Better Word Representations with Word Weight

Gege Song, Xianglin Huang, Gang Cao, Zhulin Tao, Wei Liu, Lifang Yang
School of Computer Science and Cybersecurity
Communication University of China
Beijing, China
sggever@163.com, {huangxl,gangcao,zhulintao}@cuc.edu.cn, cuclw12@163.com, yanglifang@cuc.edu.cn

***Abstract*—As a fundamental task of natural language processing, text classification has been widely used in various applications such as sentiment analysis and spam detection. In recent years, the continuous-valued word embedding learned by neural network attaches extensive attentions. Although word embedding achieves impressive results in capturing similarities and regularities between words, it fails to highlight important words for identifying text category. Such deficiency can be attenuated by word weight, which conveys word contribution in text categorization. Toward this end, we propose an effective text classification scheme by incorporating word weight into word embedding in this paper. Specifically, in order to enrich word representation, bidirectional gated recurrent units (Bi-GRU) is employed to grasp the context information. Then word weights yielded by term frequency (TF) are first used to modulate the word representation of Bi-GRU for constructing text representation. Extensive experimental results on several large text datasets verify that the accuracy of our proposed text classification scheme outperforms the state-of-the-art ones.**

***Index Terms*—text classification, word embedding, word weight, Bi-GRU, TF**

I. Introduction

Text classification is a fundamental problem in natural language processing (NLP). The task is to annotate a given text sequence with one or multiple class label(s) describing its textual content. Text classification has many applications, such as sentiment classification [1,2] and spam detection [3]. Aiming to process by existing machine learning methods, text data is typically first transformed into numerical data by some methods, *i.e.*, word embedding. Recently, neural models have been employed to learn continuous-valued word embedding, including convolutional neural network (CNN) [4-6] and recurrent neural network (RNN) [7,8]. The similarities and regularities between words can be effectively captured by such word embedding [9-11].

Deep learning has been widely used in text classification depending on the effectiveness of word embedding. The different kernel sizes of CNNs are employed to extract text features for sentence classification [12]. Different types of RNN are widely used in natural language processing tasks, such as long short-term memory (LSTM) [7] and gated recurrent units (GRU) [13]. LSTM is cooperated with CNN to improve the performance of text classification [14,15]. A GRU-based text modeling method is explored for sentiment classification [16]. Although the state-of-the-art text classification results have been achieved by using the above methods, poor performances can be occurred in a small or noisy dataset. It is noted that each word enjoys different contribution in text understanding and inference. The important words for text category classification can not be highlighted by word embedding. Therefore, it is rather significant to investigate the approaches revealing the contribution of word for text category classification.

The prevailing approaches are to exploit attention mechanism attending differentially to more and less important content when constructing the text representation. Attention mechanism is directly learned from word/sentence representation via complex networks. Yang *et al.* [17] apply neural networks at the word- and sentence-level for computing the attention allocation of word/sentence. Zhou *et al.* [18] design double-layer feed-forward neural networks for grasping important words/sentences. Complex networks lead to longer running time and higher requirement for equipment. Hence, it is rather significant to investigate an attention mechanism which enjoys relatively simple structure and capability of grasping important words.

Attention mechanism can be interpreted as a weight calculation method based on neural network. In traditional methods, many mature and effective methods for calculating word weight have been developed, such as TFIDF [19] and LDA [20]. However, there are less works incorporating such priori knowledge into the deep learning methods based on word representation. In this paper, we propose such a solution by injecting word weight into word representation for the further improvement of text classification. The whole scheme as illustrated in Fig. 1. Word representation is yielded by a bidirectional gated recurrent units (Bi-GRU) [21]. Term frequency (TF) is computed for revealing the contribution of word for text category classification, which is fused into word representation to construct the text representation. Extensive experiments on several text-classification tasks have demonstrated the effectiveness of our model and provided state-of-the-art results on benchmark datasets.

The rest of this paper is organized as follows. The text classification scheme is proposed in Section II. Experimental results and discussion are reported in Section III, which is followed by the conclusion drawn in Section IV.

This work was supported by the National Natural Science Foundation of China (grant nos.61401408, 61772539) and the Fundamental Research Funds for the Central Universities (grant nos. CUC2019B021, 3132017XNG1742).

