ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20190506102016
Original scientific paper

An Online Word Vector Generation Method Based on Incremental Huffman Tree Merging

Kui QIAN*, Lei TIAN, Xiulan WEN, Zhenzhong SONG

Abstract: Aiming at high real-time performance processing requirements for large amounts of online text data in natural language processing applications, an online word vector model generation method based on incremental Huffman tree merging is proposed. Maintaining the inherited word Huffman tree in existing word vector model unchanged, a new Huffman tree of incoming words is constructed and ensures that there is no leaf node identical to the inherited Huffman tree. Then the Huffman tree is updated by a method of node merging. Thus based on the existing word vector model, each word still has a unique encoding for the calculation of the hierarchical softmax model. Finally, the generation of incremental word vector model is realized by using neural network on the basis of hierarchical softmax model. The experimental results show that the method could realize the word vector model generation online based on incremental learning with faster time and better performance.

Keywords: Huffman tree; hierarchical softmax; incremental learning; neural network; online word vector

1 INTRODUCTION

Since the word2vec model is proposed by Mikolov et al. (2013) [1], academia and industry have set off a wave of research in the field of natural language processing (NLP) in recent years [2]. The distributed vectors of words represent a certain semantic or grammatical feature, that has been shown to be useful in various NLP tasks, such as text classification [3, 4], semantic analysis [5-7]. However, since the initial feature vector of word is randomly generated, the generated word vector is different each time. If each corpus data is trained separately, the generated word vector will have an error impact on subsequent classification, clustering, predicting and other processing tasks. In particular, when incremental small sample data needs to be processed, it will take a lot of time and computational resources if all of words are retrained. Therefore, it is necessary to combine the old corpus and new corpus for incremental training.

In order to generate the word vector faster, Pennington et al. (2014) proposed a global vector word model Glove [8], that it does not need to go through each corpus for local word vector optimization, but only global optimization of the matrix. Ji et al. (2019) calculated word2vec model by paralleling in shared and distributed memory [9], reducing network communication and keeping the model synchronized effectively when number of nodes increases.

However, traditional word2vec, Glove, and parallel training are static, batch learning modes, that do not support incremental learning, only for offline training. As mentioned above, in practice, training corpus is usually not available all at once, but is gradually obtained over time. It will take a lot of time and space to retrain all the data after the new corpus arrives. Especially when the new corpus is relatively smaller than the existing corpus, it will be much less efficient to retrain the word vectors with the combined corpus. Incremental learning method is practical and necessary.

As known, word2vec employed two techniques called hierarchical softmax and negative sampling [10, 11]. Hierarchical softmax constructs a Huffman tree to index all the words in a corpus as leaves, while negative sampling is developed based on noise contrastive estimation. It is empirically shown that hierarchical softmax performs better for infrequent words while negative sampling performs better for frequent words. Due to the good performance on rare words, which can benefit the further incremental training for new corpus, hierarchical softmax model is widely accepted in incremental learning. Peng et al. (2017) proposed an incremental training model based on hierarchical softmax for new corpus [12], and a new Huffman tree was constructed with exiting Huffman tree changed to update all the word vectors and parameters. Stochastic gradient based optimization is performed respectively through the new and old Huffman tree.

In this paper, a more practical method for online word vector model generation based on incremental learning is proposed. While maintaining the inherited word Huffman tree in existing word vector model unchanged, a new Huffman tree of incoming words is constructed and there is no leaf node identical to the inherited Huffman tree. Then an updated Huffman tree is constructed by a method of node merging. In the node merging method, find the shortest path leaf node of the inherited Huffman tree and use it as the root of the new Huffman tree. Thus based on the existing word vector model, each word still has a unique encoding for the calculation of the hierarchical softmax model [13]. Based on the hierarchical softmax model, distributed vectors would be generated using Continuous Bag-of-Words (CBOW) [14] or Skip-gram methods [15] as fully retrained ones. Under this method, the previously trained model is maximally utilized. For online learning, the Huffman code of all words in the original corpus is retained with Huffman tree unchanged, and only the new corpus data is dynamically updated.

