Word Representation Learning Based on Bidirectional GRUs With Drop Loss for Sentiment Classification

Xia Sun, Yi Gao, Richard Sutcliffe, Shou-Xi Guo, Xin Wang, and Jun Feng, *Member, IEEE*

***Abstract*—Sentiment classification is a fundamental task in many natural language processing applications. Neural networks have achieved great successes on the sentiment classification task in recent years, since recurrent neural networks and long-short-term memory networks have the ability to deal with sequences of different lengths and to capture contextual semantic information. However, the effectiveness of these methods is limited when used to extract contextual information from relatively long texts. Therefore, in our model, we apply bidirectional gated recurrent units to capture contextual information as far as possible when learning word representations, which may effectively reduce the noise compared to other methods. We also propose a novel loss function namely drop loss (DL) which makes the model focus on the hard examples—examples which are easily classified incorrectly—in order to improve the accuracy of the model. We experiment on four commonly used datasets, and the results show that the proposed method has a good performance on four datasets, and needs fewer parameters compared with recent benchmarks, such as CoVe, ULMFiT, embeddings from language models, and bidirectional encoder representations from transformers. Furthermore, we demonstrate that the classification performance of existing shallow network models can be significantly improved by using DL. In particular, the accuracy of the CNN+LSTM model improves 9% on the IMDB-10 dataset.**

***Index Terms*—Loss function, neural networks, sentiment classification, word presentation.**

Manuscript received March 17, 2019; revised June 6, 2019; accepted September 1, 2019. Date of publication October 8, 2019; date of current version June 16, 2021. This work was supported in part by the National Natural Science Foundation Projects of China under Grant 61877050, in part by the Open Project Fund of Shaanxi Province Key Laboratory of Satellite and Terrestrial Network Tech of China, and in part by the Fund Program for the Scientific Activities of Selected Returned Overseas Professionals in Shaanxi Province of China under Grant 202160002. This article was recommended by Associate Editor X. Wang. *(Corresponding authors: Richard Sutcliffe; Jun Feng.)*

X. Sun and J. Feng are with the Department of Computer Science, Northwest University, Xi'an 710127, China, and also with the State-Province Joint Engineering and Research Center of Advanced Networking and Intelligent Information Services, Northwest University, Xi'an 710127, China (e-mail: raindy@nwu.edu.cn; fengjun@nwu.edu.cn).

Y. Gao, R. Sutcliffe, S.-X. Guo, and X. Wang are with the Department of Computer Science, Northwest University, Xi'an 710127, China (e-mail: gaoyi-1@foxmail.com; rsutcl@nwu.edu.cn; jialiangjia9@gmail.com; xinchn@163.com).

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TSMC.2019.2940097.

Digital Object Identifier 10.1109/TSMC.2019.2940097

I. Introduction

SENTIMENT classification is one of the most popularly used natural language processing (NLP) techniques and has been applied to many areas, such as E-commerce websites, stock forecast [1], and political orientation analyses [2]–[4]. In the sentiment classification task, feature-based representations play an important role, often based on the bag-of-words (BoWs) model [5], where bi-grams or larger n-grams are designed to represent features. For example, a BoW model is used to represent documents by Pang *et al.* [6] and Wang and Manning [7], who both build SVM classifiers for text classification. Although SVM is an extremely strong performer, the problem of data sparsity when using the BoW features heavily affects the classification accuracy [8]. Word embedding [9] has brought a new inspiration for solving the data sparsity problem to many NLP tasks [10], because it can represent each word as a low-dimensional, continuous, and real-valued vector [11]. For neural models [12]–[15] used to achieve tasks in NLP, pretrained word embeddings [16]–[18] play a significant role and are widely used in the state-of-the-art models as their first layer [19]. Hence, many methods based on word embedding have been proposed to capture more semantic information from text and improve the classification accuracy [16], [20]. For instance, Rao *et al.* [21], Tang *et al.* [11], and Xu *et al.* [22] utilized word embeddings to present words before they use neural networks to learn word representations.

Although the aforementioned models with word embeddings can greatly alleviate the data sparsity problem to improve the sentiment classification accuracy, word embeddings alone can still not eliminate the problem of word polysemy entirely, and this will impact the models performance to a certain extent. Polysemy can be characterized as follows: the same word may have different meanings in different contexts. For example, the word *present* shows diverse meanings in the following sentences: in "*He sent me a present for my birthday*," the word *present* means gift; in contrast, the meaning of word *present* refers to attendant in "*There were 200 people present at the meeting*."

Recurrent convolutional neural network (textRCNN) [23] proposes that combining a word and its context to present a word can obtain a more precise word meaning to reduce noise caused by polysemy. However, the textRCNN model merely considers that the context has impacts on words meaning and does not take into account the effect of the word itself to the

meaning of context when it captures the contextual semantic information of a word. We can use the earlier example to illustrate how textRCNN model will ignore the influence of word itself to contextual semantics. For word "*present*," textRCNN model regards "*He sent me a*" and "*for my birthday*" as the left-side and right-side context of word "*present*," respectively, and does not include word itself when it uses unidirectional recurrent neural network (RecurrentNN) [24] to capture the contextual semantics.

In fact, the target word can sometimes influence the meaning of a context. In other words, if you put different words into the same context, the meaning of the whole sentence may be different. For example, consider the sentence "*I am a _, and I am in class.*" where "_" is an undetermined word. If the undetermined word is "*student*," then the sentence would be interpreted as meaning that the student listens to the teacher in class. In contrast, if the unknown word is "*teacher*" then the sentence meaning is that the teacher teaches students in class. From the above example, we see that the word itself also has an effect on the semantics of the context. To address this limitation of the above model, we add the word itself into its left-side and right-side contexts during word representation learning process. In addition, the bidirectional mechanism captures more contextual information than a unidirectional one has proved in many sequence processing tasks [25], [26]. So we use a bidirectional network structure and gated recurrent unit (GRU) [27], respectively, to replace the unidirectional structure and RecurrentNN of the textRCNN to capture deeper contextual semantics of each word. And then combine these contextual semantics of words with itself to make words meaning more precise to obtain better sentiment classification accuracy.

In contrast to our approach as outlined above, many researchers tend to use deep neural networks to capture the deep semantic information of documents [28]–[30] or research pretrained word representations add into various models [19], [31] to achieve the purpose of improving the models classification performance at present. Le *et al.* [32] mentioned that the importance of depth in convolutional models via analyzing the respective advantages of shallow-and-wide convolutional neural network (CNN) [33] and deep CNN. Peters *et al.* [20] used a deep bidirectional language model (biLM) to learn deep contextualized word representations. Although the aforementioned approaches have pushed forward this field significantly, we still consider whether there is a way to improve the models classification performance without increasing the depth of the neural network.

We have found that examples which are not easy to classify correctly can heavily affect the accuracy of a model during the process of experiment. For example, in "*She runs the gamut of emotions all the way from A to B*," the expression is implicit without specific emotion words, but it actually expresses positive sentiment. Hence, examples like these are difficult to classify correctly with shallow neural networks and are called hard examples. In contrast, there are easy examples that are easy to be classified correctly, such as "*Perhaps the grossest movie ever made*" where the word "*grossest*" expresses the sentiment strongly and clearly. For these, shallow neural networks work well because the expression of sentiment is explicit and uses specific emotional words.

Sentiment classification datasets consist of both easy and hard examples. As long as the classification accurancy of hard examples is improved, the classification accuracy of shallow neural network models will also increase. If a large number of easy examples exist in the data, they will play a major role in the classification model to lead the direction of the gradient when neural networks are used for sentiment classification. Under these circumstances, the neural network may be unable to learn useful information and cause hard examples are misclassified. Drawing on the idea of Lin *et al.* [34], we design a loss function suited to the sentiment classification task called drop loss (DL). Different from the idea of the Lin *et al.* work, we add a truncation factor into our loss function. DL makes the model enhance the learning of hard examples by down-weighting the loss assigned to the easy examples and preventing a vast number of easy examples from overwhelming the model during the process of training. Therefore, DL can improve the classification accuracy without changing the models depth. In addition, the time complexity of the model will be reduced with using DL because it can directly filter the loss of some easy examples via truncation factor during training.