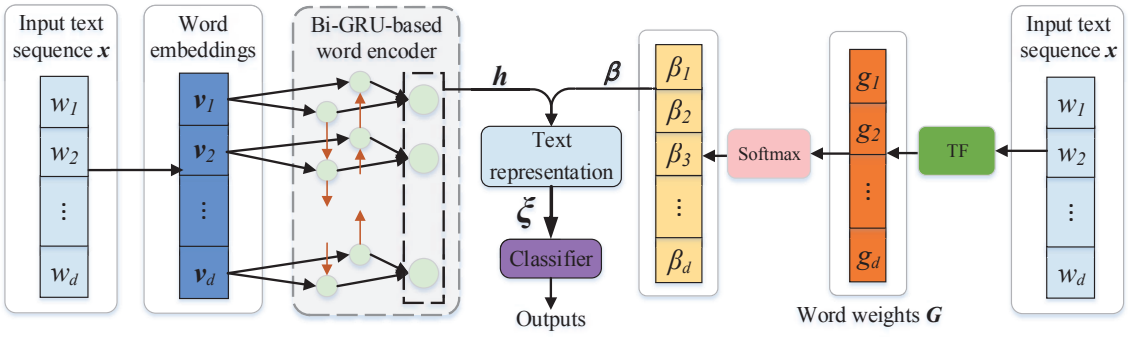

Fig. 1. Illustration of proposed model for text classification.

II. Proposed Text Classification Scheme

A. Proposed model for text classification

The proposed model projects the raw text into a representation vector by injecting TF into word representation of Bi-GRU, on which we build a classifier to perform text classification. The overall architecture of our proposed model is shown in Fig. 1.

Given an input text sequence $\boldsymbol{x} = [w_1, \cdots, w_d]$ of length d composed of words and each word w_i can be represented as a low dimensional, continuous and real-valued vector by word embedding. Therefore, the text sequence $\boldsymbol{x}$ is represented: $\boldsymbol{V} = [\boldsymbol{v}_1, \cdots, \boldsymbol{v}_d] \in \boldsymbol{R}^{d \times P}$, where $\boldsymbol{v}_i \in \boldsymbol{R}^P$, P is the dimensionality of the embedding space.

Bi-GRU-based Word Encoder. As a variant of LSTM, GRU [13] is popular due to less parameters and easy convergence. GRU is designed to remember previously read values for any given period of time and makes decisions based on such knowledge. The characteristic of GRU makes it more suitable for the extraction of text representation, because each word of a sentence is closely connected with the surrounding words, *i.e.*, previous and forthcoming words.

In order to capture contextual information of text sequence, Bi-GRU [21] is employed to obtain annotations of words by integrating information from both directions for words. The Bi-GRU contains a forward GRU $\overrightarrow{h}$ which reads the text from w_1 to w_d and a backward GRU $\overleftarrow{h}$ which reads from w_d to w_1:

$$\begin{aligned} \overrightarrow{h_i} &= \overrightarrow{GRU}(v_i), i \in [1, d], \\ \overleftarrow{h_i} &= \overleftarrow{GRU}(v_i), i \in [d, 1] \end{aligned} \tag{1}$$

The forward hidden state $\overrightarrow{h}_i$ and backward hidden state $\overleftarrow{h}_i$ are concatenated to describe the annotation of word w_i. $\boldsymbol{h}_i = [\overrightarrow{h}_i, \overleftarrow{h}_i]$ summarizes the information of the whole text centered around w_i.

Weight-based Text representation. After using Bi-GRU, the annotation for each word containing context information can be obtained. To construct a better text representation, TF, the occurrence times of a word, is employed to reveal the importance of word.

Term frequencies of text sequence $\boldsymbol{x}$ are firstly computed, *i.e.*, $\boldsymbol{T} = [t_1, \cdots, t_d] \in \boldsymbol{R}^d$, where t_i signifies the repetition times of the word w_i. Then $\boldsymbol{T}$ is normalized by dividing with the total frequency of $\boldsymbol{x}$ for obtaining the relative information, *i.e.*,

$$g_i = \frac{t_i}{\sum_{j=1}^{d} t_j} \tag{2}$$

Word weights $\boldsymbol{G} = [g_1, \cdots, g_d]$ produce a set of standardized weights $[\beta_1, \cdots, \beta_d]$ via Softmax function, each corresponding to a particular word.

