An Efficient Deep Neural Architecture for Multilingual Sentiment Analysis in Twitter

Willian Becker, Jônatas Wehrmann, Henry E. L. Cagnini, Rodrigo C. Barros
Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, Porto Alegre, RS, Brazil
Email: rodrigo.barros@pucrs.br

Abstract

Sentiment analysis of tweets is often monolingual and the models provided by machine learning classifiers are usually not applicable across distinct languages. Cross-language sentiment classification usually relies on machine translation strategies in which a source language is translated to the desired target language. Machine translation is costly and the provided results are limited by the quality of the translation that is performed. In this paper, we propose an efficient translation-free deep neural architecture for performing multilingual sentiment analysis of tweets. Our proposed approach benefits from a cost-effective character-based embedding and from optimized convolutions to learn from multiple distinct languages. The resulting model is capable of learning latent features from all languages used during training at once and it does not require any translation process to be performed whatsoever. We empirically evaluate the efficiency and effectiveness of the proposed approach in tweet corpora from four different languages and we show that it presents the best trade-off among four distinct state-of-the-art deep neural architectures for sentiment analysis.

Introduction

Twitter is one of the largest microblogging services on the Internet, in which users share their opinions about any topic. The large amount of generated data has become a rich source for researchers in different scientific fields such as machine learning (ML) and natural language processing (NLP). One of the most studied tasks regarding Twitter data is sentiment analysis, which is concerned with assigning polarity to each tweet. Tweets are short text with up to 140 characters regarding a given topic, and they may reflect the emotional state of mind of their authors with respect to the subject being approached. Thus a tweet may indicate a positive, neutral, or negative sentiment regarding its main topic. Many efforts have been dedicated to sentiment analysis for Twitter corpora over the past years (Martínez-Cámara et al. 2014).

Sentiment analysis over tweets is usually language-centric, and the models derived from the analysis are often not applicable across distinct languages. Therefore, a classification approach that is not restricted to analyzing a single language would be capable of collecting much more data from Twitter, eventually providing more robust cross-language models. The most common approach for cross-language analysis is called cross-language sentiment classification (CLSC) (Narr, Hulfenhaus, and Albayrak 2012).

Previous work on CLSC mainly focuses on the use of machine translation techniques (Banea et al. 2008; Denecke 2008). These methods operate by translating documents from the source language to the target language and then applying ML classification algorithms over the translated data. However, due to the intrinsic differences between languages, several problems may occur after translation. One of them is word-drifting, in which a word that frequently appears in the source language may rarely appear in the target language after translation, generating a discrepancy in the data distribution between source and target languages (Guo and Xiao 2012). However, even if one could have perfect translation for a particular document set, there is also the potential cultural distance between source and target languages, which may largely influence the final classification performance. Another disadvantage of machine translation approaches is regarding the availability of state-of-the-art open-source translators. Indeed, most of the robust translation APIs are not free of charge, and hence the task of translating large corpora may end up being too expensive.

Deep neural networks have recently achieved significant advancements in different NLP tasks such as language modeling (Bengio et al. 2003), sentiment analysis (Socher et al. 2013), syntactic parsing (Collobert and Weston 2008), and machine translation (Lee, Cho, and Hofmann 2016). Convolutional neural networks (CNNs) and Long short-term memory networks (LSTMs) have been extensively employed for sentiment classification (Zhang, Zhao, and LeCun 2015; Zhang and LeCun 2015; Tang, Qin, and Liu 2015). However, automatic sentiment classification of (unstructured) text data requires documents to be modeled as structured data so they can be interpreted by the ML algorithm. Representing words as dense vectors was first introduced in (Bengio et al. 2003) for the context of neural language modeling, and it was first applied to NLP tasks in the pioneering work of Collobert et al. (Collobert and Weston 2008; Collobert et al. 2011). Several distinct *embedding* strategies for converting free text into vectors have arisen recently, each with their advantages and disadvantages.