We conducted experiments on four datasets to verify the effect of our model. The experiments demonstrate that our model has a good performance on all four datasets. DL has significantly improved the classification accuracy of the model, especially in some shallow neural networks where the number of network layers is small, such as single-layered CNN, long-short-term memory (LSTM) [35], or GRU, etc. Overall, our main contributions are as follows.

1) We introduce two bidirectional GRUs to capture the left-side and right-side contexts information of each word, respectively. We add the word itself into its left-side and right-side contexts when we learn the word representation. Therefore, our method is able to capture more meaningful semantics compared to textRCNN which is regarded as one of the best existing methods.
2) For the sentiment classification task, a novel loss function DL is proposed to improve the models attention on hard examples. DL will not affect the loss of hard examples during the reduction of loss values for easy examples. Furthermore, DL can directly filter the loss of some easy classifications without increasing the amount of calculation.
3) Our model is comparable to the state-of-the-art methods for sentiment classification, such as CoVe, ULMFiT, embeddings from language model (ELMo), and bidirectional encoder representations from transformer (BERT), and is better than CoVe and ELMo on IMDB-2 and SST-5 datasets. Moreover, our model is more efficient to train than all these methods and uses substantially less parameters.
4) Experiments we have conducted to prove that when DL is incorporated into well-known shallow neural network models in place of the traditional cross-entropy (CE) loss function, the accuracy of these models is improved.

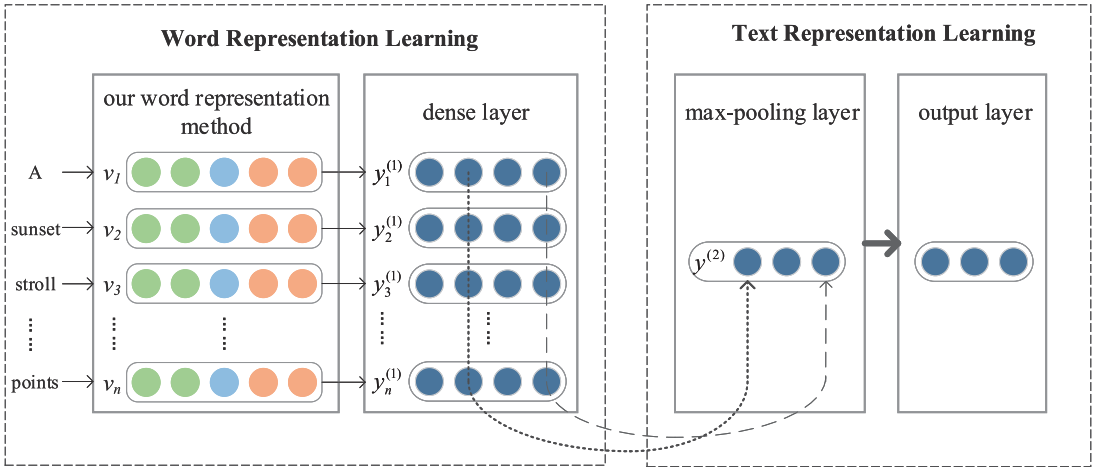

Fig. 1. Overview of our model architecture for sentiment classification. This figure regards the sentence "*A sunset stroll along the South Bank affords an array of stunning vantage points.text*" as example.

II. Related Work

Sentiment classification is a fundamental task in NLP. Word representation is regarded as a crucial intermediate component of sentiment classification [23]. BoW [5] is one of the widely used word representation methods. Pang *et al.* [6] used a supervised machine learning method SVM and represent word with BoW feature for text classification. Afterward, various approaches focus on designing the handcrafted features to improve the performance of machine learning methods for text classification [36], [37]. Representative features include n-grams [7] and text topic [38]. However, these methods have the data sparsity problem [11], [23], and their classification accuracy greatly depends on the effectiveness of the handcrafted features [21].

Recently, deep neural networks [12], [13] and word embedding [9] have offered some novel perspectives to alleviate the aforementioned problems [10]. Therefore, many neural models have been proposed based on word embedding to improve sentiment classification accuracy [16], [21], [22]. Pretrained word embeddings, such as Word2Vec [39] and GloVe [17], gradually become an important component in many neural language understanding models [20] and common initializations for the word vector of deep learning models [11], [40].

Subsequently, some researchers have found that combining contextual information with word embeddings as the input of neural networks for NLP task can improve the performance of these models [40]. For instance, Lai *et al.* [23] proposed that adding the context into word embedding to present a word vector. McCann *et al.* [40] used a deep LSTM encoder from an attentional model trained for machine translation to contextualize word vectors, and combine these context vectors with its corresponding vector based on GloVe as an input of model. Unlike previous methods for learning contextualized word vectors [40], [41], ELMo representations learned functions of the internal states of a deep biLM so that it contains deeper semantics of text [20]. Moreover, another kind of pretraining models need not to include the pretrained representations as additional features [31]. Howard and Ruder [19] proposed a model namely ULMFiT which consists of three stages, and they consider that pretraining is beneficial for tasks. BERTs [31], proposed by Devlin *et al.*, is based on a multilayer bidirectional transformer [42] and shows the state-of-the-art performance on various NLP task.

III. Proposed Method

In this section, we introduce our word representation method and loss function. Fig. 1 is the overview of our model for sentiment classification. Given a document D as the input of the network, which is a sequence of words $w_1, w_2, \ldots, w_n$. The words are then presented by our word representation method, which will be specifically illustrated in Section III-A of this section. The output of the word representation method is regarded as the input of the dense layer, where each part of word vector (the word vector contains a word embedding for the word and the words left and right contexts.) obtained from word representation method will be completely fused. After that all represented words of a text will be sent to the next layer. Next, the max-pooling layer is used to capture the information throughout the entire text. Finally, the output layer predicts the class of D by our method, and our loss function is used in the process of training the model.

A. Word Representation Learning

Polysemy is a common phenomenon in language, so we intend to combine a word and its contextual information to present a word to alleviate the impact of polysemy for model performance. In our model, we use two bidirectional GRU models to capture contextual information of each word and combine with word embedding, which can help us to get a more accurate word meaning.

The bidirectional GRU model is used to learn the semantic information of the left and right contexts of words. The output of sending the words left and right contexts to the bidirectional GRU, respectively, is called left context vector and right context vector. As shown in Fig. 2, for a word w_i, we concatenate its word embedding with its corresponding left and right context vectors. We define $c_l(w_i)$ as the left context vector of word

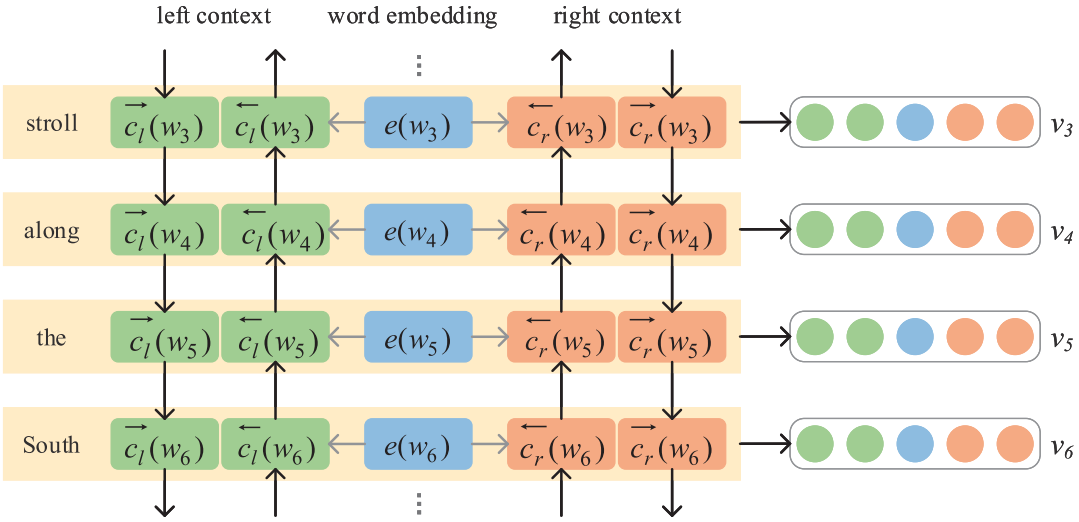

Fig. 2. Framework used for word representation. The figure is a partial example of the sentence "*A sunset stroll along the South Bank affords an array of stunning vantage points*", and subscripts denotes the positions of the corresponding word in the original sentence.