$$\boldsymbol{\beta} = Softmax\,(\boldsymbol{G}) \tag{3}$$

Finally, the text sequence representation $\boldsymbol{\xi}$ that summarizes all the information of a text is obtained via a weighted sum of the word annotations based on the weights:

$$\boldsymbol{\xi} = \sum_{i=1}^{d} \beta_i \boldsymbol{h}_i \tag{4}$$

B. Training with proposed model

Semantic labels are highly generalized descriptions of categories, and the classification network should have a good capability to distinguish such labels. Therefore, text representation $\boldsymbol{\xi}$ and labels are simultaneously used to learn the potential space in which both can achieve good classification results. The training loss can be expressed by equation (5).

$$\begin{aligned} Loss &= Loss^{(1)} + Loss^{(2)}, \\ Loss^{(i)} &= \tfrac{1}{N} \textstyle\sum_{n=1}^{N} CE(\boldsymbol{y}_n, \boldsymbol{I}^{(i)}) \end{aligned} \tag{5}$$

where N signifies the number of texts, $CE(\cdot, \cdot)$ denotes the cross entropy between two probability vectors, $\boldsymbol{I}^{(i)} = Softmax(\boldsymbol{W}_1 \boldsymbol{F}^{(i)} + \boldsymbol{b}_1)$, $\boldsymbol{W}_1$ and $\boldsymbol{b}_1$ are trainable parameters and $\boldsymbol{F}^{(1)}, \boldsymbol{F}^{(2)}$ denote $\boldsymbol{\xi}$ and the embedding vectors of semantic labels, respectively.

III. Experimental Results and Discussion

A. Dataset, performance measures and experimental details

Datasets. Four standard benchmark datasets which are used in [22] are also utilized to evaluate the effectiveness of our model. The statistics of the data sets are summarized in Table I.

TABLE I
STATISTICS OF FOUR DATASETS.

Datasets	Classes	Train Samples	Test Samples
DBPedia	14	560,000	70,000
AGNews	4	120,000	7,600
Yelp Review Polarity	2	560,000	38,000
Yelp Review Full	5	650,000	38,000

- DBPedia: DBpedia ontology dataset is collected from Wikipedia. This dataset is constructed by picking fourteen nonoverlapping classes from DBpedia 2014.
- AGNews: This dataset is divided into four categories by the topic of content, *i.e.*, World, Entertainment, Sports and Business. Each document is composed of title and description.
- Yelp Review Polarity: The same set of text reviews from Yelp Dataset Challenge in 2015. Compared with Yelp Review Full, this database contains only two emotional levels, *i.e.*, positive and negative. 1 and 2 in Yelp Review Full dataset correspond to the negative in this database, and 4 and 5 correspond to the positive.
- Yelp Review Full: The Yelp Review Full dataset is obtained from the Yelp Dataset Challenge in 2015. The sentences are classified under five grades from 1 to 5 according to the emotions expressed of sentences. The higher the score, the stronger the positive emotions of sentences.

Performance measures. In order to thoroughly measure our model and the baselines, the accuracy is employed as evaluation metric. For each text, we generate the highest ranked labels and compare the generated labels to the ground truth labels. The accuracy is the number of correctly annotated labels divided by the number of generated labels.

Experimental details. The GloVe word embeddings with 300-dimensional [11] is utilized as initialization for word embeddings and label embeddings in our model. Note that words not contained in the GloVe are initialized from a uniform distribution with range $[-0.01, 0.01]$. Rectified linear units is selected as the activation function in our model. The final classifier is implemented as an MLP layer followed by a softmax function. The Adam Optimizer [23] is used to train our model's parameters, the learning rate is initialized to 0.001 and a minibatch size is set to 100. Dropout regularization [24] is employed on the final MLP layer, with dropout rate 0.5. The model is implemented using Tensorflow and is trained on GPU Tesla.

B. Competitor Models

In this section, we display the accuracies of our proposed method, together with several state-of-the-art competitor networks. Detailed analyses are given for better understanding.

