Sentence Modeling with Deep Neural Architecture using Lexicon and Character Attention Mechanism for Sentiment Classification

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

Abstract

Tweet-level sentiment classification in Twitter social networking has many challenges: exploiting syntax, semantic, sentiment and context in tweets. To address these problems, we propose a novel approach to sentiment analysis that uses lexicon features for building lexicon embeddings (LexW2Vs) and generates character attention vectors (CharAVs) by using a Deep Convolutional Neural Network (DeepCNN). Our approach integrates LexW2Vs and CharAVs with continuous word embeddings (ContinuousW2Vs) and dependency-based word embeddings (DependencyW2Vs) simultaneously in order to increase information for each word into a Bidirectional Contextual Gated Recurrent Neural Network (Bi-CGRNN). We evaluate our model on two Twitter sentiment classification datasets. Experimental results show that our model can improve the classification accuracy of sentence-level sentiment analysis in Twitter social networking.

1 Introduction

Tweet-level sentiment classification is a fundamental task of sentiment analysis in Twitter social networking and is essential to understand user generated contents in social networking. Twitter sentiment classification have intensively researched in recent years (Go et al., 2009) (Nakov et al., 2016). There are many works related to deep learning methods involved learning word representation (Socher et al., 2011). Word representation is central to deep learning and essential feature extractor that encode different features of words in their dimensions. The combination of word representation and deep learning achieved impressive results because word embeddings enable efficient computation of word similarities through low-dimensional matrix operations (Kim, 2014). In addition, deep learning models achieved remarkable performance. Some researchers use Convolution Neural Network (CNN) for sentiment classification. CNN utilizes convolution filters applied to local features. CNN models has been shown to be effective for NLP. For example, the model of (Dos Santos and Gatti, 2014) used CNN to form a sentence-level representation for sentiment classification. In addition, Bidirectional Gated Recurrent Neural Network (Bi-GRNN) is another deep learning model that has achieved an excellent result for sentiment analysis and other traditional tasks (Chung et al., 2014).

Inspired by the models above, the goal of this research is to build a model for exploiting syntax, semantic, sentiment and context of tweets by constructing four kinds of embeddings: CharAVs, LexW2Vs, ContinuousW2Vs and DependencyW2Vs. On the other hand, we modify Bi-GRNN of (Chung et al., 2014) into Bi-CGRNN to take word embeddings in order to produce a sentence-wide representation from sentence compositions. Our paper makes the following contributions:

- We construct a tweet processor which standardizes tweets by using pre-processing steps and a semantic rule-based approach. We construct four kinds of embeddings: CharAvs, LexW2Vs, ContinuousW2Vs and DependencyW2Vs. A DeepCNN is used for training CharAVs by producing fixed-size feature vectors and attending on the best feature vectors. CharAVs can capture the morphology and shape of a word. The morphological and shape information illustrate how words are formed, and their relationship to other words.

- We create an integration of CharAVs and LexW2Vs with ContinuousW2Vs and DependencyW2Vs. Such embeddings are advanced continuous word embeddings and advanced dependency-based word embeddings.

- We modify a standard Bi-GRNN to be a Bi-CGRNN by incorporating contextual features (e.g., Syntactic contexts) in order to take both advanced word embeddings. The output of Bi-CGRNN is sentence compositions that are formed into a sentence-wide representation. The purpose of Bi-CGRNN is to connect the information of words in a sequence and maintain the order of words for a sentence-level representation.

The organization of the paper is as follows: Section 2 describes the model architecture which introduces the structure of the model. We explain the basic idea of the model and the way of constructing the model. Section 3 shows results and analysis and Section 4 summarizes this paper.