Our contributions in this paper are as follows. First, we

present an efficient method for performing multilingual sentiment analysis over tweets. To the best of our knowledge, we are the first to present a multilingual deep neural approach for sentiment analysis that does not rely on machine translation. Second, we extensively compare three different neural architectures in the context of multilingual sentiment analysis, pointing to the advantages and disadvantages of these approaches when classifying tweets from 4 different languages. Each architecture is based on a distinct embedding strategy, and hence demands different computational resources for classifying tweets. Finally, we show that our method presents the best trade-off regarding accuracy and efficiency among the neural architectures that are empirically analyzed, and also regarding traditional SVM-based classification.

Related Work

Sentiment classification techniques can be roughly divided into three approaches: ML-based, lexicon-based, and hybrid (Maynard and Funk 2011). Employing ML algorithms consists in treating sentiment analysis as a common text classification problem that makes use of syntactic and/or linguistic features (Medhat, Hassan, and Korashy 2014). Whereas some approaches identify the aspects that are being discussed together with their polarity (e.g., hotel reviews) (Gamon et al. 2005), others simply assign an overall polarity to the entire document (e.g., movie reviews) (Pang and Lee 2004).

One way of approaching CLSC consists in machine translation strategies to reduce the data to a single language (Banea et al. 2008). Such an approach usually translates training and test data into the same target language, and then applying a monolingual classifier. Nevertheless, classification performance is often negatively affected due to the cultural gap between source and translated languages (Shi, Mihalcea, and Tian 2010). In (Mihalcea, Banea, and Wiebe 2007), the authors use English corpora to train sentence-level subjectivity classifiers in Romanian language using two approaches. In the first approach, they use a bilingual dictionary to translate an existing English lexicon to build a target language subjectivity lexicon. In the second one, they generate a subjectivity-annotated corpus in a target language by projecting annotations from an automatically-annotated English corpus. The authors argue that both approaches can be applied to any language, and not only Romanian. A co-training approach is proposed in (Wan 2009) to leverage resources from both source and target languages. Experiments were conducted on automatic sentiment classification of product reviews in Chinese, successfully making use of both cross-language and within-language knowledge. For a complete review of the state-of-the-art in multilingual sentiment analysis, the reader is referred to (Dashtipour et al. 2016).

Learning Text Representations

One key aspect regarding ML-based sentiment classification relies on the feature space representation. One way of representing words is through dense vectors, where each word is embedded in a d-dimensional vector space (Bengio et al. 2003; Mikolov et al. 2013). Instead of using words, one can also use characters, in a strategy similar to the one used by NLP approaches (Dave, Lawrence, and Pennock 2003; Wiebe et al. 2004; Abbasi, Chen, and Salem 2008). Character-based embedding was recently introduced in the context of convolutional neural networks (dos Santos and Gatti 2014; Zhang and LeCun 2015), and it is said to be better-suited for machine translation than its word-based counterparts (Lee, Cho, and Hofmann 2016).

Word-level Embedding

Word-level embedding consists in mapping word ω onto a d-dimensional space in which semantically-similar words are neighbors. For instance, in single-language analysis, words such as $nice$ and $cool$ should be close to each other within the generated d-dimensional feature space. In the multilingual scenario, this property may be advantageous since similar words in different languages should lie close in the embedded d-dimensional space. However, word-level embedding requires as input a vocabulary with several words, posing two major drawbacks: i) for multilingual analysis, the vocabulary size grows substantially; and ii) *false cognates* – words syntactically identical but with different meanings across languages – will most certainly confuse the ML algorithm and harm classification performance.

Let $\mathcal{T} \in \{\omega_1, \omega_2, ..., \omega_n\}$ be a text with n words and $\phi(\omega_i) = v_i$ be a function that maps word w_i onto vector v_i, the text representation in a word-embedding space is defined by $\Gamma \in \mathbb{R}^{d \times n}$. Note that n varies with the size of text $\mathcal{T}_j$. Figure 1 depicts the word-based representation.

