A Deep Neural Architecture for Sentence-level Sentiment Classification in Twitter Social Networking

Huy Nguyen* and Minh-Le Nguyen*
*Japan Advanced Institute of Science and Technology
Ishikawa, Japan
{huy.nguyen; nguyenml}@jaist.ac.jp

***Abstract*—This paper introduces a novel deep learning framework including a lexicon-based approach for sentence-level prediction of sentiment label distribution. We propose to first apply semantic rules and then use a Deep Convolutional Neural Network (DeepCNN) for character-level embeddings in order to increase information for word-level embedding. After that, a Bidirectional Long Short-Term Memory network (Bi-LSTM) produces a sentence-wide feature representation from the word-level embedding. We evaluate our approach on three twitter sentiment classification datasets. Experimental results show that our model can improve the classification accuracy of sentence-level sentiment analysis in Twitter social networking.**

***Keywords*-Twitter; Sentiment classification.**

I. Introduction

Twitter sentiment classification have intensively researched in recent years [1] [2]. Different approaches were developed for Twitter sentiment classification by using machine learning such as Support Vector Machine (SVM) with rule-based features [3] and the combination of SVMs and Naive Bayes (NB) [4]. In addition, hybrid approaches combining lexicon-based and machine learning methods also achieved high performance described in [5]. However, a problem of traditional machine learning is how to define a feature extractor for a specific domain in order to extract important features.

Deep learning models are different from traditional machine learning in that a deep learning model does not depend on feature extractors because features are extracted during training progress. The use of deep learning methods becomes to achieve remarkable results for sentiment analysis [6][7][8]. Some researchers used Convolutional Neural Network (CNN) for sentiment classification. CNN models have been shown to be effective for NLP. For example, [7] proposed various kinds of CNN to learn sentiment-bearing sentence vectors, [6] adopted two CNNs in character-level to sentence-level representation for sentiment analysis. [8] constructs experiments on a character-level CNN for several large-scale datasets. In addition, Long Short-Term Memory (LSTM) is another state-of-the-art semantic composition model for sentiment classification with many variants described in [9]. The studies reveal that using a CNN is useful in extracting information and finding feature detectors from texts. In addition, a LSTM can be good in maintaining word order and the context of words. However, in some important aspects, the use of CNN or LSTM separately may not capture enough information.

Inspired by the models above, the goal of this research is using a Deep Convolutional Neural Network (DeepCNN) to exploit the information of characters of words in order to support word-level embedding. A Bi-LSTM produces a sentence-wide feature representation based on these embeddings. The Bi-LSTM is a version of [10] with Full Gradient described in [11]. In addition, the rules-based approach also effect classification accuracy by focusing on important sub-sentences expressing the main sentiment of a tweet while removing unnecessary parts of a tweet. The paper makes the following contributions:

- We construct a tweet processor removing unnecessary sub-sentences from tweets to help the model to learn important information in a tweet. We share ideas with [1] and [12], however, our tweet processor keeps emoticons in tweets and only uses rules to remove non-essential parts for handling negation.
- We train a DeepCNN on top of character embeddings to produce feature maps that capture the morphological and shape information of a word. The morphological and shape information illustrate how words are formed, and their relationship to other words. Thanks to DeepCNN, the character-level representation is transformed into global fixed-sized feature vectors at higher abstract level. Such character feature vectors contribute enriching the information of words in a sentence.
- We create an integration of global fixed-size character feature vectors and word-level embedding for the Bi-LSTM carefully. The Bi-LSTM connects the information of words in a sequence and maintains the order of words for a sentence-level representation.

The organization of the present paper is as follows: In section 2, we describe model architecture which introduces the structure of model. We explain the basic idea of model and the way of constructing the model. Section 3 show results and analysis and section 4 summarize this paper.