2 Model architecture

2.1 Basic Idea

Our proposed model consists of a tweet processor and a deep learning module that are treated as two distinct components. The tweet processor standardizes tweets, applies semantic rules and then generates embeddings. The deep learning module is a combination of Bi-CGRNN and DeepCNN. To formulate our challenges in increasing the classification accuracy, we illustrate the basic idea of our model in Figure 1 as follows: Tweets are firstly considered by the tweet processor based on pre-processing steps of (Go et al., 2009) and the semantic rule-based approach from (Appel et al., 2016). Then, we construct four kinds of embeddings in the representation-level step: ContinuousW2Vs, CharAVs, LexW2Vs and DependencyW2Vs, where CharAVs are generated from a DeepCNN. The DeepCNN is constructed from two wide convolutions which can learn to recognize specific n-grams at every position in a word and allow features to be extracted independently of these positions in the word. These features maintain the order and relative positions of characters and are formed at a higher abstract level. In those embeddings, ContinuousW2Vs take the syntax and semantic of words (Mikolov et al., 2013) while the LexW2Vs can capture the sentiment of words (Shin et al., 2016). DependencyW2Vs derive the syntactic relations of words and exhibit more functional similarity than the original skip-gram embeddings led to form global syntactic contexts of words (Levy and Goldberg, 2014). Twitter sentiment label belongs to global sentence level while traditional word embeddings capture local contexts only. Therefore, DependencyW2Vs are useful in capturing global context of tweets. On the other hand, we create two advanced embeddings by integrating LexW2Vs and CharAVs with ContinuousW2Vs and DependencyW2Vs for Bi-CGRNN. A Bi-CGRNN is enhanced from a standard Bi-GRNN of (Chung et al., 2014) by incorporating contextual features (e.g., dependency-based contexts) into the model. The Bi-CGRNN produces a sentence-level representation from sentence compositions in order to maintain the order of word and capture syntax, semantic, sentiment and context of a sentence based on these embeddings.

2.2 Data Preparation

- *Stanford - Twitter Sentiment Corpus (STS Corpus):* STS Corpus contains 1,600K training tweets collected by a crawler from (Go et al., 2009). (Go et al., 2009) constructed a test set manually with 177 negative and 182 positive tweets. The Stanford test set is small. However, it has been widely used in different evaluation tasks (Go et al., 2009) (Dos Santos and Gatti, 2014).

- *Health Care Reform (HCR):* This dataset was constructed by crawling tweets containing the hashtag *#hcr* (Speriosu et al., 2011). The task of this paper is to predict positive/negative tweets.

2.3 Tweet Processor

We first take the unique properties of Twitter 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 in the dataset, we consider them as words in order that deep learning classifiers can capture information from emoticons. Because the test set contains emoticons, they do not influence classifiers if emoticons do not contain in its training data. Therefore, the emoticons would be useful

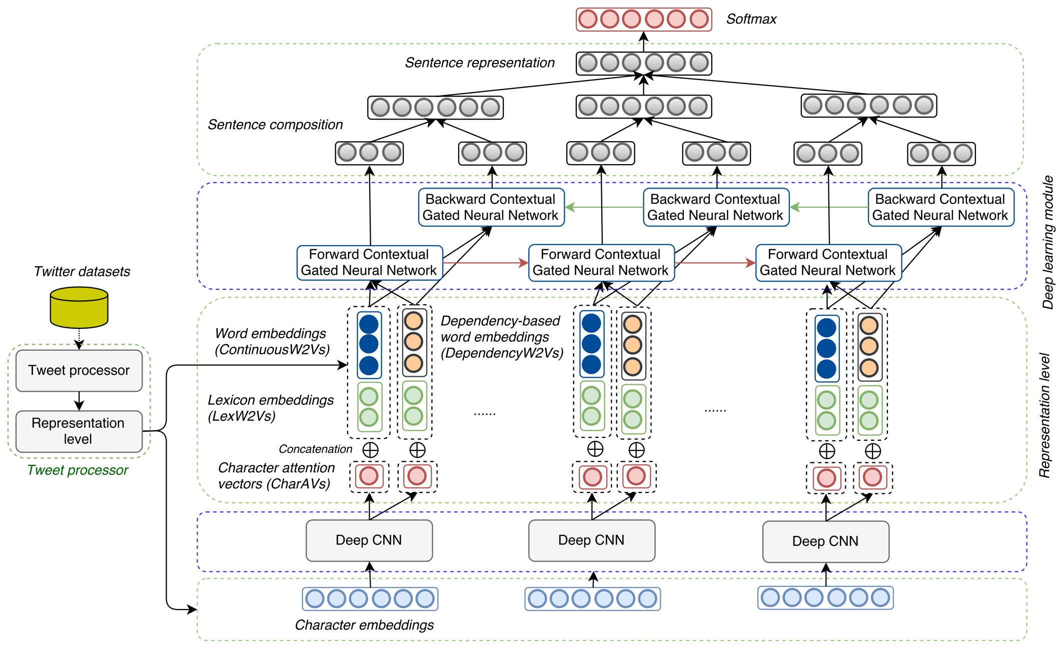

Softmax
Sentence representation
Sentence composition
Backward Contextual Gated Neural Network
Forward Contextual Gated Neural Network
Deep learning module
Twitter datasets
Tweet processor
Representation level
Tweet processor
Word embeddings (ContinuousW2Vs)
Dependency-based word embeddings (DependencyW2Vs)
Lexicon embeddings (LexW2Vs)
Concatenation
Character attention vectors (CharAVs)
Representation level
Deep CNN
Character embeddings