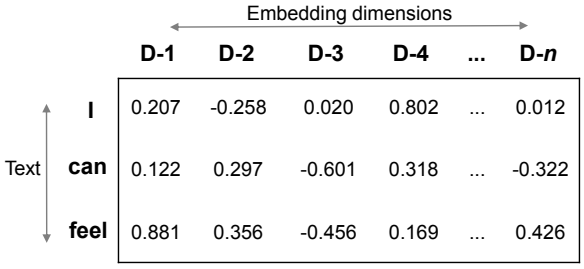

Figure 1: Word-level representation of sentence "I can feel".

For training multilingual classifiers with word-level embedding, one needs to build a large dataset with instances from the desired languages. In the context of Twitter analysis, one can use the original tweets (from their respective languages) without any preprocessing, and it is then possible to classify sentences that contain words from multiple languages at once. For example, a Spanish tweet that contains English cursing can be easily classified, which is not truth when training independent per-language classifiers.

A recent but widely-used approach for modeling sentiment classifiers is based on Recurrent Neural Networks (RNNs), which process text encoded as a sequence of word embeddings (Γ). RNNs are networks that learn recurrent weight matrices for understanding temporal relationships

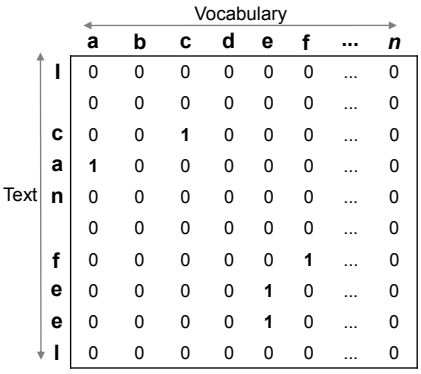

Figure 2: Character-level representation of "I can feel".

among text of variable size. This flexibility is quite attractive for text-based learning such as sentiment analysis. LSTMs (Long Short-term Memory) (Hochreiter and Schmidhuber 1997) are more complex RNN-based architectures that are capable of learning long-term dependencies and forgetting useless information within a given sentence. The LSTM-based approach that we employ in this study for multilingual sentiment analysis is hereby called as **LSTM-Emb**.

Character-level Embedding

A recent approach employs a simple convolutional layer together with a max-pooling-over-time operation instead of LSTMs (Kim 2014). Such architecture is faster than RNNs and seems to provide similar predictive performance. Formally, the convolutional layer convolves Γ with f filters $3 \times d$ resulting in feature maps $\mathcal{M} \in \mathbb{R}^{f \times (n-2)}$. The max-pooling-over-time layer is responsible for selecting the most relevant features within the temporal dimension by using filters of size $1 \times (n-2)$. The resulting representation is $\mathcal{R} \in \mathbb{R}^{f \times 1}$, which is linearly mapped to the C classes. In the context of Twitter sentiment analysis, classes can be the polarities *positive* and *negative*, configuring a softmax output with two neurons. Models that are based in this architecture of convolutional network are hereby referred as **Conv-Emb**.

An alternative way of embedding words into multidimensional spaces is through the use of its basic components, the characters (Zhang and LeCun 2015). In a character-based representation, all characters $c_i \in \{c_1, c_2, ..., c_m\}$ in $\mathcal{T}$ are mapped to a binary matrix of size $m \times \eta$, where $\mathcal{V}$ is the alphabet with η characters, as presented in Figure 2. This representation is based on a 2D fixed-size input tensor for encoding text. The original architecture proposed in (Zhang and LeCun 2015), hereafter called **Conv-Char**, comprises several convolutional layers that act as an embedding learning step, with the advantage of requiring only a small alphabet in memory instead of a large vocabulary.

Even though **Conv-Char** comprises more convolutional layers than **Conv-Emb**, it requires fewer parameters to be learned. In addition, we propose in this paper a novel character-based architecture, namely **Conv-Char-R**, that contains about $6\times$ less parameters than **Conv-Char** without significance performance degradation. Figure 3 depicts both **Conv-Char** and **Conv-Char-R** architectures.