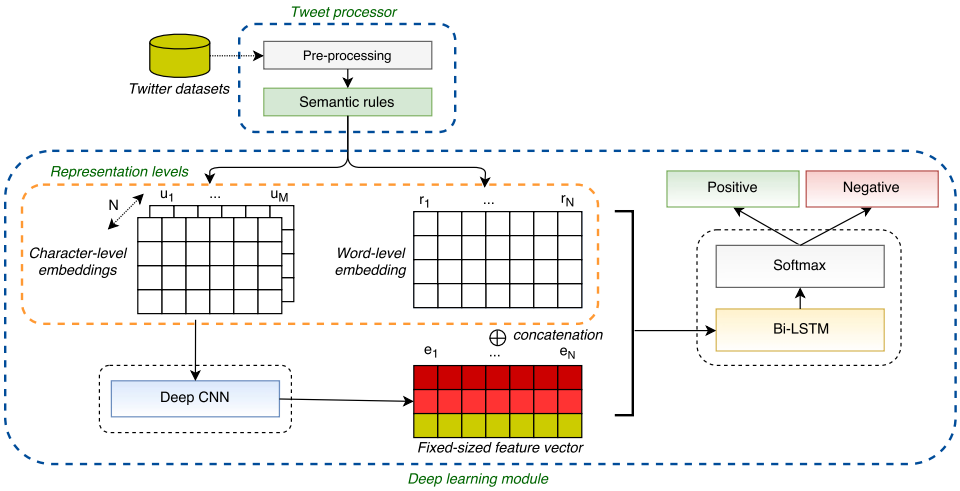

Figure 1. The overview of a deep learning system.

II. Model Architecture

A. Basic idea

Our proposed model consists of a deep learning classifier and a tweet processor. The deep learning classifier is a combination of a DeepCNN and a Bi-LSTM. The tweet processor standardizes tweets and then applies rules called semantic rules on dataset. We construct a framework that treats the deep learning classifier and the tweet processor as two distinct components. We believe that standardizing data is an important step to achieve high accuracy. To formulate our problem in increasing the accuracy of the classifier, we illustrate our model in Figure. 1 as follows:

1) Tweets are firstly considered via a processor based on preprocessing steps [1] and the semantic rules-based method [12] in order to standardize tweets and capture only important information containing the main sentiment of a tweet.
2) We use DeepCNN for character-level embeddings. A wide convolution can learn to recognize specific *n-grams* at every position in a word that allows features to be extracted independently of these positions in the word. These features maintain the order and relative positions of characters. A DeepCNN is constructed by two wide convolution layers and the need of multiple convolution layers is widely accepted that a model constructing by multiple processing layers have the ability to learn representations of data with higher levels of abstraction [13]. Therefore, we use DeepCNN for character-level embeddings to support morphological and shape information for a word. The DeepCNN produces global fixed-sized feature vectors for words.
3) A combination of the global fixed-size feature vectors and word-level embedding is fed into a Bi-LSTM. The Bi-LSTM produces a sentence-level representation by maintaining the order of words.

Our work is philosophically similar to [6]. However, our model is distinguished with their approaches in two aspects:

- Using DeepCNN with two wide convolution layers to increase representation with multiple levels of abstraction.
- Integrating global character fixed-sized feature vectors with word-level embedding to extract a sentence-wide feature set via Bi-LSTM. This deals with three main problems: (i) Sentences have any different size; (ii) The semantic and the syntactic of words in a sentence are captured in order to increase information for a word; (iii) Important information of characters that can appear at any position in a word are extracted.

In sub-section B, we introduce various kind of dataset. The modules of our model are constructed in other sub-sections.

B. Data Preparation

- *Stanford - Twitter Sentiment Corpus (STS Corpus):* STS Corpus contains 1,600K training tweets collected by a crawler from [1]. [1] constructed a test set manually with 177 negative and 182 positive tweets. Although the Stanford test set is small but it has been widely used in different evaluation tasks [1] [6] [14].
- *Sanders - Twitter Sentiment Corpus:* This dataset consists of hand-classified tweets collected by using search terms: *#apple, #google, #microsoft and #twitter*. We construct the dataset as [15] for binary classification.
- *Health Care Reform (HCR):* This dataset was constructed by crawling tweets containing the hashtag *#hcr* [16]. Task is to predict positive/negative tweets [15].