- Bag-of-words: For each dataset, the bag-of-words model is constructed by selecting 50,000 most frequent words from the training subset. The term frequency is used as the features [22].
- Large/Small word CNN: The word-based ConvNet is used to classify texts. The lookup table is chose to learn word representations and the embedding size is 300. Large and small stand for the features of 1024 dimension and the feature of 256 dimension, respectively [22].
- LSTM: The text feature vector is formed by taking mean of the outputs of all word-based LSTM cells [22].
- FastText: FastText extends word2vec to tackle sentence and text classification. The word representations are averaged into a text representation, which is in turn fed to a linear classifier [25].
- VDCNN: VDCNN operates directly at the character level and uses only small convolutions and pooling operations. And 29 convolutional layers are contained in this model [26].
- SWEM: SWEM investigates the raw modeling capacity of word embeddings, which has no additional compositional parameters to encode natural language sequences [27].
- Bi-BloSAN: Bi-BloSAN splits the entire sequence into blocks, and applies an intra-block self-attention networks (SAN) to each block for modeling local context, then applies an inter-block SAN to the outputs for all blocks to capture long-range dependency. The feature-level attention is used to handle the variation of contexts around the same word, and use forward/backward masks to encode temporal order information [28].
- LEAM: LEAM makes use of the label embedding to improve text classification. LEAM is implemented by jointly embedding the word and label in the same latent space, and the text representations are constructed directly using the text-label compatibility [29].

TABLE II
TEST ACCURACIES OF DIFFERENT MODELS. NUMBERS ARE IN PERCENTAGE.

Models	DBPedia	AGNews	Yelp P.	Yelp F.
Bag-of-words [22]	96.61	88.81	92.24	57.99
Small word CNN [22]	98.15	89.13	94.46	58.59
Large word CNN [22]	98.28	91.45	95.11	59.48
LSTM [22]	98.55	86.06	94.74	58.17
fastText [25]	98.60	92.50	95.70	63.90
VDCNN [26]	98.71	91.27	95.72	64.26
SWEM [27]	98.42	92.24	93.76	61.11
Bi-BloSAN [28]	98.77	93.32	94.56	62.13
LEAM [29]	99.02	92.45	95.31	64.09
Proposed	**99.03**	**93.33**	**96.63**	**66.62**

The accuracies of all methods are shown in Table II. The accuracy of our proposed model outperforms other state-of-the-art baselines on all the datasets. In particular, although our architecture is slightly superior to the best baseline on DBPedia and AGNews datasets, it achieves higher accuracy

TABLE III
CONFUSION MATRIX OF PROPOSED METHOD ON YELP F. DATASET. THE RELATIVE NUMBER OF CORRECT AND INCORRECT CLASSIFICATIONS ARE SHOWN AND THE NUMBER ARE WRITTEN IN PERCENT. 1-5 DENOTE FIVE GRADES.

	1	2	3	4	5
1	**77.64**	19.74	1.33	0.49	0.8
2	18.05	**63.89**	15.74	1.57	0.75
3	2.48	19.91	**58.95**	16.88	1.78
4	0.61	1.34	16.72	**61.13**	20.20
5	0.68	0.45	1.50	25.88	**71.49**

TABLE IV
TEST ACCURACIES OF PROPOSED MODEL WITH DIFFERENT SETTING OF WORD ENCODER LAYER. NUMBERS ARE IN PERCENTAGE.

Word encoders	DBPedia	AGNews	Yelp P.	Yelp F.
Without encoder	98.43	92.14	94.00	64.73
CNN	98.57	92.96	95.79	62.93
RNN	98.49	91.89	94.08	61.04
Bi-RNN	98.15	92.11	93.88	60.27
GRU	98.93	92.78	95.94	64.82
Bi-GRU	**99.03**	**93.33**	**96.63**	**66.62**

over other models on Yelp P. and Yelp F. datasets by up to 0.91% and 2.36% respectively. In DBPedia and AGNews databases, a large number of words with similar semantics are used to represent the same topic and some words may appear only once. Instead, the Yelp P. and Yelp F. databases reuse words with significant emotions. This is the main reason why our proposed method can gain the various improvements in different databases. Meanwhile, the effectiveness of our method can further be verified by the important advancement in the accuracy on Yelp P. and Yelp F. datasets, because text data in the both datasets is closer to real world.

Table III shows the confusion matrix achieved by proposed method on Yelp F. dataset. Actuals belong on the side of the confusion matrix and predictions are across the top. The results show the greatest precision is 77.64% for the grade 1. Due to the small differences between adjacent grades, distinctions between such grades are blurred, such as the grade 2 seems to be difficult to distinguish from the grade 1 and 3. Therefore, the classification effect to the grades tends to bad and the accuracy rate is only 66.62%.