Table 1: Amount of parameters (considering word-embedding as trainable parameters) and memory required for storing both data and model for each method.

Network	# Parameters	Memory
LSTM-Emb (Hochreiter and Schmidhuber 1997)	$\approx 49M$	$\approx 1.2GB$
Conv-Emb (Kim 2014)	$\approx 47M$	$\approx 1GB$
Conv-Char (Zhang, Zhao, and LeCun 2015)	$\approx 3M$	$\approx 0.30GB$
Conv-Char-R [Ours]	$\approx 0.53M$	$\approx 0.32GB$

Conv-Char-R is based on the use of 1×1 convolutional filters to improve the non-linearity within the model while reducing the dimensionality of the tensors, resulting in fewer parameters. We modify the **Conv-Char** structure by adding 1×1 convolutions after each convolutional layer except for the last one. We also removed the fully-connected (dense) layers, mapping the max-pooled tensors (256×4) to the two classes. We also slightly changed the second convolutional layer by using 1×3 filters instead of the original 1×7, requiring $\approx 2.4\times$ fewer parameters to be learned in that layer. Table 1 shows the total number of parameters for each deep neural architecture. Note that **Conv-Char-R** contains only $\approx 527,000$ learnable parameters, as opposed to $49,000,000$ (**LSTM-Emb**) and $47,000,000$ (**Conv-Emb**). In addition, it has 6 times fewer parameters than **Conv-Char**. We show in the next sections that **Conv-Char-R** presents the best trade-off regarding model complexity and predictive performance among the 4 architectures.

Experimental Setup

Dataset

During the experimental analysis, we make use of the Twitter data from (Mozetič, Grčar, and Smailović 2016) to evaluate the proposed architecture. It contains data from 13 European languages, around 1.6 million annotated tweets, which is by far the largest corpora made publicly available so far. All tweets have been manually labeled into three classes: *positive*, *neutral*, and *negative*. Due to the semantic and syntactic structural differences among the 13 languages, we only consider tweets from four specific languages: English, Spanish, Portuguese, and German. We also reduce the problem to binary classification, i.e., we discard all neutral tweets. Note that the dataset does not provide the tweet itself, but rather a URL that leads to the tweets. Due to this particularity, some tweets are no longer available. Statistics of the data used in this work are presented in Table 2.

Baseline Algorithms and Hyper-Parameters

We compare **Conv-Char-R** with **LSTM-Emb**, **Conv-Emb**, **Conv-Emb-Freeze** (a variant of **Conv-Emb** in which we do not update the word embedding), **Conv-Char**, and with a traditional SVM approach (Vapnik and Cortes 1995; Vapnik 1999). The parameters used by each method are as follows.

For **LSTM-Emb**, we use the default LSTM implementation with input and forget gates, though discarding peephole connections. The LSTM comprises a single hidden

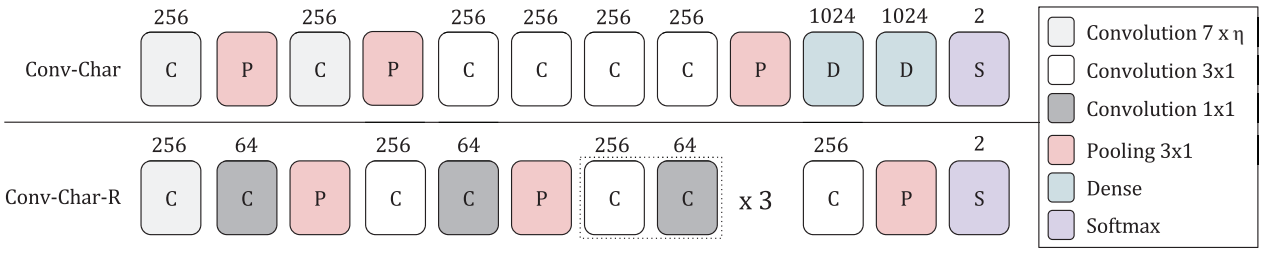

Figure 3: Architectures based on character-level embedding: **Conv-Char** (Zhang and LeCun 2015) and **Conv-Char-R** (ours).