The incremental Huffman tree merging method can vectorize all words (including new words) so that the words can quantitatively measure the relationship and explore the relationship between words [16]. The experimental results show that the method could realize the word vector model generation online based on incremental learning simply and effectively.

2 INCREMENTAL HUFFMAN TREE MERGING

As mentioned above, hierarchical softmax model would be selected in incremental learning for online processing tasks [17]. Huffman tree is constructed according to the frequency of words and the occurrence of words in the corpus. The default left side (coded as 0) is a

negative class, and the right side (coded as 1) is a positive class. The leaf nodes of the Huffman tree are all in the corpus. The higher the word frequency, the closer the distance from the root node.

Take the word w_2 in Fig. 1 as an example. The input word is defined as w, and input layer word vector into the Huffman tree root node is x_w, from the root node to the leaf node where w is located as shown by bold nodes, the total number of nodes included is l_w, word w in Huffman tree, starting from the root node, the i^{th} node passing through is represented as p_i^w, and the corresponding Huffman code is $d_i^w \in 0,1$, where $i = 2, 3, \ldots, l_w$. The hidden layer neuron parameters corresponding to these non-leaf nodes are expressed as θ_i^w, where $i = 2, 3, \ldots, l_{w-1}$. No $i = l_w$ because the model parameters are only for the internal nodes of the Huffman tree

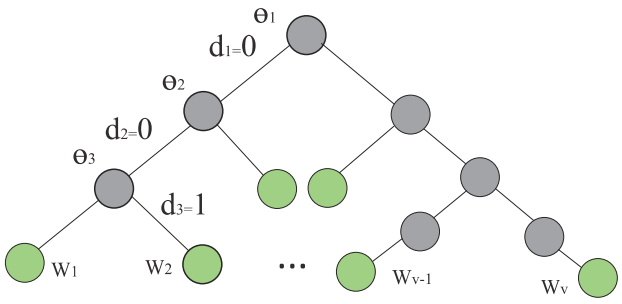


Figure 1 Hierarchical softmax model

2.1 Gradient Optimization

The hierarchical softmax model uses a binary tree to represent all words in the vocabulary. Probability is defined as the probability of a random walk starting from the root ending at the leaf node. At each inner node, it is necessary to assign the probabilities of going left and going right. When $d_i^w = 0$, left path is defined as positive, and when $d_i^w = 1$, right path is defined as negative. Apparently the probability of going right or right at node p_i^w is

$$P\left(d_j^w \mid x_w, \theta_{j-1}^w\right) = \begin{cases} \sigma\left(x_w^T \theta_{j-1}^w\right), & d_j^w = 0 \\ 1 - \sigma\left(x_w^T \theta_{j-1}^w\right), & d_j^w = 1 \end{cases} \tag{1}$$

$$\sigma\left(x_w^T \theta_{j-1}^w\right) = \frac{1}{1 + e^{-x_w^T \theta_{j-1}^w}} \tag{2}$$

Then for a target output word w, its maximum likelihood is

$$\begin{aligned} &\prod_{j=2}^{l_w} P\left(d_j^w \mid x_w, \theta_{j-1}^w\right) = \\ &= \prod_{j=2}^{l_w} \sigma\left(x_w^T \theta_{j-1}^w\right)^{1-d_j^w} \left[1 - \sigma\left(x_w^T \theta_{j-1}^w\right)\right]^{d_j^w} \end{aligned} \tag{3}$$

For a target output word w, the log-likelihood function is

$$\begin{aligned} E &= \log \prod_{j=2}^{l_w} P\left(d_j^w \mid x_w, \theta_{j-1}^w\right) = \\ &= \prod_{j=2}^{l_w} \left[\left(1 - d_j^w\right) \log \sigma\left(x_w^T \theta_{j-1}^w\right) + d_j^w \log\left(1 - \sigma\left(x_w^T \theta_{j-1}^w\right)\right)\right] \end{aligned} \tag{4}$$

Then take the derivative of E with regard to the vector representation of the inner node θ_{j-1}^w, obtaining

$$\begin{aligned} \frac{\partial E}{\partial \theta_{j-1}^w} &= \left(1 - d_j^w\right)\left(1 - \sigma\left(x_w^T \theta_{j-1}^w\right)\right) x_w - d_j^w \sigma\left(x_w^T \theta_{j-1}^w\right) x_w \\ &= \left(1 - d_j^w - \sigma\left(x_w^T \theta_{j-1}^w\right)\right) x_w \end{aligned} \tag{5}$$