Figure 1: The overview of a deep learning system.

when classifying test data by using deep learning model. However, our preprocessing steps are different from (Go et al., 2009), they remove the emoticons out from their training datasets because they revealed that the training process makes the usage of emoticons as noisy labels and if they consider the emoticons, there is a negative impact on classification accuracy. In addition, traditional classifiers can not focus on non-emoticons (e.g., unigrams and bi-grams) to predict exactly the sentiment of tweets. After the pre-processing steps, we apply the semantic rules based on the idea of (Appel et al., 2016) and use a tweet processor from our previous work (Nguyen and Nguyen, 2017) to remove unnecessary sub-sentences from tweets in order that the deep learning model learns essential features from tweets. The semantic rule-based approach can handles negation and shows advances led to effectively affect the output of classifiers.

2.4 Representation Level

In this section, we describe how the kinds of embeddings are constructed by the representation module. 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 continuous word-level embedding r_{word}, we use a pre-trained word-level. We build every word w_i into two advanced word embeddings:

- Advanced continuous word embeddings $v_i = [r_i; e_i; l_i]$ is constructed by three sub-vectors: the pre-trained word-level embedding $r_i \in R^{d^{word}}$, the character attention vector $e_i \in R^l$ of w_i where l is the length of the filter of wide convolutions, the lexicon embedding $l_i \in R^{d^{score}}$ where d^{score} is list of sentiment scores for that word in lexicon datasets.

- Advanced dependency-based word embeddings $d_i = [de_i, e_i; l_i]$ is also built by three sub-vectors: the dependency-based word embedding $de_i \in R^{d^{word}}$, the character attention vector e_i and the lexicon embedding l_i. The advanced dependency-based word embeddings contain syntactic contexts and is increased information from LexW2Vs and CharAVs.

This deals with three main problems: (i) Sentences have any different size; (ii) Important information of characters that can appear at any position in a word are extracted; (iii) The syntax, semantic, sentiment, context and the morphology of words in a sentence are captured by concatenating two advanced embeddings via Bi-CGRNN in order to produce sentence representation.

We have N fixed-size CharAVs corresponding to word-level embeddings in a sentence. In subsection 2.5, 2.6 and 2.7, we illustrate the methods of constructing LexW2Vs, CharAVs using DeepCNN and DependencyW2Vs.

2.5 Lexicon Embeddings (LexW2Vs)

The LexW2Vs are constructed by taking scores from various lexicon datasets. In lexicon datasets, each word contains key-value pairs in which the key is a word and the value is a list of sentiment scores for that word. These scores range from -1 to 1, where -1 is most negative and 1 is most positive, respectively.

For each word $w_i \in V^{word}$, where V^{word} is a fixed-sized word vocabulary, a lexicon embedding is constructed by concatenating all of the scores among lexicon datasets with respect to w_i. If w_i does not exist in a certain dataset, 0 value is substituted. The lexicon embedding is a form of a vector $l_i \in R^{d^{score}}$, where d^{score} is the total number of scores across all lexicon datasets. We use seven lexicon datasets for building LexW2Vs:

- Bing Liu Opinion Lexicon (Hu and Liu, 2004).
- NRC Hashtag Sentiment Lexicon (Mohammad et al., 2013).
- Sentiment140 Lexicon (Mohammad et al., 2013).
- NRC Sentiment140 Lexicon (Kiritchenko et al., 2014).
- MaxDiff Twitter Sentiment Lexicon (Kiritchenko et al., 2014).
- National Research Council Canada (NRC) Hashtag Affirmative and Negated Context Sentiment Lexicon (Kiritchenko et al., 2014).
- Large-Scale Twitter-Specific Sentiment Lexicon (Tang et al., 2014).

The purpose of building LexW2Vs is to take the different kinds of words for capturing the different sentiments of words. Table 1 illustrates the type of words for each dataset. We share idea with (Shin et al., 2016). However, our LexW2Vs are distinguished their approaches in the aspect: We cover the colloquial expressions and colloquial emoticons in tweets by using Large-scale Twitter-Specific Sentiment Lexicon.