Table 2: Tweet corpora. Each of the sets contains positive and negative tweets from 4 languages.

	Training		Validation		Test		
Language	Negative	Positive	Negative	Positive	Negative	Positive	Total
English	7,784	7,645	1,131	1,146	2,200	2,264	22,170
German	7,502	10,727	1,063	1,544	2,057	2,944	25,837
Portuguese	17,170	12,202	2,427	1,745	4,990	3,560	42,094
Spanish	8,211	18,491	1,189	2,574	2,372	5,251	38,088
Multilingual	40,667	49,065	5,810	7,009	11,619	14,019	128,189
Share of entire corpora	89,732 (70%)		12,819 (10%)		25,638 (20%)		

layer with 512 $tanh$ neurons. We set the embedding size to $d = 300$, and use dropout of 0.8 in the final layers.

Regarding **Conv-Emb**, we set the embedding size to $d = 300$ as in (Kim 2014). The word embedding are initialized with random values and then optimized through backpropagation during training. The vocabulary size is set to $157,000$ words, composed by words in the training set from the four languages. We use 100 convolutional filters and dropout of 0.8 in the final dense layer. The learning rate is set to 1×10^{-4}. The variant **Conv-Emb-Freeze** has the same parameters than **Conv-Emb** though the word embeddings are not updated throughout the training process.

For **Conv-Char**, it takes as input $m = 140$ characters and accepts an alphabet size $\eta = 73$. Given the large amount of parameters and its capability of quickly overfitting the training data, we apply dropout rates of 0.8 and 0.9 in the two final layers of the network, respectively. Learning rate is set to $\alpha = 10^{-3}$. **Conv-Char-R** has the same hyperparameters, and dropout of 0.8 is applied over its single dense layer.

For all neural networks, we employ the Adam update rule for optimization with mini-batches of 128 training instances. Since the convolutional architectures comprise ReLU neurons, we initialize their weight matrices with the procedure described in (He et al. 2015). We only use dropout to regularize dense layers. All models are trained with the negative log-likelihood loss function over a softmax layer that comprises two neurons (positive and negative).

The SVM baseline employs a Gaussian kernel over the features, whose representation is based on unigrams, bigrams, and trigrams. We do not remove stopwords nor execute any kind of language preprocessing since our goal is to verify the effectiveness of each approach over the embedded data without data preprocessing.

Evaluation Criteria

We evaluate the experiments with two traditional classification performance measures: *accuracy* and $F1$-*score*. Accuracy is the rate of correctly classified instances ($(TP + TN)/TP + TN + FP + FN$) whereas $F1$-score is the harmonic mean of precision ($TP/(TP + FP)$) and recall ($TP/(TP+FN)$), as defined in Equation 1. Both measures range within $[0, 1]$, with better scores at higher values.

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} \quad (1)$$

Experimental Results and Discussion

We compare four different deep neural network architectures, as well as SVM-based approaches regarding their predictive performance in the multilingual Twitter dataset: **LSTM-Emb**, **Conv-Emb[-Freeze]**, **Conv-Char**, **Conv-Char-R**, and **SVM-[U,B,T]**. For the SVM-based models, we evaluate the performance of using only unigrams (SVM-U), only bigrams (SVM-b), only trigrams (SVM-t), and combinations among them (SVM-UB, SVM-UBT). Results of accuracy and $F1$-score are presented in Table 3. We also show the per-language results for the neural architectures, so we can evaluate if there is significant variation of results in a language basis (see Tables 4 and 5).

Note that the best $F1$-score is achieved by **LSTM-Emb**, 0.753, and that the LSTM model also achieves the second best accuracy value, 0.713, losing only to the **Conv-Emb** model. Note that the SVM-based models with *n*-grams do not perform nearly as good as the neural architectures. These poor results are most certainly due to the lack of preprocessing techniques (e.g., stopwords removal or stemming) over the input data. Preprocessing was not applied to any of the

Table 3: Accuracy and $F1$-score values.

