier works on opinion mining were primarily concerned with the analysis of narratives (Wiebe, 1994) or newspaper articles (Wiebe et al., 2003), the explosive emergence of social media (SM) services in the mid-2000s has brought about a dramatic focus change in this field.

A particularly important role in this regard was played by Twitter – a popular microblogging service first introduced by Jack Dorsey in 2006 (Dorsey, 2006). The sudden availability of huge amounts of data combined with the presence of all possible social and national groups on this stream rapidly gave rise to a plethora of scientific studies. Notable examples of these were the works conducted by Go et al. (2009) and Pak and Paroubek (2010), who obtained their corpora using distant supervision and subsequently trained several classifiers on these data; Kouloumpis et al. (2011), who trained an AdaBoost system on the Edinburgh Twitter corpus[2]; and Agarwal et al. (2011), who proposed tree-kernel methods for doing message-level sentiment analysis of tweets.

Eventually, with the introduction of the SemEval corpus (Nakov et al., 2013), a great deal of automatic systems and resources have appeared on the scene. Though most of these systems typically rely on traditional supervised classification methods, such as SVM (Mohammad et al., 2013; Becker et al., 2013) or logistic regression (Hamdan et al., 2015; Plotnikova et al., 2015), in recent years, the deep learning (DL) tsunami (Manning, 2015) has also started hitting the shores of this "battlefield".

In this paper we investigate whether one of the newest lines of research in DL – the use of character-level deep neural networks (charDNNs) – would be a perspective way for addressing the sentiment analysis task on Twitter as well.

Introduced by Sutskever et al. (2011), char-DNNs have already proved their efficiency for a variety of NLP applications, including part-of-speech tagging (dos Santos and Zadrozny, 2014), named-entity recognition (dos Santos and Guimarães, 2015), and general language modeling (Kim et al., 2015; Józefowicz et al., 2016). We hypothesized that the reduced feature sparsity of this approach, its lower susceptibility to informal spellings, and the shift of the main discriminative classification power from input units to transformation layers would make it suitable for doing opinion mining on Twitter as well.

3 Method

To test our conjectures, we downloaded the training and development data provided by the organizers of the SemEval-2016 Task 4 (Sentiment Analysis in Twitter; Nakov et al., 2016b). Due to dynamic changes of this content, we were only able to retrieve a total of 5,178 messages for subtasks B and D (two-way polarity classification) and 7,335 microblogs for subtasks C and E (joint five-way prediction of polarity and intensity).

We deliberately refused to do any heavy-weight NLP preprocessing of these data to check whether the applied DL method alone would suffice to get acceptable results. In order to facilitate the training and reduce the variance of the learned weights though, we applied a shallow normalization of the input by lower-casing messages' strings and filtering out stop words before passing the tweets to the classifiers.

As stop words we considered all auxiliary verbs (e.g., *be*, *have*, *do*) and auxiliary parts of speech (prepositions, articles, particles, and conjunctions) up to a few exceptions – we kept the negations and words that potentially could inverse the polarity of opinions, e.g., *without*, *but*, and *however*. Furthermore, we also removed hyperlinks, digits, retweets, @-mentions, common temporal expressions, and mentions of tweets' topics, since all of these elements were a priori guaranteed to be objective. An example of such preprocessed microblog is provided below:

EXAMPLE 3.1.
Original: *Going to MetLife tomorrow but not to see the boys is a weird feeling*
Normalized: *but not see boys weird feeling*

3.1 Adversarial Convolutional Networks (Subtasks B and D)

We then defined a multi-layer deep convolutional network for subtasks B and D as follows:

At the initial step, we map the input characters to their appropriate embeddings, obtaining an input matrix $E \in \mathbb{R}^{n \times m}$, where n stands for the length of

[2] `http://demeter.inf.ed.ac.uk`

the input instance, and m denotes the dimensionality of the embedding space (specifically, we use $m = 32$).

Next, three sets of convolutional filters – positive (+), negative (−), and shifter (x) convolutions – are applied to the input matrix E. Each of these sets in turn consists of three subsets: one subset with 4 filters of width 3, another subset comprising 8 filters of width 4, and, finally, a third subset having 12 filters of width 5.[3]

Each subset filter F forms a matrix $\mathbb{R}^{w \times m}$ with the number of rows w corresponding to the filter width and the number of columns m being equal to the embedding dimensionality as above. A subset of filters S^p_w for $p \in \{+, -, \mathrm{x}\}$ is then naturally represented as a tensor $\mathbb{R}^{c \times w \times m}$, where c is the number of filters with the given width w.