Table I
SEMANTIC RULES [12]

Rule	Semantic rules	Example - STS Corpus	Output
R11	If a sentence contains "but", disregard all previous sentiment and only take the sentiment of the part after "but".	@kirstiealley my dentist is great *but* she's expensive...=(	she's expensive...=(
R12	If a sentence contains "despite", only take sentiment of the part before "despite".	I'm not dead *despite* rumours to the contrary.	I'm not dead
R13	If a sentence contains "unless", and "unless" is followed by a negative clause, disregard the "unless" clause.	laptop charger is broken - *unless* a little cricket set up home inside it overnight. typical at the worst possible time.	laptop charger is broken
R14	If a sentence contains "while", disregard the sentence following the "while" and take the sentiment only of the sentence that follows the one after the "while".	My throat is killing me, and *While* I got a decent night's sleep last night, I still feel like I'm about to fall over.	I still feel like I'm about to fall over
R15	If the sentence contains "however", disregard the sentence preceding the "however" and take the sentiment only of the sentence that follows the "however".	@lonedog bwahahah...you are amazing! *However*, it was quite the letdown.	it was quite the letdown.

Table II
THE NUMBER OF TWEETS ARE PROCESSED BY USING SEMANTIC RULES

Dataset	**Set**	**# Sentences/ tweets**
STS Corpus	Train	138703
	Test	25
Sanders	Train	39
	Dev	74
	Test	54
HCR	Train	164

C. Preprocessing

We firstly take unique properties of Twitter in order to reduce the feature space such as *Username*, *Usage of links*, *None*, *URLs* and *Repeated Letters*. We then process *retweets*, *stop words*, *links*, *URLs*, *mentions*, *punctuation* and *accentuation*. For emoticons, [1] revealed that the training process makes the use of emoticons as noisy labels and they stripped the emoticons out from their training dataset because [1] believed that if we consider the emoticons, there is a negative impact on the accuracies of classifiers. In addition, removing emoticons makes the classifiers learns from other features (e.g. unigrams and bi-grams) presented in tweets and the classifiers only use these non-emoticon features to predict the sentiment of tweets. However, there is a problem is that if the test set contains emoticons, they do not influence the classifiers because emoticon features do not contain in its training data. This is a limitation of [1], because the emoticon features would be useful when classifying test data. Therefore, we keep emoticon features in both datasets because emoticons can help deep learning models to capture more information for increasing classification accuracy.

D. Semantic Rules (SR)

In Twitter social networking, people express their opinion with sub-sentences. These sub-sentences using specific PoS particles (Conjunction and Conjunctive adverbs), like *"but, while, however, despite, however"* have different polarities. However, the overall sentiment of tweets often focus on certain sub-sentences. For example:

- *@lonedog bwahahah...you are amazing! However, it was quite the letdown.*
- *@kirstiealley my dentist is great but she's expensive...=(*

In two tweets above, the overall sentiment is negative. However, the main sentiment is only on the sub-sentence following *but* and *however*. This inspires a processing step to remove unessential parts in a tweet. Rule-based approach can assists these problems in handling negation and deal with specific PoS particles led to effectively affect the final output of classification [12][17]. They summarized a full presentation of their semantic rules approach and devised ten semantic rules in their hybrid approach based on the presentation of [17]. We use five rules in the semantic rules set because other five rules are only used to compute polarity of words after POS tagging or Parsing steps. We follow the same naming convention for rules utilized by [12] to represent the rules utilized in our proposed method. The rules utilized in the proposed method are displayed in Table I in which is included examples from STS Corpus and output after using rules. Table II illustrates the number of processed sentences on each dataset.