Method	Accuracy	$F1$-score
SVM-U	0.558	0.424
SVM-UB	0.548	0.433
SVM-UBT	0.549	0.433
SVM-B	0.502	0.032
SVM-T	0.511	0.057
Conv-Emb-Freeze	0.688	0.728
Conv-Emb	**0.714**	0.749
LSTM-Emb	0.713	**0.753**
Conv-Char	0.696	0.721
Conv-Char-R	0.695	0.722

methods so the evaluation could be as fair as possible, and we could clearly verify the ability of each model when directly dealing with embedded text.

Regarding the convolutional models that employ different text representations, one can observe that word-embedding models seem to perform slightly better than character-based models. Also, note that learning the embedding through backpropagation is always better than freezing the embedding and then only learning the remaining parameters. However, as presented in Table 1, the excessively large amount of parameters to be learned in word-embedding-based models is a prominent disadvantage when adopting this representation. The character-based models are much cheaper and their slightly inferior performance does not seem to justify the use of the much larger word-based models.

Table 4: Per-language $F1$-score.

Method	English	German	Portuguese	Spanish
Conv-Emb-Freeze	0.736	0.752	0.583	0.795
Conv-Emb	0.763	0.782	0.608	0.804
LSTM-Emb	**0.779**	**0.783**	**0.619**	**0.806**
Conv-Char	0.729	0.758	0.589	0.776
Conv-Char-R	0.722	0.755	0.605	0.774

Table 5: Per-language accuracy values.

Method	English	German	Portuguese	Spanish
Conv-Emb-Freeze	0.734	0.694	0.671	0.678
Conv-Emb	0.766	0.733	**0.695**	0.693
LSTM-Emb	**0.770**	**0.734**	0.686	**0.697**
Conv-Char	0.736	0.718	0.688	0.668
Conv-Char-R	0.727	0.721	0.694	0.660

Regarding our proposed approach, **Conv-Char-R**, which is an optimized reduced version of **Conv-Char**, one can see that it presents virtually the same performance than its larger version, arguably presenting the best trade-off in terms of predictive performance and amount of parameters among all neural architectures. Indeed, **Conv-Char-R** is only $\approx 2\%$ short of the best model (**LSTM-Emb**) in terms of accuracy and $\approx 3\%$ in terms of $F1$-score, while containing ≈ 90 times fewer parameters than the word-based models, and consuming ≈ 4 times less memory.

Finally, we argue on the advantages of the multilingual strategy rather than using one classifier per language. For the latter, one would need to train a prior classifier that first detects the language of the tweet and then redirects it to the proper per-language classifier. This cascade of classifiers enhances the cost of the classification process and may eventually deteriorate the final results, since the incorrect identification of the language will harm the chances of correctly classifying the sentiment of the tweet. Hence, our multilingual approach seems to be much more reasonable, since it uses a single classifier that can understand any of the languages that were used during training. Another advantage of the multilingual setup is that the tweet to be analyzed can contain terms from distinct languages without affecting the classification process, which is not true for the per-language classification approach.

Conclusions

In this paper, we have presented an efficient method for performing multilingual sentiment analysis over tweets in four languages. This is the first approach that uses character-based models with Convolutional Neural Networks in the multilingual scenario. Our approach does not rely on machine translation to perform multilingual sentiment classification. We also present a comparison of distinct neural models in multilingual sentiment analysis and we empirically test different convolutional architectures to find the most efficient set of parameters. Our results indicate that the proposed architecture, **Conv-Char-R**, reaches competitive results when compared with state-of-the-art deep neural models for sentiment classification, with the main advantage of containing ≈ 90 times fewer parameters than word-based models and consuming ≈ 4 times less memory. In future work, we plan to investigate the impact of including more languages in the Twitter multilingual corpora, and we intend to compare **Conv-Char-R** with traditinal machine-translation-based approaches.