We apply the usual convolution operation with max-pooling over time for each filter, getting an output vector $\vec{v}_{S^p_w} \in \mathbb{R}^c$ for each subset. All output vectors $\vec{v}_{S^p_*}$ of the same subset are then concatenated into one vector $\vec{v}_{S^p} = [\vec{v}_{S^p_3}, \vec{v}_{S^p_4}, \vec{v}_{S^p_5}]$ of size $4 + 8 + 12 = 24$.

The results of the three sets are subsequently joined using the following equation:

$$\vec{v}_{conv} = \mathrm{sig}(\vec{v}_{S^+} - \vec{v}_{S^-}) \odot \tanh(\vec{v}_{S^\mathrm{x}}),$$

where $\vec{v}_{S^+}$, $\vec{v}_{S^-}$, and $\vec{v}_{S^\mathrm{x}}$ mean the output vectors for the positive, negative, and shifter sets respectively, and $\odot$ denotes the Hadamard product.

The motivation behind this choice of unification function is that we first want to obtain the difference between the positive and negative predictions (thus $\vec{v}_{S^+} - \vec{v}_{S^-}$), then map this difference to the range $[0, 1]$ (therefore the sigmoid), and finally either inverse or dampen these results depending on the output of the shifter layer, whose values are guaranteed to be in the range $[-1, 1]$ thanks to $\tanh$. Since we simultaneously apply competing convolutions to the same input, we call this layer "adversarial" as all of its components have different opinions regarding the final outcome.

After obtaining $\vec{v}_{conv}$, we consecutively use three non-linear transformations (linear rectification, hyperbolic tangent, and sigmoid function) with linear modifications in-between:

$$\begin{aligned}
\vec{v}_{relu} &= \mathrm{relu}(\vec{v}_{conv} \cdot M_{relu} + \vec{b}_{relu}),\\
\vec{v}_{tanh} &= \tanh(\vec{v}_{relu} \cdot M_{tanh} + \vec{b}_{tanh}),\\
\vec{v}_{sig} &= \mathrm{sig}(\vec{v}_{tanh} \cdot M_{sig} + \vec{b}_{sig}).
\end{aligned}$$

In this equation, M_{relu}, M_{tanh}, and $M_{sig} \in \mathbb{R}^{24 \times 24}$ stand for the linear transform matrices, and $\vec{b}_{relu}, \vec{b}_{tanh}, \vec{b}_{sig} \in \mathbb{R}^{24}$ represent the usual bias terms. With this combination, we hope to first prune unreliable input signals by using a hard rectifying linear unit (Jarrett et al., 2009) and then gain more discriminative power by successively applying $\tanh$ and sig, thus funneling the input to increasingly smaller ranges: $[-1, 1]$ in the case of $\tanh$, and $[0, 1]$ in the case of sigmoid.

At the last stage, after applying a binomial dropout mask with $p = 0.5$ to the $\vec{v}_{sig}$ vector (Srivastava et al., 2014), we compute the final prediction as:

$$y' = \begin{cases} 1, & \text{if } \mathrm{sig}(\sum \vec{v}_{sig} \cdot M_{pred} + \vec{b}_{pred}) \geq 0.5 \\ 0, & \text{otherwise,} \end{cases} \tag{1}$$

where $M_{pred} \in \mathbb{R}^{24 \times 2}$ and $\vec{b}_{pred} \in \mathbb{R}^2$ stand for the transformation matrix and bias term respectively, and the summation runs over the two elements of the resulting $\mathbb{R}^2$ vector.

To train our classifier, we normally define the cost function as:

$$cost = \sum_i y_i * (1 - y'_i) + (1 - y_i) * y'_i, \tag{2}$$

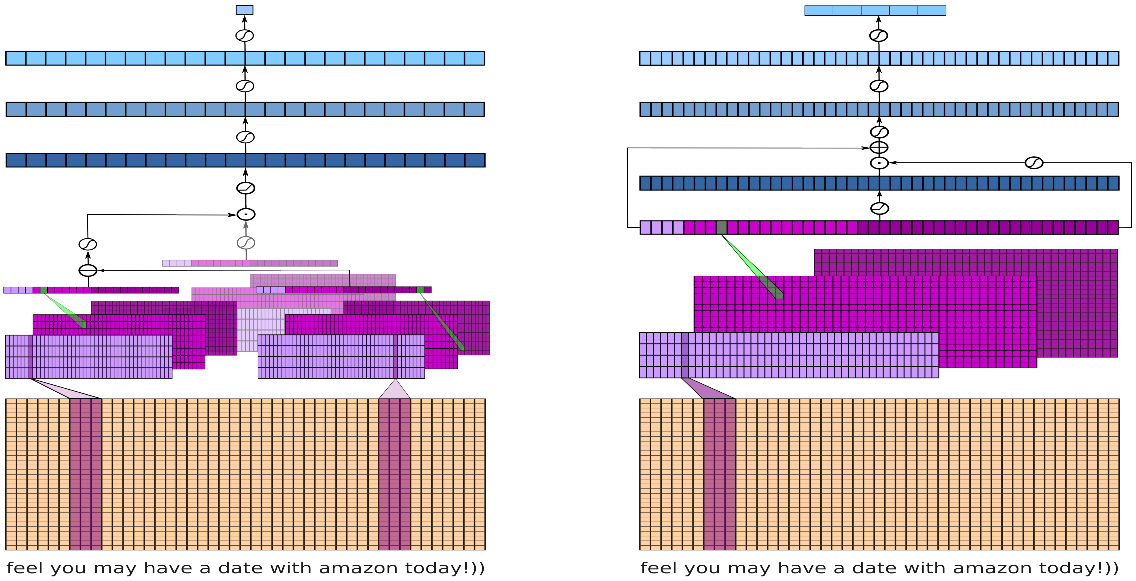
where y_i denotes the gold category of the i-th training instance and y'_i stands for its predicted class, and optimize this function using RMSProp (Tieleman and Hinton, 2012).

3.2 Highway Convolutional Networks (Subtasks C and E)

A slightly different model was used for subtasks C and E:

In contrast to the previous two-way classification network, we only use one set of convolutions with 4 filters of width 3, 16 filters of width 4, and 24 filters of width 5, and the number of dimensions of the

[3] By simultaneously applying multiple filter sets of different width to the same input, we hoped to improve the precision-recall trade-off, getting more accurate outputs from wider filters while reducing their sparsity with narrower kernels.


feel you may have a date with amazon today!))
feel you may have a date with amazon today!))

(a) Adversarial network used for subtasks B and D.

(b) Highway network used for subtasks C and E.

Figure 1: Network architectures.

resulting $\vec{v}_{conv}$ vector being equal to 44 instead of 24.

After normally computing and max-pooling the convolutions, we pass the output convolution vector through a highway layer (Srivastava et al., 2015) in addition to using relu, i.e.:

$$
\begin{aligned}
\vec{v}_{hw_{trans}} &= \mathrm{sig}(\vec{v}_{conv} \cdot M_{hw_{trans}} + \vec{b}_{hw_{trans}}),\\
\vec{v}_{hw_{carry}} &= \vec{v}_{conv} \odot (1 - \vec{v}_{hw_{trans}}),\\
\vec{v}_{relu'} &= \mathrm{relu}(\vec{v}_{conv} \cdot M_{conv'} + \vec{b}_{conv'}),\\
\vec{v}_{relu} &= \mathrm{sig}(\vec{v}_{relu'} \odot \vec{v}_{hw_{trans}} + \vec{v}_{hw_{carry}}).
\end{aligned}
$$

The rest of the network is organized the same way as in the previous model, up to the final layer. Since this task involves multivariate classification, instead of computing the sigmoid of the sum as in Equation 1, we obtain a softmax vector $\vec{v}_\sigma \in \mathbb{R}^5$ and consider the argmax value of this vector as the predicted class:

$$
\begin{aligned}
\vec{v}_\sigma &= \sigma(\vec{v}_{sig} \cdot M_\sigma + \vec{b}_\sigma)\\
y' &= \mathrm{argmax}(\vec{v}_\sigma)
\end{aligned}
$$

The corresponding cost function is appropriately defined as:

$$
cost = \sum_i - \ln \vec{v}_\sigma[y_i] + \ell_2 * \sum_p \|p\|^2 + \ell_3 * (y_i - y'_i)^2,
$$

where $\vec{v}_\sigma[y_i]$ means the probability value for the gold class in the $\vec{v}_\sigma$ vector, ℓ_2 and ℓ_3 are constants (we use $\ell_2 = 1\mathrm{e}^{-5}$ and $\ell_3 = 3\mathrm{e}^{-4}$), p's denote the training parameters of the model, and $(y_i - y'_i)^2$ stand for the squared difference between the numerical values of the predicted and gold classes.