w_i. Similarly, $c_r(w_i)$ denotes the right context vector of word w_i. $c_l(w_i)$ is calculated by (1)–(3), where $e(w_i)$ represents the word embedding of word w_i, $\overrightarrow{c_l}(w_{i-1})$ and $\overleftarrow{c_l}(w_{i-1})$ refer to the results of contexts of w_{i-1} (w_{i-1} is the previous word of w_i) pass the forward and backward GRUs, respectively. For the parameters W_l and W_l^s, "$\rightarrow$"and "$\leftarrow$" indicate the direction of the bidirectional network. W_l and W_l^s are both weight matrices, W_l is used to transform the layer (context) into the next layer and W_l^s is used to joining $e(w_i)$ into the previous word's left context. f is a nonlinear activation function. The calculation method of $c_r(w_i)$ is similar to $c_l(w_i)$, and the specific formula is shown in (4)–(6)

$$\overrightarrow{c_l}(w_i) = f\left(\overrightarrow{W_l}\overrightarrow{c_l}(w_{i-1}) + \overrightarrow{W_l^s}e(w_i)\right) \tag{1}$$

$$\overleftarrow{c_l}(w_i) = f\left(\overleftarrow{W_l}\overleftarrow{c_l}(w_{i+1}) + \overleftarrow{W_l^s}e(w_i)\right) \tag{2}$$

$$c_l(w_i) = [\overrightarrow{c_l}(w_i);\ \overleftarrow{c_l}(w_i)] \tag{3}$$

$$\overleftarrow{c_r}(w_i) = f\left(\overleftarrow{W_r}\overleftarrow{c_r}(w_{i+1}) + \overleftarrow{W_r^s}e(w_i)\right) \tag{4}$$

$$\overrightarrow{c_r}(w_i) = f\left(\overrightarrow{W_r}\overrightarrow{c_r}(w_{i-1}) + \overrightarrow{W_r^s}e(w_i)\right) \tag{5}$$

$$c_r(w_i) = [\overleftarrow{c_r}(w_i);\ \overrightarrow{c_r}(w_i)]. \tag{6}$$

Equations (3) and (6) are used to concatenate the semantics of context captured from forward and backward direction to present left and right context vector, respectively. We will illustrate our word representation learning with a concrete example. For the word representation process of "*along*" in Fig. 2, $\overrightarrow{c_l}(w_4)$ encodes the semantics of the context "*... stroll along*" together with all previous words in the sentence, $\overleftarrow{c_l}(w_4)$ encodes the semantics of context "*along the South ...*" We connect $\overrightarrow{c_l}(w_4)$ and $\overleftarrow{c_l}(w_4)$ to get the left context vector $c_l(w_4)$. For the right context vector $c_r(w_4)$, it is combined in the order of $\overleftarrow{c_r}(w_4)$ and $\overrightarrow{c_r}(w_4)$. Then, the representation of word w_i is defined as (7), which is made up of $c_l(w_i)$, $e(w_i)$, and $c_r(w_i)$. Our model may be better able to disambiguate the meaning of w_i through combining this contextual information

$$v_i = [c_r(w_i);\ e(w_i);\ c_r(w_i)]. \tag{7}$$

We apply a linear transformation together with the relu [43] activation function to v_i after obtaining the representation v_i of the word w_i, and then send the result to the next layer. Equation (8) shows the dense layer of Fig. 1. $y_i^{(1)}$ is a latent semantic vector, in which each component of v_i will be fused to obtain better word representation effect

$$y_i^{(1)} = \text{relu}(W_1 v_i + b_1). \tag{8}$$

B. Text Representation Learning

All of the representations v_i for the words w_i are calculated, we apply a max-pooling layer after the dense layer (shown in Fig. 1). The max function is an element-wise function, in which the kth element of $y^{(2)}$ is the maximum in the kth elements of $y_i^{(1)}$

$$y^{(2)} = \max_{1 \le i \le n} y_i^{(1)}. \tag{9}$$

The pooling layer not only reduces the text vectors dimensions in order to simplify the computational complexity of the network; it also captures the information throughout the entire text. There are other types of pooling layers such as average pooling layers [44]. Because only a few words and their combination contribute to the sentiment of a document, we do not use average pooling here. The max-pooling layer will capture the k meaningful elements of the text, so we select it as our pooling layer to determine the most meaningful factor for classification.

The last layer of our model is the output layer. Similar to traditional neural networks, it is defined as

$$y^{(3)} = W_3 y^{(2)} + b_3. \tag{10}$$

Finally, the softmax function is used to output the probability of classifying, where C is the number of sentiment categories

$$P_i = \frac{\exp\left(y_i^{(3)}\right)}{\sum_{k=1}^{C} \exp\left(y_i^{(3)}\right)}. \tag{11}$$

The parameters of our model need to be updated during training are denoted as $\triangle$:

$\triangle = \{E, \overrightarrow{W_l}, \overrightarrow{W_l^s}, \overleftarrow{W_l}, \overleftarrow{W_l^s}, \overrightarrow{W_r}, \overrightarrow{W_r^s}, \overleftarrow{W_r}, \overleftarrow{W_r^s}, W_1, W_3, b_1, b_3\}$.

Specifically, $E \in \mathbb{R}^{|e|\times V}$ refers to word embedding, where $|e|$ is the dimention of each word and V presents vocabulary size. $\overrightarrow{W_l}, \overrightarrow{W_l^s}, \overrightarrow{W_r}, \overleftarrow{W_r} \in \mathbb{R}^{|c|\times|c|}$, where $|c|$ is number of GRU units. $\overrightarrow{W_l^s}, \overleftarrow{W_l^s}, \overrightarrow{W_r^s}, \overleftarrow{W_r^s} \in \mathbb{R}^{|e|\times|c|}$, $W_1 \in \mathbb{R}^{H\times(|e|+4|c|)}$, $W_3 \in \mathbb{R}^{C\times H}$, the bias vectors $b_1 \in \mathbb{R}^H$ and $b_3 \in \mathbb{R}^C$, where H refers to the hidden layer size.

C. Drop Loss

DL is designed to solve the problem of easy/hard examples in sentiment classification tasks. We introduce DL starting from the CE loss function

$$\mathrm{CE} = -\sum_{j=1}^{C} y_j \log\left(\widehat{y}_j\right) \tag{12}$$

where y is the truth class of an example and it is encoded as a one-hot vector. $\widehat{y}$ is the predicted result by softmax function. C is the number of sentiment categories. For notational convenience, we define p and rewrite the CE formula as follows:

$$p = \max_{1\le i\le C} y_j \widehat{y}_j \tag{13}$$

$$\mathrm{CE}(p) = -\log(p) \tag{14}$$

where $p \in [0, 1]$ is the models estimated probability for the truth class. One notable property of this loss is that it does not discriminate between the loss of easy examples and hard ones. In other words, even examples that are easily classified ($p \gg 0.5$) lead to a loss with nontrivial magnitude. When summing a large number of easy examples, their loss values can overwhelm those of hard examples.

As our experiments show, the vast number of easy examples account for most of the loss and dominate the gradient. Therefore, we down-weight the loss of easy examples through reshaping the loss function. Then, the hard examples are focused on during the training process. More specifically, we refer to the modulating factor $(1-p)^\gamma$ of focal loss (FL) [34] to solve the above problems. We transform the CE loss into (11), where the factor $\gamma \in [0, 5]$ is for adjusting the rate at which easy examples are down-weighted

$$\mathrm{CE}^{(1)} = -(1-p)^\gamma \log(p). \tag{15}$$

Intuitively, the modulating factor reduces the contribution of easy examples to the loss. For instance, when $\gamma = 2$, an example classified with $p = 0.9$ would have 100 times lower loss compared with CE. Certainly, if the p of an example is higher, it should have the lower loss. In brief, the modulating factor has two advantages as follows: 1) the factor down-weights the loss of easy examples and 2) the parameter γ steadily adjusts the rate at which easy examples are down-weighted.