Lexicon dataset	**The type of words**
Bing Liu Opinion Lexicon	Sentiment adjective words
NRC Hashtag Sentiment Lexicon	Hashtag emotion words & Hashtag topic words
Sentiment140 Lexicon	Emoticons & Sentiment words
NRC Sentiment140 Lexicon	Affirmative context words & Sentiment140 Negated Context words
MaxDiff Twitter Sentiment Lexicon	Twitter sentiment words
Hashtag Affirmative and Negated Context Sentiment Lexicon	Hashtag affirmative words & Negated contextual words
Large-Scale Twitter-Specific Sentiment Lexicon	Colloquial words & Emoticons

Table 1: The types of words in lexicon dataset.

We call such lexicon-based features as lexicon embeddings because embeddings are a feature input of deep learning model and describe the properties of a word or a phrase. Each word in each lexicon datasets actually has many values that can be built by training a neural network. The deep learning model uses this input for calculating a computational graph (weight matrix) that describe relatedness among words (n-gram order).

2.6 Character Attention Vectors (CharAVs)

Figure 2 describes the way of forming a character attention vector. We use a DeepCNN with two wide convolutions. The first convolution produces a fixed-size character feature vector named *m-gram* features by extracting local features around each character window of the given word and using a max pooling over vertical character windows. The second convolution retrieves the fixed-size character feature and transforms the representation to yield a character attention vector by performing max pooling on each row of matrix instead of each column. The purpose of this method is to attend on the highest *n-gram* feature in order to transform these *m-gram* features at previous level into a representation at a more focused abstract level and produces an attention over only the best feature vector. Character attention vectors has two advantages: One is that this model could adaptively assign an importance score to each piece of word embedding according to its semantic relatedness with characters of each word. Another advantage is that this attention model is differentiable, so that it could be easily trained together with other components in an end-to-end fashion. In the next sub-section, we introduce the structure of CNN with wide convolution.

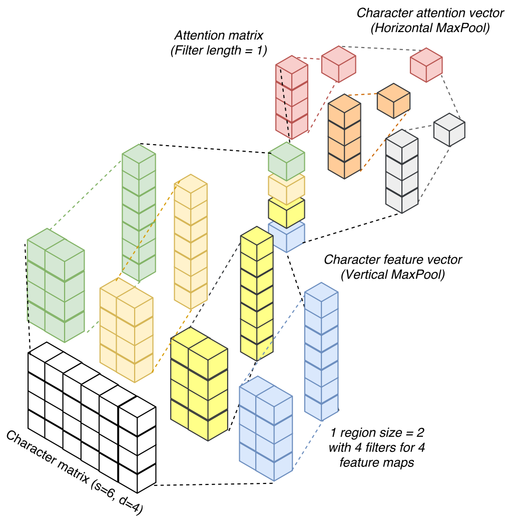

Character attention vector (Horizontal MaxPool)
Attention matrix (Filter length = 1)
Character feature vector (Vertical MaxPool)
1 region size = 2 with 4 filters for 4 feature maps
Character matrix (s=6, d=4)

Figure 2: 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. The CharAVs is then created by performing max pooling on each row of the attention matrix.

Convolutional Neural Network: The convolution 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 requires that $s \geq m$ and produce a sequence $c \in R^{s-m+1}$. The wide type 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.
Wide Convolution: Given a word w_i composed of M characters $\{c_1, c_2, ..., c_M\}$, we take a character embedding $u_i \in R^{d^{char}}$ for each character c_i and construct a character matrix $W^{char} \in R^{d^{char}} \times |V^{char}|$. 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 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 $d \times (s + m - 1)$.