In this task, we opted for the L_2 regularization instead of using dropout, since we found it working slightly better on the development set, though the differences between the two methods were not very big, and the derivative computation with dropout was significantly faster.

3.3 Initialization and Training

Because initialization has a crucial impact on the results of deep learning approaches (Sutskever et al., 2011), we did not rely on purely random weights but used the uniform He method (He et al., 2015) for initially setting the embeddings, convolutional filters, and bias terms instead. The inter-layer transformations were set to orthogonal matrices to ensure their full rank.

Additionally, to guarantee that each preceding network stage came maximally prepared and provided best possible output to its successors, after adding each new intermediate layer, we temporarily short-circuited it to the final output node(s) and pretrained this abridged network for 5 epochs, removing the short-circuit connections afterwards. The final training then took 50 epochs with each epoch lasting for 35 iterations over the provided training data.

Since our models appeared to be very susceptible to imbalanced classes, we subsampled the training data by getting $\min(1.1 * n_{min}, n_c)$ samples for each distinct gold category c, where n_{min} is the number of instances of the rarest class in the corpus, and n_c denotes the number of training examples belonging to the c-th class. This subset was resampled anew for each subsequent training epoch.

Finally, to steer our networks towards recognizing correct features, we randomly added additional training instances from two established sentiment lexica: Subjectivity Clues (Wilson et al., 2005) and NRC Hashtag Affirmative Context Sentiment Lexicon (Kiritchenko et al., 2014). To that end, we drew n binary random numbers for each polarity class in the corpus from a binomial distribution $B(n, 0.1)$, where n stands for the total size of the generated training set, and added a uniformly chosen term from either lexica whenever the sampled value was equal to one. In the same way, we randomly (with the probability $B(m, 0.15)$, where m means the number of matches) replaced occurrences of terms from the lexica in the training tweets with other uniformly drawn lexicon items.

4 Evaluation

To train our final model, we used both training and development data provided by the organizers, setting aside 15 percent of the samples drawn in each epoch for evaluation and using the remaining 85 percent for optimizing the networks' weights.

We obtained the final classifier by choosing the network state that produced the best task-specific score on the set-aside part of the corpus during the training. For this purpose, in each training iteration, we estimated the *macroaveraged recall* ρ^{PN} on the evaluation set for subtask B:

$$\rho^{PN} = \frac{\rho^{Pos} + \rho^{Neg}}{2},$$

and computed the *macroaveraged mean absolute error* measure MAE^M (cf. Nakov et al., 2016a) to select a model for track C :

$$MAE^M(h, Te) = \frac{1}{|\mathcal{C}|} \sum_{j=1}^{|\mathcal{C}|} \frac{1}{|Te_j|} \sum_{x \in Te_j} |h(x_i) - y_i|$$

The resulting models were then used in both classification and quantification subtasks of the SemEval competition, i.e., we used the adversarial network with the maximum ρ^{PN} score observed during the training to generate the output for tracks B and D and applied the highway classifier with the minimum achieved MAE^M rate to get predictions for subtasks C and E.[4] The scores of the final evaluation on the official test set are shown in Table 1.

Since many of our parameter and design choices were made empirically by analyzing systems' errors at each development step, we decided to recheck whether these decisions were still optimal for the final configuration. To that end, we re-evaluated the effects of the preprocessing steps by temporarily switching off lower-casing and stop word filtering, and also estimated the impact of the network structure by applying the model architecture used for subtask B to the five-way prediction task, and vice versa using the highway network for the binary classification. The output layers, costs, and regularization functions of these two approaches were also swapped in these experiments when applied to different objectives.

Because re-running the complete training from scratch was relatively expensive (taking eight to ten hours on our machine), we reduced the number of training epochs by a factor of five, but tested each configuration thrice in order to overcome the random factors in the He initialization. The arithmetic mean and standard deviation (with $N = 2$) of these three outcomes for each setting are also provided in the table.

As can be seen from the results, running fewer training epochs does not notably harm the final prediction quality for the binary task. On the contrary, it might even lead to some improvements for the adversarial network. We explain this effect by the fact that the model selected during the shorter training had a lower score on the evaluation set than the network state chosen during 50 epochs. Nevertheless,

[4] We used the official aggregating scripts to generate the results for the quantification tasks.