While $(1-p)^\gamma$ causes the loss for well-classified examples to down-weight, the loss of hard examples goes down as well. Although the rate of loss for easy examples becomes faster as the parameter γ increases, the loss of some hard examples also decreases. In order to avoid the problem of hard examples loss decrease when we down-weight the loss for easy examples, we propose to add a threshold m (truncation factor) into $\mathrm{CE}^{(1)}$. If $p > m$ ($m \in [0, 1]$), DL will directly ignore the loss for the example. By this means, we accelerate the declining rate of loss for easy examples without increasing γ. We define the DL as

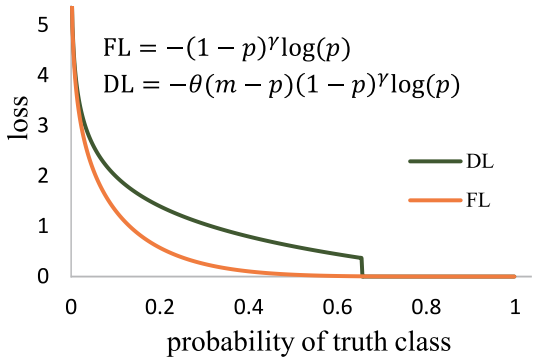

Fig. 3. Loss function (DL function and FL function) curve. We set $\gamma = 5$ in FL, and set $\gamma = 5, m = 0.65$ in DL. For FL, all examples are treated by it that means all sample losses decrease with increasing γ, and the loss for some hard examples has approached 0.

$$\mathrm{DL} = -\theta(m-p)(1-p)^\gamma \log(p) \tag{16}$$

where θ is described as (17), which helps us to achieve the function of ignoring the loss for the example with $p > m$

$$\theta(x) = \begin{cases} 1, & x > 0 \\ 0.5, & x = 0 \\ 0, & x < 0. \end{cases} \tag{17}$$

Since $(1-p)^\gamma$ has reduced the loss of easy examples, why do we need the truncation factor m? Consider the instance for $\mathrm{CE}^{(1)}$ where $\gamma = 5$; an easy example classified with $p = 0.65$ will have 100 times lower loss, but for a difficult-to-classify example and we assume its probability $p = 0.46$ which will be classified correctly, its loss will also dropped by 100 times. In order to explain the introducing truncation factor m more clearly, we use Fig. 3 to illustrate the difference between FL and DL. For FL, setting $\gamma > 0$ reduces the relative loss for easy examples ($p > 0.5$), but some hard examples' loss are also reduced. Moreover, some hard examples' loss values will be infinitely approaching 0 as shown in Fig. 3, and this will have an impact on the performance of the model. For DL, however, if we set $\gamma = 1$ and $m = 0.65$ for DL, the loss is filtered for the easy examples classified with $p > 0.65$, while the loss of hard examples is unaffected.

According to the above description, we note two properties of the DL.

1) When an example is misclassified and $p < m$, the loss is unaffected and the loss for some easy examples ($p > m$) will be directly removed.
2) We reduce the network time complexity because some example losses are filtered directly. Generally speaking, it down-weights the loss assigned to well-classified examples.

IV. Experiments

In this section, we will introduce the datasets and experimental parameters used in our experiments. Besides, we will analyze the results of our model and other baselines on these

TABLE I
STATISTICAL INFORMATION OF FOUR DATASETS

Datasets	Train size	Valid size	Test size	Class
MOOC	7,902	987	989	2
IMDB-2	20,000	2,500	2,500	2
IMDB-10	40,000	5,000	5,000	10
SST-5	75,360	9,420	9,420	5

MOOC, IMDB-2, IMDB-10 and SST-5 are datasets for experiments. Train size, Valid size and Test size represent the number of train/valid/test set entries. Class is the number of classes.

TABLE II
OPTIMAL HYPER-PARAMETER CONFIGURATION FOR FOUR DATASETS

Datasets	$\|c\|$	γ	m
MOOC	50	1	0.9
IMDB-2	50	2	0.9
IMDB-10	50	1	0.9
SST-5	50	1.5	0.8

common datasets to verify model performance. Our experiment expands along the following three aspects: 1) verifying our word representation method effect; 2) verifying the effect of DL; and 3) combine our word representation method with DL for experimentation.

A. Datasets

For sentiment classification, we evaluate our model on the following datasets: the Stanford massive open online courses (MOOCs) posts datasets (http://datastage.stanford.edu/StanfordMoocPosts), IMDB (http://datasets.imdbws.com), and the Sentiment Treebank (http://nlp.stanford.edu/sentiment). Table I provides detailed information about each dataset.

Stanford MOOC Posts: This dataset contains forum posts pertaining to three domain areas: 1) humanities, 2) medicine, and 3) education. We use the area of Education and set the posts with a sentiment rating greater than four in the MOOC post dataset to fall into the sentiment class as positive, while all other posts fall into the negative class.

IMDB-2 and IMDB-10: We use the binary version of IMDB as well as its ten-class version. IMDB-10 is a large movie review dataset that consists of ten classes [45] and contains 50 000 reviews. IMDB-2 contains 25 000 reviews, and its class is binary. They include movie reviews from around the world.

SST-5: This dataset contains movie reviews parsed and labeled by Socher *et al.* [46]. The labels are very negative, negative, neutral, positive, and very positive.

B. Experiment Settings

Our preprocessing is as follows. For all documents, we use the natural language toolkit (nltk) to obtain the tokens. For the four datasets, we divide the datasets into training, validation, and testing sets with proportion 8/1/1. We train word embeddings using the default parameters in word2vec with the Skip-gram algorithm. We use accuracy as the metric (in %) to evaluate the performance of our approach. Our network is trained on one NVIDIA GeForce GT730 GPU in a 64-bit Dell computer with one 3.60-GHz CPU and 8 GB main memory.

For the choice of hyper-parameters, we choose a set of commonly used values, following previous studies [23], [40], [44], [47]. Furthermore, we use RMSprop [48] as an optimizer and its learning rate is set to 0.001. The vector size of the word embedding is $|e| = 300$, the hidden layer size $H = 100$, and the number of vocabulary $V = 300$. Formally, some parameters of our method will be slightly adjusted according to different datasets. These parameters set for each dataset are shown in Table II.

C. Baselines

To evaluate the effectiveness of our approach, we compared our methods with several existing algorithms.

1) *Bag of Words/Bigrams + LR/SVM:* Wang and Manning [7] build an SVM classifier after representing a document with unigram features. These baselines mainly use machine learning algorithms with unigram or bigrams as features. We train logistic regression (LR) and SVM, respectively, with unigrams and bigrams as features.
2) *LSTM:* LSTM is an RecurrentNN with memory cells and a three-gate mechanism [35], [49]. It can capture further contextual information than RecurrentNN during training. LSTM prevent the gradient vanishing of RecurrentNN.
3) *CNN:* The CNN has strong adaptability and is adept at extracting local features [50].
4) *GRU:* GRU is proposed by Bahdanau *et al.* whose network structure adopts a two-gate mechanism and is leaner than LSTM [27].
5) *2-Layer LSTM:* Both of its two hidden layers are LSTMs, the first hidden layer is used as input to the second layer in the same time step [13]. Here, the idea is to let the second layer capture longer-term dependencies of the input sequence.
6) *CNN+LSTM:* This model is used for sentiment classification and consists of a convolutional layer, a max pooling layer and an LSTM layer [51].
7) *textRCNN:* Lai *et al.* [23] used a recurrent structure to capture the semantics of the contexts and combine it with a word to present a word.
8) *SR-LSTM:* Rao *et al.* [21] proposed a model with two LSTM layers. The first layer learns sentence vectors to represent semantics of sentences, and the document representation is encoded by the relationship of its constituent sentences in the second layer.
9) *CoVe:* CoVe [40] is an effectual transfer learning method for NLP, which use two-layer bidirectional LSTM to obtain the contextualized word vectors. And then each word vector in GloVe with its corresponding vector in character vectors and context vectors as input sent to biattentive classification network (BCN) for classification.
10) *ULMFiT:* ULMFiT is proposed by Howard and Ruder to deal with NLP task, which contains three stages:

TABLE III
RESULTS OF OUR MODEL AGAINST MACHINE LEARNING ALGORITHMS

Model	IMDB-10	SST-5
BoW+LR [23]	-	40.86
Bigram+LR [23]	-	36.24
Bigram+SVM	39.91	36.70
Bigram+SVM	40.92	36.70
BGDL	**50.70**	**53.25**

a) language model (LM); b) pretraining, LM fine-tuning; and c) classification fine-tuning [19].