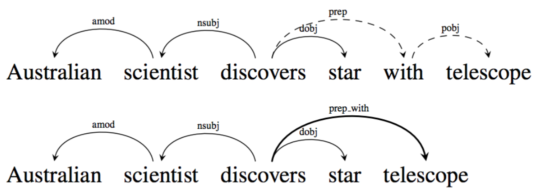
2.7 Dependency-based Word Vectors (DependencyW2Vs)

To construct context embeddings, we use the idea of (Levy and Goldberg, 2014) to derive syntactic contexts based on the syntactic relations of a word. Most previous works on neural word embeddings take the contexts of a word by computing *linear-context* words that precede and follow the target word. However, these contexts can be exploited similar by generalizing the *SKIP-GRAM* model. The model for learning Dependency-based Word Vectors is improved from *SKIP-GRAM* model in which the linear bag of words contexts are replaced with arbitrary word contexts from dependency tree. Syntactic contexts are derived from produced dependency parse-trees. Specifically, the bag-of-words in the *SKIP-GRAM* model yield *broad topical similarities*, while the dependency-based contexts yield more *functional similarities* of a *cohyponym nature*. In the *SKIP-GRAM* model, the contexts of a word are the words surrounding it in the text. However, there is a limitation of *SKIP-GRAM* word embeddings: Contexts need not correspond to all of the words and the number of context-types maybe larger than the number of word-types. Therefore, dependency-based contexts capture more information than bag-of-words contexts. In Figure 3, the contexts are extracted for each word in the sentence and the contexts of a word are derived from syntactic relations of a word in the sentence. For parsing syntactic dependencies, we use a parser from (Goldberg and Nivre, 2013) for Stanford dependencies and the corpus are tagged with parts-of-speech using Stanford parser [1].

After parsing each sentence, we consider word context as Figure 3: For a target word *w* with modifiers $m_1, m_2, ..., m_n$ and a head h, we form the contexts as $(m_1, lbl_1), ..., (m_n, lbl_n), (h, lbl_h^{-1})$, where *lbl* is the type of the dependency relation between the head and the modifier, lbl^{-1} is used to mark the inverse-relation. The advantages of syntactic dependencies are inclusive and more focused than bag-of-words. In addition, they can capture relations that out-of-reach with small windows and filter out contexts that are not directly related to the target word. For example, *Australian*

[1]https://nlp.stanford.edu/software/lex-parser.shtml

is not used as the context for *discovers*. Therefore, we have more focused embeddings that capture more functional and less topical similarity.



amod
nsubj
prep
dobj
pobj
Australian scientist discovers star with telescope
amod
nsubj
prep_with
dobj
Australian scientist discovers star telescope

WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Figure 3: Dependency-based context extraction example (Levy and Goldberg, 2014)

2.8 Contextual Gated Recurrent Neural Network (CGRNN)

Gated Recurrent Neural Network: The Bi-GRNN is a version of (Chung et al., 2014) in which a Gated Recurrent Unit (GRU) has two gates, a reset gate r_t and a update gate z_t. Intuitively, the reset gate determines how to combine the new input with the previous memory, and the update gate defines how much of the previous memory to keep around. We use GRUs for our model because GRUs are quite new and their tradeoffs have not been fully explored yet. On the other hand, GRUs have fewer parameters (U and W are smaller) and thus may train a bit faster or need less data to generalize. The equation 2 illustrates the construction of a GRU cell:

$$\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z) \\
\hat{h}_t &= g(x_t W_{xh} + (r_t \odot h_{t-1})W_{hh} + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t.
\end{aligned} \tag{2}$$

Contextual Gated Recurrent Neural Network: Based on the idea of GRNN model, we build a power of syntactic contexts into a standard Bi-GRNN model which adapt GRNN cell to take both words and syntactic contexts by modifying the equations representing operations of the GRNN cell. A GRNN cell is added dependency-based word vector T to reset gate, update gate and hidden state. In the Equation 3, the term in bold is the modification made to the original GRNN equation. Based on these equations, adding dependency-based word vector T is corresponding to considering a composite input $[x_i, T]$ to the GRNN cell that concatenates the advanced continuous word embeddings and advanced dependency-based word embeddings.

$$\begin{aligned}
r_t &= \sigma(W_{xr}x_t + W_{hr}h_{t-1} + \mathbf{W_{Ti}T} + b_r) \\
z_t &= \sigma(W_{xz}x_t + W_{hz}h_{t-1} + \mathbf{W_{Ti}T} + b_z) \\
\hat{h}_t &= g(x_t W_{xh} + (r_t \odot h_{t-1})W_{hh} + \mathbf{W_{Ti}T} + b_h) \\
h_t &= (1 - z_t) \odot h_{t-1} + z_t \odot \hat{h}_t.
\end{aligned} \tag{3}$$

This approach of concatenating syntactic contexts and word embeddings works better in practice and deal with the context challenge in sentiment analysis.