Training Configuration	$\rho^{PN}\wedge$ (Subtask B)	$MAE^{M}\vee$ (Subtask C)
Adversarial$^{1/5,\text{cs}}$	$61.34^{\pm 1.24}$	$1.3^{\pm 0.05}$
Adversarial$^{1/5,\text{sw}}$	$58.64^{\pm 0.8}$	$1.3^{\pm 0.05}$
Adversarial$^{1/5}$	$61.9^{\pm 0.66}$	$1.37^{\pm 0.03}$
Adversarial	61.8	n/a
Highway$^{1/5,\text{cs}}$	$59.87^{\pm 0.79}$	$1.26^{\pm 0.01}$
Highway$^{1/5,\text{sw}}$	$60.35^{\pm 1.5}$	$1.23^{\pm 0.05}$
Highway$^{1/5}$	$62.05^{\pm 0.75}$	$1.3^{\pm 0.04}$
Highway	n/a	1.24

Table 1: Results of the adversarial and highway networks with different preprocessing steps on Subtasks B and C. ($\wedge$ – *higher is better;* $\vee$ – *lower is better;* $^{1/5}$ – *using 1/5 of training epochs;* $^{\text{cs}}$ – *preserving the character case;* $^{\text{sw}}$ – *keeping stop words*)

despite its worse evaluation results, this first configuration was more able to fit the test data than the second system, which apparently overfitted the set-aside part of the corpus.

Furthermore, we also can observe a mixed effect of the normalization on the two tasks: while keeping stop words and preserving character case deteriorates the results for the binary classification, abandoning any preprocessing steps turns out to be a more favorable solution when doing five-way prediction. The reasons for such different behavior are presumably twofold: *a)* the character case by itself might serve as a good indicator of sentiment intensity but be rather irrelevant to expressing its polarity, and *b)* the number of training instances might have become scarce as the number of possible gold classes in the corpus increased.

Finally, one also can see that the highway network performs slightly better on both subtasks (two- and five-way) than its adversarial counterpart when used with shorter training. In this case, we assume that the swapping of the regularization and cost functions has hidden the distinctions of the two networks at their initial layers, since, in our earlier experiments, we did observe better results for the two-way classification with the adversarial structure.

5 Discussion and Conclusion

Unfortunately, despite our seemingly sound theoretical assumptions set forth at the beginning, relying on character embeddings as input did not work out in practice at the end. Our adversarial system was only ranked fourth to last on subtask B, and the highway network attained the second to last place in track C. However, knowing this outcome in advance was not possible without trying out these approaches first.

In order to make a retrospective error analysis, we computed the correlation coefficients between the character n-grams occurring in the training data and their gold classes, also comparing these figures with the corresponding numbers obtained on the test set. The results of this comparison are shown in Table 2.

3 chars	ρ_{train}	ρ_{test}	**4 chars**	ρ_{train}	ρ_{test}	**5 chars**	ρ_{train}	ρ_{test}
urk	0.128	0.039	turk	0.14	0.036	␣turk	0.127	0.036
pol	0.125	0.069	fail	0.124	0.038	␣trum	0.122	0.055
why	0.112	0.083	rkey	0.112	0.036	urkey	0.117	0.036
ion	0.106	0.024	rump	0.112	0.067	turke	0.117	0.036
␣no	0.105	0.109	urke	0.108	0.036	trump	0.112	0.067
tio	0.104	0.006	␣ame	0.107	0.047	rump␣	0.103	0.059
ate	0.104	0.031	␣pol	0.105	0.063	rkey␣	0.101	0.036
hy␣	0.103	0.9	why␣	0.105	0.085	␣not	0.097	0.1
ot␣	0.102	0.071	trum	0.104	0.054	␣poll	0.096	0.026
isi	0.097	0.075	tion	0.104	0.006	amend	0.096	0.062

Table 2: Top-10 character n-grams from the training data and their correlation coefficients with the negative class on the training (ρ_{train}) and test sets (ρ_{test}) of subtask B.

As can be seen from the table, the most reliable classification traits that could have been learned during the training are very specific to their respective topics – in particular, *Trump* and *Turkey* appear to be very negatively biased terms. This effect becomes even more evident as the length of the character n-grams increases. The reason why we did not pre-filter these substrings in the preprocessing was that the respective topics of these messages were specified as *donald trump* and *erdogan*, but we only removed exact topic matches from tweets.