11) *ELMo:* ELMo [20], a state-of-the-art word representation model in NLP field, uses a deep biLM to pretrain word vector. It can be easily applied by existing models and improve the performance of these models.
12) *BERT:* BERT [31] is based on a multilayer bidirectional transformer [42] and is designed to pretrain deep bidirectional representations by jointly conditioning on context.

D. Results

In this section, we show the experimental results. For the convenience of description, our model is called **BGDL** when it uses the common loss function (CE), and **DL-BGDL** when it uses DL.

Table III compares BGDL model with other traditional methods (e.g., BoW + LR). The accuracy of BGDL on the datasets IMDB-10 and SST-5 is 50.7% and 53.25%, respectively. Compared with the other traditional baselines in Table III, the accuracy improvements range from 9.78%–10.79%, 12.39%–17.01% on datasets IMDB-10 and SST-5, respectively. The results show that our approach may suffer from the data sparsity problem less and capture more contextual information of features compared with traditional methods using the BoW features. Furthermore, the neural networks do better in composing the semantic representation of texts than the traditional methods for both datasets, namely IMDB-10 and SST-5 IMDB and SST.

We compare BGDL and DL-BGDL with other baselines, and the results are shown in Table IV. The common CE loss function is used both for the baselines and for BGDL during training. BGDL outperforms other neural networks on IMDB-10 datasets, whose results is 50.70%. Next, we compare against textRCNN in Table IV, the accuracy of BGDL increased by 1.02%, 3.24%, 2.54%, and 3.87% on four datasets. This demonstrates that our proposed word representation learning method are more efficient than textRCNN. As shown in Table IV, the results of DL-BGDL on the four datasets are slightly higher than BGDL's ones, whose improvements are 0.41%, 1.6%, 0.12%, and 1.88%, respectively. This illustrates that the DL suppresses the loss for easy examples and makes the network focus training on hard examples to improve the model accuracy.

We now discuss the results in Table IV relating to the recent state-of-the-art baselines CoVe, ULMFiT, ELMo, and BERT. First, for the MOOC dataset, there are results for ULMFiT and BERT, DL-BGDL is worse than ULMFiT by 1.06% and BERT by 2.12%. Second, for IMDB-2 where there are results for CoVe, ULMFiT, and BERT, DL-BGDL is better than CoVe by 1.88% and BERT by 3.56% but worse than ULMFiT by 1.72%. Third, for IMDB-10 where there are results for ULMFiT and BERT, DL-BGDL is better than them of 2.42% and 3.6%, respectively. Fourth, for SST-5 where there are results for CoVe, ELMo and BERT, DL-BGDL is better than CoVe by 1.43% and better than ELMo by 0.43%, but worse than BERT by 0.74%. Although the accuracy of DL-BGDL is not optimal results on datasets MOOC, IMDB-2, and SST-5, and the number of trainable parameters of DL-BGDL is less against them (CoVe, ULMFiT, ELMo, and BERT). For example, the parameters that BGDL needs to be train is 260 K, and it consistently outperforms other models, such as CoVe, ULMFiT, ELMo, and BERT. The CoVe model has 9.04 M parameters, ULMFiT model has 64.38 M parameters and ELMo model has 303M parameters and BERT model needs 110 M parameters, so DL-BGDL requires less computational resources and is also much faster to train. Lastly, we note that our model is a simple end-to-end architecture, while ULMFiT and ELMo require pretraining, and in particular ULMFiT consists of three stages to solve the task.

In order to demonstrate that DL is a universal loss function for all neural networks, we devised an experiment in which DL is used for partial models in baselines during the process of training. Table V shows the results of these models after using DL. By comparing the results of each model

TABLE IV
EXPERIMENTAL RESULTS FOR OUR MODEL AND OTHER COMPETITIVE NEURAL NETWORK MODELS

Model	MOOC	IMDB-2	IMDB-10	SST-5
LSTM	77.07	86.04	40.30	42.13
CNN	77.58	86.71	42.88	44.87
GRU	77.18	87.26	40.84	43.88
2-layer LSTM	77.56	89.30	42.64	47.63
CNN+LSTM	78.99	88.89	39.16	44.12
SR-LSTM [21]	-	-	44.00	-
textRCNN	79.18	88.84	48.16	49.38
CoVe [40]	-	91.80	-	53.70
ULMFiT	81.67	**95.40** [19]	48.4	53.11
ELMo [20]	-	-	-	54.70
BERT	**82.73**	90.12	47.14	**55.87**
BGDL	80.20	92.08	50.70	53.25
DL-BGDL	80.61	93.68	**50.82**	55.13

Best results in each group are in bold. The result of the ULMFiT on the IMDB-2 dataset cites from the reference [19], and the results of another three datasets are obtained from the published source code by ourselves.

TABLE V
EXPERIMENTS FOR DL

Model+DL	MOOC	IMDB-2	IMDB-10	SST-5
LSTM	78.59	87.97	48.00	49.63
CNN	78.48	89.67	46.60	46.59
GRU	78.59	89.00	47.82	51.75
2-layer LSTM	78.79	90.00	47.52	53.37
CNN+LSTM	79.60	90.37	48.16	52.38
textRCNN	80.51	89.61	47.80	50.72
DL-BGDL	**80.61**	**93.68**	**50.82**	**55.13**

All models are trained with the DL function, and we set the parameters γ, m of DL as $1, 0.9$ for models other than DL-BGDL. For DL-BGDL, our parameter settings are as shown in Table II. Best results in each group are in bold.

TABLE VI
EFFECT OF DIFFERENT STRATEGIES ON PERFORMANCE

Strategies	MOOC		IMDB-2		IMDB-10		SST-5	
	Acc	∇	Acc	∇	Acc	∇	Acc	∇
basic BGDL + CE	79.29	-	90.78	-	49.68	-	51.72	-
basic BGDL + word itself + CE	80.20	+0.91	92.08	+1.30	50.70	+1.02	53.25	+1.53

TABLE VII
TIME COSTING OF MODELS WITH DL AND CE LOSS FUNCTION

Model	MOOC		IMDB-10		SST-5	
	CE	DL	CE	DL	CE	DL
LSTM	272	256	1261	1226	247	242
CNN	97	77	440	433	62	59
GRU	239	217	1197	1057	132	116
2-layer LSTM	368	357	1463	1348	381	363
CNN+LSTM	195	173	750	747	160	154
textRCNN	660	626	2096	1930	348	337
BGDL	778	621	2564	2466	581	579

The results (lower is better) in the table represent the time (s, presenting time unit is seconds) required for the neural networks to iterate on average.

using DL in Table V, and the results of the model using CE in Table IV, we can find the results of those model with DL in Table V are almost always better than the results shown in Table IV on all datasets. The results of models mentioned in Table V compare against the results shown in Table IV, the accuracy of these models is improved by 0.41%–1.52%, 0.70%–2.96%, 0.12%–9%, and 1.34%–8.26% on the four datasets, MOOC, IMDB-2, IMDB-10, and SST-5, respectively. This proves that our novel loss function is suited to all neural networks. Moreover, it also demonstrates that DL can effectively solve the easy/hard examples problem so that improved classification accuracy can be attained.