3 Results and Analysis

3.1 Experimental setups

For the Stanford Twitter Sentiment Corpus (STS Corpus), we use the number of samples as (Dos Santos and Gatti, 2014). The training data is selected 80K tweets for a training data and 16K tweets for the development set randomly from the training data of (Go et al., 2009). We conduct a binary prediction for STS Corpus.

In Health Care Reform Corpus (HCR Corpus), we also select 10% randomly for the development set in a training set and construct as (Da Silva et al., 2014) for comparison. We describe the summary of datasets in Table 2.

Data	Set	N	c	l_w	l_c	$\|V_w\|$	$\|V_c\|$
STS	Train	80K	2	33	110	67083	134
	Dev	16K		28	48		
	Test	359		21	16		
HCR	Train	621	2	25	70	3100	60
	Dev	636		26	16		
	Test	665		20	16		

Table 2: 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.

Hyperparameters: For all datasets, the filter window size (h) is 7 with 6 feature maps each for the first wide convolution layer, the second wide convolution layer has a filter window size of 1 with 14 feature maps each. Dropout rate (p) is 0.5, l_2 constraint, learning rate is 0.1 and momentum of 0.9.

Mini-batch size for STS Corpus is 100 and HCR corpus is 4. Training is done through stochastic gradient descent over shuffled mini-batches with Adadelta update rule (Zeiler, 2012).

Pre-trained Word Vectors: We use the publicly available *Word2Vec* [2] trained from 100 billion words from Google and *TwitterGlove* [3] 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.

3.2 Experimental results

Table 3 shows the results of our model for sentiment analysis against other models. The different kinds of models are constructed to evaluate the impacts of embeddings on classification accuracy. We build the Bi-CGRNN enhanced CharAVs and LexW2Vs. In addition, we evaluate separately the effectiveness of CharAVs and LexW2Vs on Twitter datasets by incorporating with standard Bi-GRNN.

We compare our model performance with the approaches of (Go et al., 2009) and (Dos Santos and Gatti, 2014). The model of (Go et al., 2009) displays the good results in the previous time and the model of (Dos Santos and Gatti, 2014) reported the state-of-the-art so far by using a charSCNN. Our model shows the result of *88.57* is the best prediction accuracy for STS Corpus.

For HCR Corpus, we compare the performance with the results of (Da Silva et al., 2014) that used an ensemble of multiple base classifiers (ENS) such as Naive Bayes (NB), Random Forest (RF), SVM and Logistic Regression (LR). The ENS model is combined with bag-of-words (BoW), feature hashing (FH) and lexicons (lex). Our model outperforms the model of (Da Silva et al., 2014) with result of *80* so far.

3.3 Analysis

The model with CharAVs and LexW2Vs built on top of $GoogleW2V$ vectors and DependencyW2Vs is effective in order to increase classification accuracy. These experiments show that CharAVs and LexW2Vs achieve good performances and contribute in enhancing information for words. However, the experiments indicate that

Model	**STS**	**HCR**
CharSCNN/Pre-trained (Dos Santos and Gatti, 2014)	**86.4**	-
CharSCNN/Random (Dos Santos and Gatti, 2014)	81.9	-
SCNN/Pre-trained (Dos Santos and Gatti, 2014)	85.2	-
SCNN/Random (Dos Santos and Gatti, 2014)	82.2	-
MaxEnt (Go et al., 2009)	83.0	-
NB (Go et al., 2009)	82.7	-
SVM (Go et al., 2009)	82.2	-
SVM-BoW	-	73.99
SVM-BoW + lex	-	75.94
RF-BoW	-	70.83
RF-BoW + lex	-	72.93
LR-BoW	-	73.83
LR-BoW + lex	-	74.73
MNB-BoW	-	72.48
MNB-BoW + lex	-	75.33
ENS (RF + MNB + LR) - BoW	-	75.19
ENS (SVM + RF + MNB + LR) - BoW + lex	-	**76.99**
Bi-CGRNN + CharAVs + LexW2Vs + GoogleW2Vs	**88.5**	78.47
Bi-CGRNN + CharAVs + LexW2Vs + GloveW2Vs	86.9	**80.0**
Bi-CGRNN + GoogleW2Vs	85.7	77.74
Bi-GRNN + CharAVs + GoogleW2Vs	86.0	79.09
Bi-GRNN + LexW2Vs + GoogleW2Vs	88.0	78.79

Table 3: Accuracy of different models for binary classification.