E. Representation Levels

To construct embedding inputs for our model, we use a fixed-sized word vocabulary V^{word} and a fixed-sized character vocabulary V^{char}. Given a word w_i is composed from characters $\{c_1, c_2, ..., c_M\}$, the character-level embeddings are encoded by column vectors u_i in the embedding matrix $W^{char} \in R^{d^{char} \times |V^{char}|}$, where V^{char} is the size of the character vocabulary. For word-level embedding r_{word}, we use a pre-trained word-level embedding with dimension 200 or 300. A pre-trained word-level embedding can capture the syntactic and semantic information of words [18]. We build every word w_i into an embedding $v_i = [r_i; e_i]$ which is constructed by two sub-vectors: the word-level embedding $r_i \in R^{d^{word}}$ and the character fixed-size feature vector $e_i \in R^l$ of w_i where l is the length of the filter of convolutions. We have N character fixed-size feature vectors

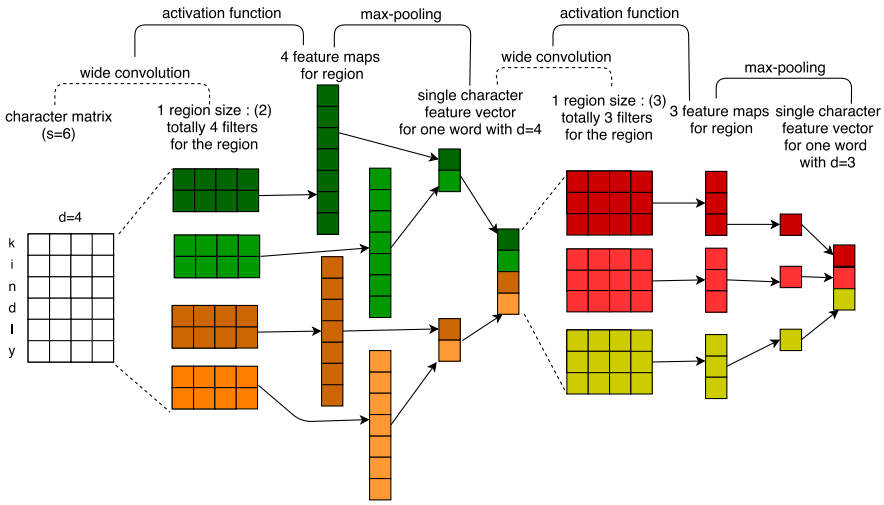

Figure 2. Deep Convolutional Neural Network (DeepCNN) for the sequence of character embeddings of a word. For example with 1 region size is 2 and 4 feature maps in the first convolution and 1 region size is 3 with 3 feature maps in the second convolution.

corresponding to word-level embedding in a sentence.

F. Deep Learning Module

DeepCNN in the deep learning module is illustrated in Figure. 2. The DeepCNN has two wide convolution layers. The first layer extract local features around each character windows of the given word and using a max pooling over character windows to produce a global fixed-sized feature vector for the word. The second layer retrieves important context characters and transforms the representation at previous level into a representation at higher abstract level. We have N global character fixed-sized feature vectors for other words.

In the next step of Figure. 1, we construct the vector $v_i = [r_i, e_i]$ by concatenating the word-level embedding with the global character fixed-size feature vectors. The input of Bi-LSTM is a sequence of embeddings $\{v_1, v_2, ..., v_N\}$. The use of the global character fixed-size feature vectors increases the relationship of words in the word-level embedding. The purpose of this Bi-LSTM is to capture the context of words in a sentence and maintain the order of words toward to extract sentence-level representation. The top of the model is a softmax function to predict sentiment label. We describe in detail the kinds of CNN and LSTM that we use in next parts.

1) Convolutional Neural Network: The one-dimensional convolution called time-delay neural net has a filter vector m and take the dot product of filter m with each *m-grams* in the sequence of characters $s_i \in R$ of a word in order to obtain a sequence c:

$$c_j = m^T s_{j-m+1:j} \tag{1}$$

Based on Equation 1, we have two types of convolutions that depend on the range of the index j. The narrow type of convolution requires that $s \geq m$ and produce a sequence $c \in R^{s-m+1}$. The wide type of convolution does not require on s or m and produce a sequence $c \in R^{s+m-1}$. Out-of-range input values s_i where $i < 1$ or $i > s$ are taken to be zero. We use wide convolution for our model.