Finally, in order to further investigate the effectiveness of each strategy proposed by us, we divide DL-BGDL into its component strategies in order to demonstrate the effectiveness of each. The results are shown in Table VI, the basic BGDL model refers to a version of BGDL which does not contain the word itself during the application of the bidirectional GRU to capture the context of the word. When only using the basic BGDL model with CE for sentiment classification, the model achieves results on the four datasets of 79.29%, 90.78%, 49.68%, and 51.72%. After that, we added the current word into its left and right context when we used the basic BGDL to extract the contextual information of the word. We were pleasantly surprised to find that the accuracy over the four datasets improved to different degrees, the improvements being 0.91%, 1.3%, 1.02%, and 1.53%, respectively. This demonstrates the point that the word itself has an effect on the meaning of the context. Formally, when we consider this impact and incorporate the solution into the model, the performance is further improved.

E. Discussion

The results in Tables IV and V indicate the effectiveness of our proposed methods. To illustrate more clearly, we carry out a detailed analysis in this section.

First, we observe the experimental results of Table IV. Our method and textRCNN outperform other models on MOOC and IMDB-10 datasets. We believe the main reason is that the contexts help us to obtain a more precise word meaning during word representation learning and avoid the influence of polysemy on classification tasks. Furthermore, the gap in accuracy between our model and textRCNN on the IMDB-2, IMDB-10, and SST-5 datasets are great. The accuracy of BGDL is 92.08%, 50.70%, and 53.25% and textRCNN is 88.84%, 48.16%, and 49.38% on the IMDB-2, IMDB-10, and SST-5 dataset, respectively. This illustrates that a bidirectional structure can capture more semantic information than a unidirectional one, especially in long documents.

Second, the word representation method proposed by this article presents a word meaning more precisely by incorporating its context. To illustrate this, we choose three words from the SST-5 dataset and display their word2vec representations along with those produced by our BGDL method in different contexts in Fig. 4. All vectors are reduced to two dimensions using principal component analysis in order to allow them to be plotted on a diagram. Thus, for example, the first part of Fig. 4 shows the word2vec representation of the word "second," which appears at the top left of the plot. The representations for second as produced by our method within four different sentences, s1, s2, s3, and s4 are also shown. As can be seen, the representations for s1–s4 are in four different positions and are also far from the word2vec representation. The same effect can also be observed for the two other words, "elegant" and "yellow." This demonstrates that the context of a word indeed results in that word having a different representation within our approach, an effect which is reflected in our improved results.

Third, when we compare the results of shallow neural networks (such as GRU, CNN, and LSTM *et al.*) in Table V with some complex networks (like textRCNN) in Table IV, we find that the accuracy of shallow neural networks is similar to that of complex neural networks. That means DL makes it possible to improve on the accuracy of shallow neural networks without compromising the complexity of the neural network. The results of the simple neural model using DL function can even match the accuracy of complex neural networks using traditional loss function. Now, with these shallow neural networks, the accuracy of the model has been greatly improved by using DL. This has profound significance for application, such as E-commerce websites, stock forecast, etc., because the time complexity of a shallow neural network is much lower than that of a deep neural network.

Forth, to evaluate more accurately the differences between using DL and CE for the model, we compute the experimental

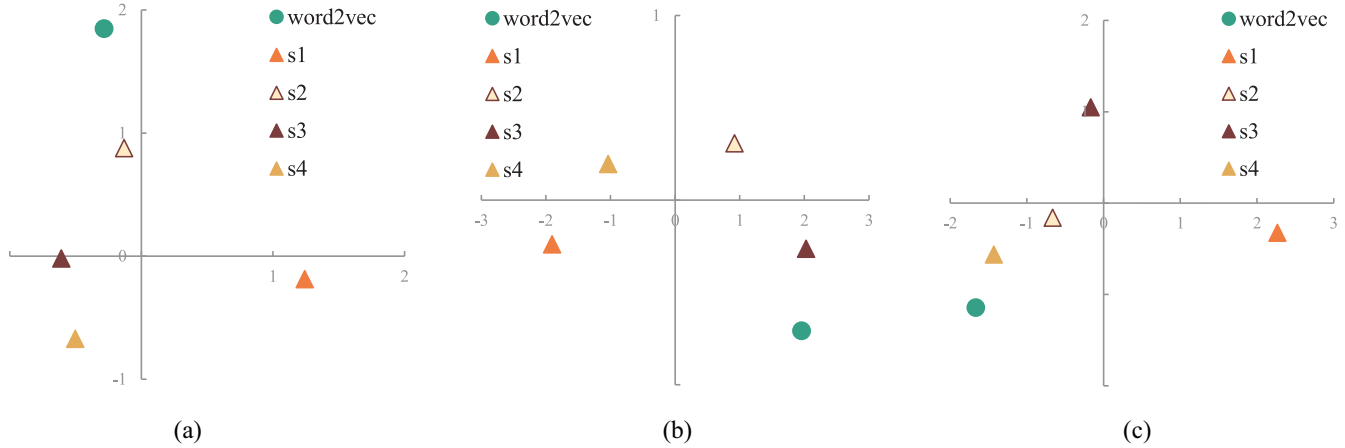

Fig. 4. Distance between a word represented by word2vec and the same word represented by BGDL within different SST-5 sentences. (a) Shows the word "second" as it appears in the following sentences: s1—*"Yet in its own aloof, unreachable way it is so fascinating you will not be able to look away for a second"*; s2—*"Those moviegoers who would automatically bypass a hip-hop documentary should give 'scratch' a second look"*; s3—*"Crush is so warm and fuzzy you might be able to forgive its mean-spirited second half"*; s4—*"The fact that the 'best part' of the movie comes from a 60-s homage to one of Demme's good films does not bode well for the rest of it."* (b) Shows the word "elegant" as it appears in the following sentences: s1—*"Handled correctly, Wilde's play is a masterpiece of elegant wit and artifice"*; s2—*"An elegant work, Food of Love is as consistently engaging as it is revealing"*; s3—*"The movie's thesis—elegant technology for the masses—is surprisingly refreshing"*; s4—*"The hard-to-predict and absolutely essential chemistry between the down-to-earth Bullock and the nonchalant Grant proves to be sensational, and everything meshes in this elegant entertainment"*. (c) Shows the word "yellow" as it appears in the following sentences: s1—*"Brilliantly written and well-acted, Yellow Asphalt is an uncompromising film"*; s2—*"Like Rudy Yellow Lodge, Eyre needs to take a good sweat to clarify his cinematic vision before his next creation and remember the lessons of the trickster spider"*; s3—*"Pumpkin struts about with 'courage' pinned to its huckster lapel while a yellow streak a mile wide decorates its back"*; s4—*"The disjointed mess flows as naturally as Jolie's hideous yellow do."*

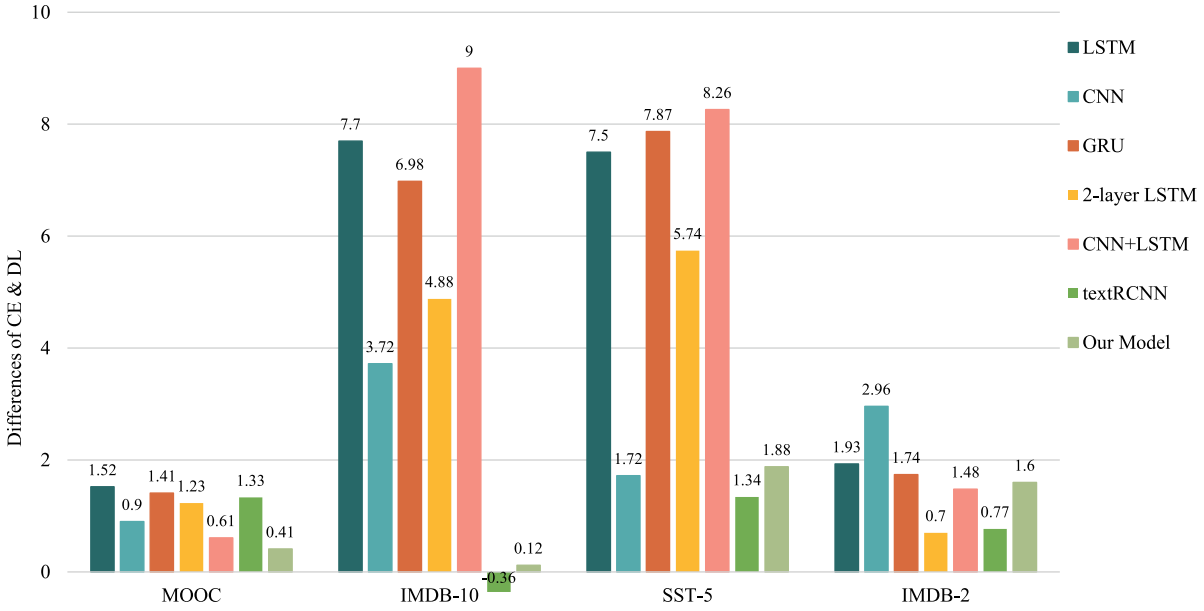

Fig. 5. Differences of DL and CE on model performance.