Acknowledgments

The authors would like to thank CAPES, FAPERGS, CNPq, and Google for funding this research.

References

Abbasi, A.; Chen, H.; and Salem, A. 2008. Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums. *ACM Transactions on Information Systems (TOIS)* 26(3):12.

Banea, C.; Mihalcea, R.; Wiebe, J.; and Hassan, S. 2008. Multilingual subjectivity analysis using machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 127–135.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *JMLR* 3(Feb):1137–1155.

Collobert, R., and Weston, J. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th international conference on Machine learning*, 160–167.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12(Aug):2493–2537.

Dashtipour, K.; Poria, S.; Hussain, A.; Cambria, E.; Hawalah, A. Y. A.; Gelbukh, A.; and Zhou, Q. 2016. Multilingual sentiment analysis: State of the art and independent comparison of techniques. *Cognitive Computation* 8(4):757–771.

Dave, K.; Lawrence, S.; and Pennock, D. M. 2003. Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, 519–528.

Denecke, K. 2008. Using sentiwordnet for multilingual sentiment analysis. In *IEEE 24th International Conference on Data Engineering*, 507–512.

dos Santos, C. N., and Gatti, M. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, 69–78.

Gamon, M.; Aue, A.; Corston-Oliver, S.; and Ringger, E. 2005. Pulse: Mining customer opinions from free text. In *International Symposium on Intelligent Data Analysis*, 121–132.

Guo, Y., and Xiao, M. 2012. Cross language text classification via subspace co-regularized multi-view learning. *arXiv preprint arXiv:1206.6481*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE International Conference on Computer Vision*, 1026–1034.

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

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Lee, J.; Cho, K.; and Hofmann, T. 2016. Fully character-level neural machine translation without explicit segmentation. *arXiv preprint arXiv:1610.03017*.

Martínez-Cámara, E.; Martín-Valdivia, M. T.; Urena-López, L. A.; and Montejo-Ráez, A. R. 2014. Sentiment analysis in twitter. *Natural Language Engineering* 20(01):1–28.

Maynard, D., and Funk, A. 2011. Automatic detection of political opinions in tweets. In *Extended Semantic Web Conference*, 88–99.

Medhat, W.; Hassan, A.; and Korashy, H. 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams Engineering Journal* 5(4):1093–1113.

Mihalcea, R.; Banea, C.; and Wiebe, J. M. 2007. Learning multilingual subjective language via cross-lingual projections.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Mozetič, I.; Grčar, M.; and Smailović, J. 2016. Multilingual twitter sentiment classification: The role of human annotators. *PloS one* 11(5):e0155036.

Narr, S.; Hulfenhaus, M.; and Albayrak, S. 2012. Language-independent twitter sentiment analysis. *Knowledge Discovery and Machine Learning (KDML), LWA* 12–14.

Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, 271.

Shi, L.; Mihalcea, R.; and Tian, M. 2010. Cross language text classification by model translation and semi-supervised learning. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, 1057–1067.

Socher, R.; Perelygin, A.; Wu, J. Y.; Chuang, J.; Manning, C. D.; Ng, A. Y.; and Potts, C. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, volume 1631, 1642.

Tang, D.; Qin, B.; and Liu, T. 2015. Document modeling with gated recurrent neural network for sentiment classification. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1422–1432.

Vapnik, V., and Cortes, C. 1995. Support vector networks. *Machine Learning* 20:273.

Vapnik, V. 1999. An overview of statistical learning theory. *IEEE Transactions on Neural Networks* 10:988–999.

Wan, X. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, 235–243.

Wiebe, J.; Wilson, T.; Bruce, R.; Bell, M.; and Martin, M. 2004. Learning subjective language. *Computational linguistics* 30(3):277–308.

Zhang, X., and LeCun, Y. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710*.

Zhang, X.; Zhao, J.; and LeCun, Y. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 649–657.