Wide Convolution: Given a word w_i composed of M characters $\{c_1, c_2, ..., c_M\}$, we take a character embedding $u_i \in R^d$ for each character c_i and construct a character matrix $W^{char} \in R^{d \times |V^{chr}|}$ as following Equation. 2:

$$W^{char} = \begin{bmatrix} | & | & | \\ u_1 & ... & u_M \\ | & | & | \end{bmatrix} \tag{2}$$

The values of the embeddings u_i are parameters that are optimized during training. The trained weights in the filter m correspond to a feature detector which learns to recognize a specific class of *n-grams*. The *n-grams* have size $n \geq m$. The use of a wide convolution has some advantages more than a narrow convolution because a wide convolution ensures that all weights of filter reach the whole characters of a word at the margins. The resulting matrix has dimension $d \times (s + m - 1)$.

2) Long Short-Term Memory: Long Short-Term Memory networks usually called LSTMs are a improved version of RNN. The core idea behind LSTMs is the cell state which can maintain its state over time, and non-linear gating units which regulate the information flow into and out of the cell. The LSTM architecture that we used in our proposed model is described in [10]. A single LSTM memory cell is implemented by the following composite function:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{3}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{4}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (6)$$

$$h_t = o_t tanh(c_t) \quad (7)$$

where σ is the logistic sigmoid function, i, f, o and c are the *input gate, forget gate, output gate, cell* and *cell input* activation vectors respectively. All of them have a same size as the hidden vector h. W_{hi} is the hidden-input gate matrix, W_{xo} is the input-output gate matrix. The bias terms which are added to i, f, c and o have been omitted for clarity. In addition, we also use the full gradient for calculating with full backpropagation through time (BPTT) described in [11]. A LSTM gradients using finite differences could be checked and making practical implementations more reliable.

G. Regularization

For regularization, we use a constraint on $l_2 - norms$ of the weight vectors [19].

III. Results and Analysis

A. Experimental setups

For the Stanford Twitter Sentiment Corpus, we use the number of samples as [6]. The training data is selected 80K tweets for a training data and 16K tweets for the development set randomly from the training data of [1]. We conduct a binary prediction for STS Corpus.

For Sander dataset, we use standard 10-fold cross validation as [15]. We construct the development set by selecting 10% randomly from 9-fold training data.

In Health Care Reform Corpus, we also select 10% randomly for the development set in a training set and construct as [15] for comparison. We describe the summary of datasets in Table III.

Table III
SUMMARY STATISTICS FOR THE DATASETS AFTER USING SEMANTIC RULES. c: THE NUMBER OF CLASSES. N: THE NUMBER OF TWEETS. l_w: MAXIMUM SENTENCE LENGTH. l_c: MAXIMUM CHARACTER LENGTH. $|V_w|$: WORD ALPHABET SIZE. $|V_c|$: CHARACTER ALPHABET SIZE.

Data	Set	N	c	l_w	l_c	$\|V_w\|$	$\|V_c\|$	*Test*
STS	Train	80K	2	33	110	67083	134	-
	Dev	16K		28	48			
	Test	359		21	16			
Sanders	Train	991	2	31	33	3379	84	CV
	Dev	110		27	47			
	Test	122		28	21			
HCR	Train	621	2	25	70	3100	60	-
	Dev	636		26	16			
	Test	665		20	16			

1) Hyperparameters: for all datasets, the filter window size (h) is 7 with 6 feature maps each for the first convolution layer, the second convolution layer has a filter window size of 5 with 14 feature maps each. Dropout rate (p) is 0.5, l_2 constraint and mini-batch size of 100. In addition, learning rate is 0.1 and momentum of 0.9. In addition, training is done through stochastic gradient descent over shuffled mini-batches with Adadelta update rule [20].

2) Pre-trained Word Vectors: we use the publicly available *Word2Vec* [1] trained from 100 billion words from Google and *TwitterGlove* [2] of Stanford is performed on aggregated global word-word co-occurrence statistics from a corpus. *Word2Vec* has dimensionality of 300 and *Twitter Glove* have dimensionality of 200. Words that do not present in the set of pre-train words are initialized randomly.