results for DL on each dataset minus the results for CE, as shown in Fig. 5. Each bar represents the performance difference between DL and CE for each model on the corresponding datasets. As Fig. 5 shows, the accuracy of the CNN+LSTM model improves the most on IMDB-10 dataset: with DL, its accuracy increased by 9%. Generally, we find that DL has a greater impact on simple networks (LSTM, GRU, etc.) than on complex ones. We consider the reason is that complex networks already have a strong learning ability and are thus not so sensitive to the down-weighted easy examples. For simple networks, however, their ability to learn features is weaker than complex ones. Therefore, DL focuses on training on hard examples and prevents the vast number of easy examples from overwhelming the network during training.

Lastly, analyzing our model in terms of time complexity, the parameter m [see (16)] is a threshold value at which, if the examples estimated probability by the softmax function is greater than m, its loss value is not calculated. The time costing of each model with DL or CE is shown in Table VII. We see that the time taken by the model during training is less with DL than it is with CE. We believe the main reason is that DL can reduce the time complexity of the network by filtering the loss of some easy examples via truncation factor, because the loss

of an easy example that is directly truncated is not calculated during the network training process. As a result, the models using our DL function require lower computational time.

V. Conclusion

We have applied bidirectional RecurrentNNs incorporating a novel loss function, DL, to the task of sentiment classification. Our model not only combines a word and its contextual information to present a word, but also solves the problem of easy versus hard examples to improve the accuracy of the networks. Moreover, DL ensures that the loss values for hard examples are not affected. The experiments demonstrate that our approach has a good performance on four different sentiment classification datasets. In addition, DL makes it possible to improve on the accuracy of many kinds of shallow neural networks without compromising their complexity. In particular, the results of the simple model using DL function can match the accuracy of the complex networks using traditional loss function.

Acknowledgment

The code of this article is at: https://github.com/Yolk-justlike/BGDL.

References

[1] D. D. Wu, L. Zheng, and D. L. Olson, "A decision support approach for online stock forum sentiment analysis," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 44, no. 8, pp. 1077–1087, Aug. 2014. [Online]. Available: https://doi.org/10.1109/TSMC.2013.2295353

[2] P. Ji, H.-Y. Zhang, and J.-Q. Wang, "A fuzzy decision support model with sentiment analysis for items comparison in e-commerce: The case study of http://PConline.com," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 49, no. 10, pp. 1993–2004, Oct. 2019.

[3] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proc. HLT/EMNLP*, vol. 7, 2005, pp. 347–354.

[4] R. Ren and D. Wu, "An innovative sentiment analysis to measure herd behavior," *IEEE Trans. Syst., Man, Cybern., Syst.*, to be published.

[5] A. Agarwal, B. Xie, I. Vovsha, O. Rambow, and R. Passonneau, "Sentiment analysis of Twitter data," in *Proc. Workshop Lang. Soc. Media (LSM)*, Stroudsburg, PA, USA, 2011, pp. 30–38. [Online]. Available: http://dl.acm.org/citation.cfm?id=2021109.2021114

[6] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2002, pp. 79–86.

[7] S. I. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proc. Meeting Assoc. Comput. Linguist. Assoc. Comput.*, vol. 2, 2012, pp. 90–94.

[8] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[9] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, "Neural probabilistic language models," *J. Mach. Learn. Res.*, vol. 3, no. 6, pp. 1137–1155, 2003.

[10] Q. Huang, R. Chen, X. Zheng, and Z. Dong, "Deep sentiment representation based on CNN and LSTM," in *Proc. Int. Conf. Green Informat. (ICGI)*, Aug. 2017, pp. 30–33.

[11] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2015, pp. 1422–1432.

[12] K. S. Tai, R. Socher, and C. D. Manning, "Improved semantic representations from tree-structured long short-term memory networks," in *Proc. 53rd Annu. Meeting Assoc. Comput. Linguist. 7th Int. Joint Conf. Nat. Lang. Process. Asian Federation Nat. Lang. Process.*, vol. 1, Jul. 2015, pp. 1556–1566.

[13] A. Mnih and G. Hinton, "Three new graphical models for statistical language modelling," in *Proc. Int. Conf. Mach. Learn.*, 2007, pp. 641–648.

[14] T. A. Mikolov, "Statistical language models based on neural networks," Ph.D. dissertation, Dept. Comput. Graph. Multimedia, Brno Univ. Technol., Brno, Czechia Republic, 2012.

[15] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.

[16] T. Mikolov, E. Grave, P. Bojanowski, C. Puhrsch, and A. Joulin, "Advances in pre-training distributed word representations," in *Proc. 11th Int. Conf. Lang. Resources Eval. (LREC)*, Miyazaki, Japan, May 2018, pp. 52–55.

[17] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, Oct. 2014, pp. 1532–1543. [Online]. Available: http://aclweb.org/anthology/D/D14/D14-1162.pdf

[18] T. Mikolov, I. Sutskever, C. Kai, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 26, 2013, pp. 3111–3119.

[19] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," in *Proc. 56th Annu. Meeting Assoc. Comput. Linguist. (ACL)*, vol. 1. Melbourne, VIC, Australia, Jul. 2018, pp. 328–339. [Online]. Available: https://aclanthology.info/papers/P18-1031/p18-1031

[20] M. E. Peters *et al.*, "Deep contextualized word representations," in *Proc. Conf. North Amer. Assoc. Comput. Linguist. Human Lang. Technol.*, vol. 1, 2018, pp. 2227–2237.

[21] G. Rao, W. Huang, Z. Feng, and Q. Cong, "LSTM with sentence representations for document-level sentiment classification," *Neurocomputing*, vol. 308, pp. 49–57, Sep. 2018. [Online]. Available: https://doi.org/10.1016/j.neucom.2018.04.045

[22] J. Xu, D. Chen, X. Qiu, and X. Huang, "Cached long short-term memory neural networks for document-level sentiment classification," in *Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, Austin, TX, USA, Nov. 2016, pp. 1660–1669. [Online]. Available: http://aclweb.org/anthology/D/D16/D16-1172.pdf

[23] S. Lai, L. Xu, K. Liu, and J. Zhao, "Recurrent convolutional neural networks for text classification," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 2267–2273.

[24] J. L. Elman, "Finding structure in time," *Cogn. Sci.*, vol. 14, no. 2, pp. 179–211, 1990.

[25] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional LSTM networks," in *Proc. IEEE Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 4, 2005, pp. 2047–2052.

[26] R. Cai, X. Zhang, and H. Wang, "Bidirectional recurrent convolutional neural network for relation classification," in *Proc. Meeting Assoc. Comput. Linguist.*, 2016, pp. 756–765.

[27] K. Cho *et al.*, "Learning phrase representations using RNN encoder–decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2014, pp. 1724–1734.