C. Bi-GRU-based Word Encoder

In order to investigate the setting of word encoder layer, we compare different models:

- Proposed without encoder: Word embedding $\boldsymbol{V}$ is directly combined with the word weight $\boldsymbol{\beta}$ to form text representation, *i.e.*, the proposed model without word encoder layer.
- Proposed with encoder: The word encoder is set as CNN, RNN, Bi-RNN, GRU and Bi-GRU, respectively.

TABLE V
TEST ACCURACIES OF OUR PROPOSED MODEL WITH DIFFERENT WEIGHT METHODS. NUMBERS ARE IN PERCENTAGE.

Word weights	DBPedia	AGNews	Yelp P.	Yelp F.
Without weight	98.85	92.42	95.57	64.26
TFIDF	98.99	93.03	96.56	66.44
TF without normalization	99.00	93.11	93.08	64.93
TF	**99.03**	**93.33**	**96.63**	**66.62**

As observed from Table IV, the model with Bi-GRU achieves the best accuracy and the RNN-based models yield worse accuracies. The result of model with CNN is superior to the RNN-based models, while it is inferior to the GRU-based models. It can be explained by their network architecture. Long-term dependencies cannot be resolved in RNN, which is a fatal disadvantage for long text. Although CNN is incapable of catching the context relationship, the key words of the text can be well captured. And due to the realization of temporal memory function in GRU, the GRU-based models are better than the RNN-based models. Meanwhile, the performance of the model without word encoder is also investigated, which is much lower than that of the model with Bi-GRU. The model with Bi-GRU outperforms the closest result by more than 1.19% except DBPedia database, which can be shown the importance and effectiveness of the word encoder based on Bi-GRU.

D. Impacts of Word Weight

The influence of different word weight calculation method on text classification is also evaluated.

- Proposed without weight: Word representation $\boldsymbol{h}$ is directly utilized for classification without combining word weight $\boldsymbol{\beta}$, *i.e.*, the proposed model without weight.
- TFIDF: TF is replaced with term frequency-inverse document frequency (TFIDF) to construct text representation.
- TF without normalization: Word weights $\boldsymbol{G}$ are substituted with the repetition times of words $\boldsymbol{T}$ to feed into Softmax, *i.e.*, TF without normalization.

Table V shows the various settings of word weight calculation method and their accuracies. The model with TF is superior to the model with TFIDF. The main reason may be that the impact of important words related to text categories can be minimized by inverse document frequency. The model with TF is better than the model with unnormalized TF. This may be due to the fact that normalized information can better exhibit the relative information between words. Besides, the model without weight is surveyed and the performance is decreased significantly. This further proves that the word weight can indeed improve the classification performance of the network.

IV. CONCLUSION

Text classification becomes increasingly essential in NLP applications. In this paper, we propose a novel text classification framework by incorporating priori knowledge into the

deep learning methods based on word representation. TF is directly integrated into word annotations generated by Bi-GRU for constructing text representation. Such operation can highlight the important words and enable model to capture word-level features more effectively. Evaluation results on four benchmark text classification datasets demonstrate that our proposed scheme achieves higher accuracy on all datasets and outperforms some state-of-the-art methods. We believe that the potential of combining traditional methods and deep learning to address text classification remains unexplored.

REFERENCES

[1] R. He, W. S. Lee, H. T. Ng, and D. Dahlmeier, "Exploiting Document Knowledge for Aspect-level Sentiment Classification," in *Proc. ACL,* Melbourne, Australia, 2018, pp. 579–585.

[2] Z. Lei, Y. Y., M. Y., and Y. L., "A Multi-sentiment-resource Enhanced Attention Network for Sentiment Classification," in *Proc. ACL,* Melbourne, Australia, 2018, pp. 758–763.

[3] X. Wang, K. Liu, and J. Zhao, "Handling Cold-Start Problem in Review Spam Detection by Jointly Embedding Texts and Behaviors," in *Proc. ACL,* Vancouver, Canada, 2017, pp. 366–376.

[4] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. ACL,* Baltimore, MD, USA, 2014, pp. 655–665.

[5] Y. Zhang, D. Shen, G. Wang, Z. Gan, R. Henao, and L. Carin, "Deconvolutional paragraph representation learning," in *Proc. NIPS,* Long Beach, CA, USA, 2017, pp. 4172–4182.