Table IV
ACCURACY OF DIFFERENT MODELS FOR BINARY CLASSIFICATION

Model	STS	Sanders	HCR
CharSCNN/ pre-training [6]	**86.4**	-	-
CharSCNN/ random [6]	81.9	-	-
SCNN/ pre-training [6]	85.2	-	-
SCNN/ random [6]	82.2	-	-
MaxEnt [1]	83.0	-	-
NB [1]	82.7	-	-
SVM [1]	82.2	-	-
SVM-BoW	-	82.43	73.99
SVM-BoW + lex	-	83.98	75.94
RF-BoW	-	79.24	70.83
RF-BoW + lex	-	82.35	72.93
LR-BoW	-	77.45	73.83
LR-BoW + lex	-	79.49	74.73
MNB-BoW	-	79.82	72.48
MNB-BoW + lex	-	83.41	75.33
ENS (RF + MNB + LR) - BoW	-	-	75.19
ENS (SVM + RF + MNB + LR) - BoW + lex	-	-	**76.99**
ENS (SVM + RF + MNB + LR) - BoW	-	82.76	-
ENS (SVM + RF + MNB) - BoW + lex	-	**84.89**	-
DeepCNN + SR + Glove	85.23	62.38	76.84
Bi-LSTM + SR + Glove	85.79	84.32	78.49
(DeepCNN + Bi-LSTM) + SR + Glove	**86.63**	**85.14**	79.55
(DeepCNN + Bi-LSTM) + SR + GoogleW2V	86.35	85.05	**80.9**
(DeepCNN + Bi-LSTM) + GoogleW2V	86.07	84.23	80.75

B. Experimental results

Table IV shows the result of our model for sentiment classification against other models. We compare our model performance with the approaches of [1][6] on STS Corpus. [1] reported the results of Maximum Entropy (MaxEnt), NB, SVM on STS Corpus having good performance in previous time. The model of [6] is a state-of-the-art so far by using a CharSCNN. As can be seen, *86.63* is the best prediction accuracy of our model so far for the STS Corpus.

For Sanders and HCR datasets, we compare results with the model of [15] that used a ensemble of multiple base classifiers (ENS) such as NB, Random Forest (RF), SVM and Logistic Regression (LR). The ENS model is combined with bag-of-words (BoW), feature hashing (FH) and lexicons. [15] model is a state-of-the-art on Sanders and HCR datasets. Our models outperform the model of [15] for the Sanders dataset and HCR dataset.

[1]https://code.google.com/p/word2vec/
[2]https://nlp.stanford.edu/projects/glove/

Table V
THE LABEL PREDICTION BETWEEN THE MODEL USING SEMANTIC RULES AND THE MODEL WITHOUT SEMANTIC RULES

Model	Inputs from HCR dataset	Label	Predict
(DeepCNN + Bi-LSTM) without SR	#hcr #whitehouse #Obama - To stay alive in this country, isn't war in another country, but war right here to live without proper care.	Positive	False
	i'm not necessarily against #hcr, but i feel ignored as a late-twenties, single, childless woman above the poverty line & well under wealthy.	Negative	False
	#Stupak to vote "Yes" on #HCR - but in exchange for what? #underthebus	Negative	False
(DeepCNN + Bi-LSTM) using SR	war right here to live without proper care	Positive	True
	i feel ignored as a late-twenties, single, childless woman above the poverty line & well under wealthy.	Negative	True
	in exchange for what? #underthebus	Negative	True