[28] H. Schwenk, L. Barrault, A. Conneau, and Y. LeCun, "Very deep convolutional networks for text classification," in *Proc. 15th Conf. Eur. Assoc. Comput. Linguist. (EACL)*, vol. 1. Valencia, Spain, Apr. 2017, pp. 1107–1116. [Online]. Available: https://aclanthology.info/papers/E17-1104/e17-1104

[29] D. Tang, B. Qin, X. Feng, and T. Liu, "Effective LSTMS for target-dependent sentiment classification," in *Proc. 26th Int. Conf. Comput. Linguist. Tech. Papers (COLING)*, Osaka, Japan, Dec. 2016, pp. 3298–3307. [Online]. Available: http://aclweb.org/anthology/C/C16/C16-1311.pdf

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

[31] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," *CoRR*, vol. abs/1810.04805, Oct. 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[32] H. T. Le, C. Cerisara, and A. Denis, "Do convolutional networks need to be deep for text classification?" in *Proc. Workshops 32nd AAAI Conf. Artif. Intell.*, New Orleans, LA, USA, Feb. 2018, pp. 29–36. [Online]. Available: https://aaai.org/ocs/index.php/WS/AAAIW18/paper/view/16578

[33] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Nat. Lang. Process. (EMNLP)*, Oct. 2014, pp. 1746–1751. [Online]. Available: http://aclweb.org/anthology/D/D14/D14-1181.pdf

[34] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.

[35] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[36] G. Paltoglou and M. Thelwall, "A study of information retrieval weighting schemes for sentiment analysis," in *Proc. 48th Annu. Meeting Assoc. Comput. Linguist. (ACL)*, Uppsala, Sweden, Jul. 2010, pp. 1386–1395. [Online]. Available: http://www.aclweb.org/anthology/P10-1141
[37] L. Qu, G. Ifrim, and G. Weikum, "The bag-of-opinions method for review rating prediction from sparse text patterns," in *Proc. 23rd Int. Conf. Comput. Linguist. (COLING)*, Beijing, China, Aug. 2010, pp. 913–921. [Online]. Available: http://aclweb.org/anthology/C10-1103
[38] R. Xia and C. Zong, "Exploring the use of word relation features for sentiment classification," in *Proc. 23rd Int. Conf. Comput. Linguist. Posters (COLING)*, vols. 23–27. Beijing, China, Aug. 2010, pp. 1336–1344. [Online]. Available: http://aclweb.org/anthology/C/C10/C10-2153.pdf
[39] T. Mikolov, K. Chen, G. Corrado, and J. Dean. (2013). *Efficient Estimation of Word Representations in Vector Space*. [Online]. Available: https://arxiv.org/abs/1301.3781
[40] B. McCann, J. Bradbury, C. Xiong, and R. Socher, "Learned in translation: Contextualized word vectors," in *Proc. Adv. Neural Inf. Process. Syst. 30th Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 6297–6308. [Online]. Available: http://papers.nips.cc/paper/7209-learned-in-translation-contextualized-word-vectors
[41] M. E. Peters, W. Ammar, C. Bhagavatula, and R. Power, "Semi-supervised sequence tagging with bidirectional language models," in *Proc. 55th Annu. Meeting Assoc. Comput. Linguist. (ACL)*, vol. 1. Vancouver, BC, Canada, Jul./Aug. 2017, pp. 1756–1765. [Online]. Available: https://doi.org/10.18653/v1/P17–1161
[42] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst. 30th Annu. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, Dec. 2017, pp. 5998–6008. [Online]. Available: http://papers.nips.cc/paper/7181-attention-is-all-you-need
[43] R. H. R. Hahnloser and H. S. Seung, "Permitted and forbidden sets in symmetric threshold-linear networks," in *Proc. Adv. Neural Inf. Process. Syst. 13th Neural Inf. Process. Syst. (NIPS)*, Denver, CO, USA, 2000, pp. 217–223. [Online]. Available: http://papers.nips.cc/paper/1793-permitted-and-forbidden-sets-in-symmetric-threshold-linear-networks
[44] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 12, pp. 2493–2537, Feb. 2011.
[45] Q. Diao, M. Qiu, C.-Y. Wu, A. J. Smola, J. Jiang, and C. Wang, "Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS)," in *Proc. ACM 20th Int. Conf. Knowl. Disc. Data Min.*, 2014, pp. 193–202.
[46] R. Socher *et al.*, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proc. Conf. Empirical Methods Nat. Lang. Process. (ACL)*, 2013, pp. 1631–1642.
[47] F. Kokkinos and A. Potamianos, "Structural attention neural networks for improved sentiment analysis," in *Proc. 15th Conf. Eur. Assoc. Comput. Linguist.*, 2017, pp. 586–591.
[48] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA Neural Netw. Mach. Learn.*, vol. 4, pp. 26–30, Nov. 2012.
[49] A. Graves, "Long short-term memory," in *Supervised Sequence Labelling With Recurrent Neural Networks*. Heidelberg, Germany: Springer, 2012, pp. 37–45.
[50] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
[51] A. Yenter and V. Abhishek, "Deep CNN-LSTM with combined kernels from multiple branches for IMDb review sentiment analysis," in *Proc. 8th IEEE Annu. Ubiquitous Comput. Electron. Mobile Commun. Conf.*, 2017, pp. 540–546.

Xia Sun received the Ph.D. degree in computer science from Xi'an Jiaotong University, Xi'an, China, in 2006.

She is an Associate Professor with the School of Information Science and Technology, Northwest University, Xi'an. Recent projects have included causality extraction from educational data, and relationship extraction from bioinformatics text. She has coauthored 40 articles and edited or coedited 4 books. Her current research interests include natural language processing and machine learning.

Dr. Sun has served as a Reviewer for the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, and the *Chinese Journal of Electronics*. She has also reviewed for the IEEE International Conference on Computational Science and Engineering.

Yi Gao received the B.S. degree in computer science from Northwest University, Xi'an, China, in 2017, where she is currently pursuing the M.S. degree in computer science.

Her current research interests include neural networks and natural language processing. In addition to sentiment classification, she has applied deep learning to student performance prediction.

Richard Sutcliffe received the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 1989.

He is an Associate Professor with Northwest University, Xi'an, China. His current research interests include natural language processing, information retrieval, and music information retrieval. He has coauthored 101 articles and coedited 3 books and 10 conference proceedings. Recent projects have included persuasive conversational agents, public sector message classification, analysis of classical music texts, and personality and translation ability.

Dr. Sutcliffe has reviewed for *Artificial Intelligence Review*, *Computational Linguistics*, *Computers and the Humanities*, *Information Processing and Management*, *Information Retrieval Journal*, *Journal of Natural Language Engineering*, and *Journal Traitement Automatique des Langues*. Conferences he has reviewed for include ACL, CIKM, COLING, IJCNLP, LREC, NAACL-HTL, and SIGIR.

Shou-Xi Guo received the B.S. degree in software engineering from the Shenyang University of Chemical Technology, Shenyang, China, in 2017. He is currently pursuing the M.S. degree in computer science with Northwest University, Xi'an, China.

His current research interests include neural networks and natural language processing. Current projects include the application of convolutional neural networks to ancestry estimation of skulls, and the use of recurrent convolutional neural networks to categorize textual blog posts.

Xin Wang received the B.S. degree in computer science from Northwest University, Xi'an, China, in 2018, where she is currently pursuing the M.S. degree in computer science.

Her research combines neural networks with natural language processing. Recent work develops new neural network algorithms and frameworks for sentiment classification.

Jun Feng (M'07) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2006.

She is a Professor with the School of Information Science and Technology, Northwest University, Xi'an, China. She has coauthored 132 articles and coedited three books. Her current research interests include pattern recognition and machine learning, especially in the fields of medical imaging analysis and intelligent education. Recent projects have included medical image analysis with deep learning, and intelligent education based on AI and brain–human interaction.

Prof. Feng has reviewed for many journals, including the IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE ACCESS, *EURASIP Journal on Image and Video Processing*, *Multimedia Tools and Applications*, *Journal of Digital Information Management*, *Canadian Journal of Cardiology*, *Journal of Clinical and Aesthetic Dermatology*, OPE, and INFPHY. Conferences she has reviewed for include IEEE-VR, MICCAI, SIGCSE, IWCSE, and CompEd. She is a member of ACM.