[6] D. Shen, Y. Zhang, R. Henao, Q. Su, and L. Carin, "Deconvolutional latent-variable model for text sequence matching," in *Proc. AAAI,* New Orleans, Louisiana, USA, 2018, pp. 5438–5445.

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

[8] W. Wang, Z. Gan, Wenqi Wang, Dinghan Shen, J. Huang, Wei Ping, Sanjeev Satheesh, and L. Carin, "Topic compositional neural language model," in *Proc. AISTATS,* Playa Blanca, Lanzarote, Canary Islands, Spain, 2018, pp. 356–365.

[9] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Proc. NIPS,* Lake Tahoe, Nevada, USA, 2013, pp. 3111–3119.

[10] T. Mikolov, K. Chen, G. Corrado, J. Dean, "Efficient Estimation of Word Representations in Vector Space," in *Proc. ICLR,* Scottsdale, Arizona, USA, 2013, Available: http://arxiv.org/abs/1301.3781.

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. EMNLP,* Doha, Qatar, 2014, pp. 1532–1543.

[12] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. EMNLP,* Doha, Qatar, 2014, pp. 1746–1751.

[13] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. (2014). "Empirical evaluation of gated recurrent neural networks on sequence modeling." [Online]. Available: http://arxiv.org/abs/1412.3555.

[14] Q. Vo, H. Nguyen, B. Le, and M. Nguyen, "Multi-channel LSTM-CNN model for Vietnamese sentiment analysis," in *Proc. KSE,* Hue, Vietnam, 2017, pp. 24–29.

[15] X. Wei, H. Lin, L. Yang and Y. Yu, "A Convolution-LSTM Based Deep Neural Network for Cross-Domain MOOC Forum Post Classification," *Information*., vol. 8, no. 3, pp. 92, 2017.

[16] D. Tang, B. Qin, and T. Liu, "Document modeling with gated recurrent neural network for sentiment classification," in *Proc. EMNLP,* Lisbon, Portugal, 2015, pp. 1422–1432.

[17] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proc. NAACL-HLT,* San Diego California, USA, 2016, pp. 1480–1489.

[18] X. Zhou, X. Wan, and J. Xiao, "Attention-based lstm network for cross-lingual sentiment classification," in *Proc. EMNLP,* Austin, Texas, USA, 2016, pp. 247–256.

[19] G. Salton, Y. T Clement, "On the construction of effective vocabularies for information retrieval," proceedcings of the 1973 meeting on Programming Languages and Information Retrieval. New York:ACM,1973:11.

[20] D. M. Blei, A. Y. Ng, M. I. Jordan, "Latent Dirichlet allocation," *Journal of Machine Learning Research*., vol. 3, pp. 993–1022, 2003.

[21] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR,* San Diego, CA, USA, 2015.

[22] X. Zhang, J. Zhao, and Y. LeCun, "Character-level convolutional networks for text classification," in *Proc. NIPS,* Montreal, Quebec, Canada, 2015, pp. 649–657.

[23] D. P Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR,* San Diego, CA, USA, 2015.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*., vol. 15, no. 1, pp. 1929–1958, 2014.

[25] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. (2016). "Bag of tricks for efficient text classification." [Online]. Available: https://arxiv.org/abs/1607.01759.

[26] A. Conneau, H. Schwenk, L. Barrault, and Y. Lecun, "Very deep convolutional networks for text classification," in *Proc. EACL,* Valencia, Spain, 2017, pp. 1107–1116.

[27] D. Shen, G. Wang, W. Wang, M. R. Min, Q. Su, Y. Zhang, C. Li, R. Henao, and L. Carin, "Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms," in *Proc. ACL,* Melbourne, Australia, 2018, pp. 440–450.

[28] T. Shen, T. Zhou, G. Long, J. Jiang, and C. Zhang, "Bi-directional block selfattention for fast and memory-efficient sequence modeling," in *Proc. ICLR,* Vancouver, Canada, 2018.

[29] G. Wang, C. Li, , W. Wang, Y. Zhang, D. Shen, X. Zhang, R. Henao and L. Carin, "Joint Embedding of Words and Labels for Text Classification," in *Proc. ACL,* Melbourne, Australia, 2018, pp. 2321–2331.