Due to this evident topic susceptibility, as a possible way to improve our results, we could imagine the inclusion of more training data. Applying ensemble approaches, as it was done by the top-scoring systems this year, could also be a perspective direction to go. We would, however, advise the reader from further experimenting with network architectures (at least when training on the original SemEval dataset only), since both the recursive (RNTN, Socher et al., 2012) and recurrent variants (LSTM, Hochreiter and Schmidhuber, 1997) of neural classifiers were found to perform worse in our experiments than the feed-forward structure we described.

References

Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment Analysis of Twitter Data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lee Becker, George Erhart, David Skiba, and Valentine Matula. 2013. AVAYA: Sentiment Analysis on Twitter with Self-Training and Polarity Lexicon Expansion. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 333–340, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Eric Breck, Yejin Choi, and Claire Cardie. 2007. Identifying expressions of opinion in context. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 2683–2688.

Jack Dorsey. 2006. just setting up my twttr.

Cícero Nogueira dos Santos and Victor Guimarães. 2015. Boosting named entity recognition with neural character embeddings. *CoRR*, abs/1505.05008.

Cícero Nogueira dos Santos and Bianca Zadrozny. 2014. Learning character-level representations for part-of-speech tagging. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Proceedings*, pages 1818–1826. JMLR.org.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter Sentiment Classification using Distant Supervision. *Technical report*, pages 1–6.

Hussam Hamdan, Patrice Bellot, and Frederic Bechet. 2015. Lsislif: Feature Extraction and Label Weighting for Sentiment Analysis in Twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 568–573, Denver, Colorado, June. Association for Computational Linguistics.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, abs/1502.01852.

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

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the best multi-stage architecture for object recognition? In *IEEE 12th International Conference on Computer Vision, ICCV 2009, Kyoto, Japan, September 27 - October 4, 2009*, pages 2146–2153. IEEE.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Yoon Kim, Yacine Jernite, David Sontag, and Alexander M. Rush. 2015. Character-aware neural language models. *CoRR*, abs/1508.06615.

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M. Mohammad. 2014. Sentiment analysis of short informal texts. *J. Artif. Intell. Res. (JAIR)*, 50:723–762.

Efthymios Kouloumpis, Theresa Wilson, and Johanna D. Moore. 2011. Twitter Sentiment Analysis: The Good the Bad and the OMG! In Lada A. Adamic, Ricardo A. Baeza-Yates, and Scott Counts, editors, *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*. The AAAI Press.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Christopher D. Manning. 2015. Computational linguistics and deep learning. *Computational Linguistics*, 41(4):701–707.

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. *CoRR*, abs/1308.6242.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Preslav Nakov, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016a. Evaluation measures for the SemEval-2016 task 4: "sentiment analysis in Twitter".

Preslav Nakov, Alan Ritter, Sara Rosenthal, Veselin Stoyanov, and Fabrizio Sebastiani. 2016b. SemEval-2016 task 4: Sentiment analysis in Twitter. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval 2016)*, San Diego, California, June. Association for Computational Linguistics.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010,* *Valletta, Malta*. European Language Resources Association.

Nataliia Plotnikova, Micha Kohl, Kevin Volkert, Stefan Evert, Andreas Lerner, Natalie Dykes, and Heiko Ermer. 2015. KLUEless: Polarity Classification and Association. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 619–625, Denver, Colorado, June. Association for Computational Linguistics.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 151–161. ACL.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In Jun'ichi Tsujii, James Henderson, and Marius Pasca, editors, *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2012, July 12-14, 2012, Jeju Island, Korea*, pages 1201–1211. ACL.

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.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Highway networks. *CoRR*, abs/1505.00387.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In Lise Getoor and Tobias Scheffer, editors, *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1017–1024. Omnipress.

T. Tieleman and G. Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural Networks for Machine Learning.

Janyce Wiebe, Eric Breck, Chris Buckley, Claire Cardie, Paul Davis, Bruce Fraser, Diane J. Litman, David R. Pierce, Ellen Riloff, Theresa Wilson, David S. Day, and Mark T. Maybury. 2003. Recognizing and organizing opinions expressed in the world press. In Mark T. Maybury, editor, *New Directions in Question Answering, Papers from 2003 AAAI Spring Symposium, Stanford University, Stanford, CA, USA*, pages 12–19. AAAI Press.

Janyce Wiebe. 1994. Tracking point of view in narrative. *Computational Linguistics*, 20(2):233–287.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. The Association for Computational Linguistics.

Ainur Yessenalina and Claire Cardie. 2011. Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing, EMNLP 2011, 27-31 July 2011, John McIntyre Conference Centre, Edinburgh, UK, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 172–182.