C. Analysis

As can be seen, the models with SR outperforms the model with no SR. Semantic rules are effective in order to increase classification accuracy. Table V describes the comparison between the model using SR and the model without SR. In Table V, we list examples with its true labels that our model without SR fails in prediction and the red detectors affect the model to fail in prediction. The semantic rules assist our model to succeed in predicting true labels. We also conduct two experiments on two separate models: DeepCNN and Bi-LSTM in order to show the effectiveness of combination of DeepCNN and Bi-LSTM. In addition, the model using *TwitterGlove* outperform the model using *GoogleW2V* because *TwitterGlove* captures more information in Twitter than *GoogleW2V*. These results show that the character-level information and SR have a great impact on Twitter Data. The pre-train word vectors are good, universal feature extractors. The difference between our model and other approaches is the ability of our model to capture important features by using SR and combine these features at high benefit. The use of DeepCNN can learn a representation of words in higher abstract level. The combination of global character fixed-sized feature vectors and a word embedding helps the model to find important detectors for particles such as 'not' that negate sentiment and potentiate sentiment such as 'too', 'so' standing beside expected features. The model not only learns to recognize single n-grams, but also patterns in n-grams lead to form a structure significance of a sentence.

IV. CONCLUSIONS

In the present work, we have pointed out that the use of character embeddings through a DeepCNN to enhance information for word embeddings built on top of *Word2Vec* or *TwitterGlove* improves classification accuracy in Tweet sentiment classification. Our results add to the well-establish evidence that character vectors are an important ingredient for word-level in deep learning for NLP. In addition, semantic rules contribute handling non-essential sub-tweets in order to improve classification accuracy.

REFERENCES

[1] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *CS224N Project Report, Stanford*, vol. 1, no. 12, 2009.

[2] P. Nakov, A. Ritter, S. Rosenthal, F. Sebastiani, and V. Stoyanov, "SemEval-2016 task 4: Sentiment analysis in twitter," *Proceedings of SemEval*, pp. 1–18, 2016.

[3] J. Silva, L. Coheur, A. C. Mendes, and A. Wichert, "From symbolic to sub-symbolic information in question classification," *Artificial Intelligence Review*, vol. 35, no. 2, pp. 137–154, 2011.

[4] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*. Association for Computational Linguistics, 2012, pp. 90–94.

[5] A. Muhammad, N. Wiratunga, and R. Lothian, "Contextual sentiment analysis for social media genres," *Knowledge-Based Systems*, vol. 108, pp. 92–101, 2016.

[6] C. N. Dos Santos and M. Gatti, "Deep convolutional neural networks for sentiment analysis of short texts." in *COLING*, 2014, pp. 69–78.

[7] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[8] X. Zhang and Y. LeCun, "Text understanding from scratch," *CoRR*, vol. abs/1502.01710, 2015. [Online]. Available: http://arxiv.org/abs/1502.01710

[9] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on*, vol. 3. IEEE, 2000, pp. 189–194.

[10] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[11] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[12] O. Appel, F. Chiclana, J. Carter, and H. Fujita, "A hybrid approach to the sentiment analysis problem at the sentence level," *Knowledge-Based Systems*, vol. 108, pp. 110–124, 2016.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[14] F. Bravo-Marquez, M. Mendoza, and B. Poblete, "Combining strengths, emotions and polarities for boosting twitter sentiment analysis," in *Proceedings of the Second International Workshop on Issues of Sentiment Discovery and Opinion Mining*. ACM, 2013, p. 2.

[15] N. F. F. Da Silva, E. R. Hruschka, and E. R. Hruschka, "Tweet sentiment analysis with classifier ensembles," *Decision Support Systems*, vol. 66, pp. 170–179, 2014.

[16] M. Speriosu, N. Sudan, S. Upadhyay, and J. Baldridge, "Twitter polarity classification with label propagation over lexical links and the follower graph," in *Proceedings of the First workshop on Unsupervised Learning in NLP*. Association for Computational Linguistics, 2011, pp. 53–63.

[17] Y. Xie, Z. Chen, K. Zhang, Y. Cheng, D. K. Honbo, A. Agrawal, and A. N. Choudhary, "MuSES: Multilingual sentiment elicitation system for social media data," *IEEE Intelligent Systems*, vol. 29, no. 4, pp. 34–42, 2014.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26*. Curran Associates, Inc., 2013, pp. 3111–3119.

[19] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *arXiv preprint arXiv:1207.0580*, 2012.

[20] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.