CharAVs are more effective in small dataset than LexW2Vs. In addition, the Bi-CGRNN enhanced CharAVs and LexW2Vs have a great impact on Twitter datasets. Table 4 displays the effectiveness of our model in predicting the tweets. In the table 4, the Bi-GRNN using LexW2Vs captures the positive words (green words) and negative words (red words) for computing scores and predicts wrong labels because of the contexts of tweets while the Bi-CGRNN enhanced CharAVs and LexW2Vs recognizes contexts for true prediction. For example, the model using LexW2Vs predicts the last tweet to be positive because 'strong' word has sentiment stronger than 'no' word, however, the context of this tweet is negative. The pre-train word vectors are good, universal feature extractors. The syntactic contexts support in dealing context problems in tweets and the lexicons support word embeddings in dealing the sentiment of tweets. The difference between our model and other approaches is the ability of our model to capture enough features and combine these features at high level. In addition, the usage of DeepCNN for characters can learns a structure of words in higher abstract level. LexW2Vs and syntactic contexts

[2]code.google.com/p/word2vec/

[3]https://nlp.stanford.edu/projects/glove/

Model	Input from HCR Corpus	Gold Label	Prediction
Bi-CGRNN + CharAVs + LexW2Vs	@seanbaran74 well that's what's next. after #hcr they'll save the environment, give us CFLs and take away our TVs.	Negative	True
Bi-GRNN + LexW2Vs			False
Bi-CGRNN + CharAVs + LexW2Vs	All of us fighting for #HCR ask ourselves who #imherefor. Who are you fighting for? http://bit.ly/9-st	Positive	True
Bi-GRNN + LexW2Vs			False
Bi-CGRNN + CharAVs + LexW2Vs	Stephen Lynch strong 'no' on health bill despite talk with President obama http://bit.ly/cQIujP #hcr #tcot #tlot	Negative	True
Bi-GRNN + LexW2Vs			False

Table 4: The label prediction between the Bi-GRNN model using LexW2Vs and the Bi-CGRNN model using CharAVs and LexW2Vs (The red words are negative and the green words are positive).

contribute in supporting information for word embeddings. This helps the model not only learns to recognize single n-grams of a word, negation, but also patterns in n-grams led to form a structure significance of a sentence.

4 Conclusions

In the present work, we build a model that solves four challenges in Twitter: syntax, semantic, sentiment and context. Our results show the well-establish evidence that CharAVs, LexW2Vs and DependencyW2Vs are important ingredients for ContinuousW2Vs in increasing classification accuracy for sentiment analysis. Our source and processed data are available at Github[4].

Acknowledgments

This work was supported by JSPS KAKENHI Grant number JP15K16048 and CREST, JST.

References

Orestes Appel, Francisco Chiclana, Jenny Carter, and Hamido Fujita. 2016. A hybrid approach to the sentiment analysis problem at the sentence level. *Knowledge-Based Systems*, 108:110–124.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555.

Nadia F. F. Da Silva, Eduardo R. Hruschka, and Estevam R. Hruschka. 2014. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems*, 66:170–179.

C'icero N. Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *COLING*, pages 69–78.

Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford*, 1(12).

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the association for Computational Linguistics*, 1:403–414.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *CoRR*, abs/1408.5882.

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

Omer Levy and Yoav Goldberg. 2014. Dependency-based word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 2: Short Papers*, pages 302–308.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

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, Alan Ritter, Sara Rosenthal, Fabrizio Sebastiani, and Veselin Stoyanov. 2016. SemEval-2016 task 4: Sentiment analysis in twitter. *Proceedings of SemEval*, pages 1–18.

Huy Nguyen and Minh-Le Nguyen. 2017. A deep neural architecture for sentence-level sentiment classification in twitter social networking. *PACLING 2017*.

[4]https://github.com/titanbt/contextualGRU-attention-lexicon

Bonggun Shin, Timothy Lee, and Jinho D. Choi. 2016. Lexicon integrated CNN models with attention for sentiment analysis. *CoRR*, abs/1610.06272.

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 conference on empirical methods in natural language processing*, pages 151–161. Association for Computational Linguistics.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 53–63. Association for Computational Linguistics.

Duyu Tang, Furu Wei, Bing Qin, Ming Zhou, and Ting Liu. 2014. Building large-scale twitter-specific sentiment lexicon : A representation learning approach. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 172–182, Dublin, Ireland.

Matthew D. Zeiler. 2012. ADADELTA: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*.