In the same way, the gradient expression of x_w is as follows:

$$\frac{\partial E}{\partial x_w} = \left(1 - d_j^w - \sigma\left(x_w^T \theta_{j-1}^w\right)\right) \theta_{j-1}^w \tag{6}$$

2.2 Incremental Huffman Tree Merging Algorithm

The Huffman tree is constructed based on the word frequency. Once the corpus sample increases, the Huffman tree will change accordingly. The difficulty of word2vec online learning is how to create an incremental Huffman tree to obtain a unique Huffman code for each word. In the incremental learning method based on full updated Huffman tree, there could be almost half of the leaves changed from one side of the tree to the other side, corresponding to completely reversing the order of frequencies. It means that the coding of certain words has been changed, which will affect the subsequent network learning results.

Imagine that it is so time consuming and resource intensive to train the incremental learning model for massive data. This paper proposes a method for incremental learning based on tree merging algorithm. This algorithm is suitable for small sample online incremental learning for massive data, without the need for a large amount of computing resources to retrain the entire sample.

Based on this algorithm with in-memory database, the production system can realize online real-time application in the field of text processing.

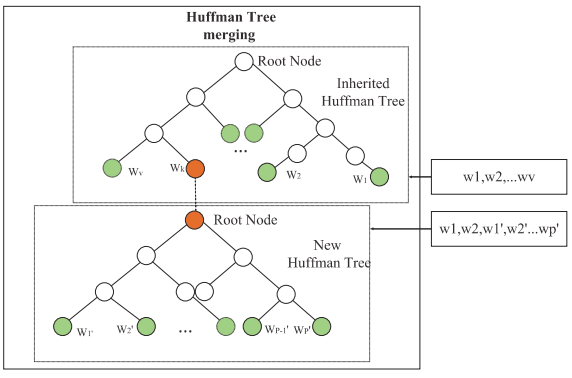


Figure 2 Huffman tree merging schematic diagram

The Huffman Tree merging schematic diagram is shown as follows in Fig. 2.

There are two corpuses, respectively corpus1 = (w_1, w_2, …, w_v) and corpus2 = $\left(w_1, w_2, w_1^{'}, w_2^{'}, \ldots, w_p^{'}\right)$. Green leaf nodes represent word nodes, and the dotted node represents the omission of subsequent connections. The merging algorithm steps are as follows.

Algorithm 1 Huffman Tree merging algorithm

1: keep exiting Huffman tree unchanged, nodes defined as $last\_neurons = p_i^{w1}, i \in (0, len(nodes))$
2: new corpus nodes defined as $new\_neurons = p_i^{w2}, i \in (0, len(newnodes))$
3: **if** $new\_neurons.word$ not in $last_{neurons}.word$**then**construct new Huffman tree
4: **end if**
5: find the shortest path node in inherited Huffman tree, thatis p_k^{w1}
6: $p_k^{w1} \leftarrow p_{root}^{w2}$
7:$p_{root}^{w2} \leftarrow p_k^{w1}$
8: Huffman tree updated

Keep existing Huffman tree unchanged as inherited tree, and find the shortest path leaf node of the inherited Huffman tree as the root of the new Huffman tree. Finally the updated Huffman tree has the inherited unchanged word node and the new corpus word node to be trained. For example, in the inherited Huffman tree, word w_1 is encoded as 0001. When the new corpus also contains the word w_1, the new Huffman tree does not contain the word w_1 node, so the word w_1 is still encoded as 0001 as original, avoiding re-learning all corpora. Then the shortest path node p_k^{w1} in the inherited Huffman tree is found, which the root node of new Huffman tree would be merged into. Thus based on the existing word vector model, each word still has a unique encoding for the calculation of the hierarchical softmax model.

Table 1 Code generation example using tree merging

Word	Frequency	Code
w_1	20	0001
w_2	18	0000
w_3	12	01001
w_4	10	01000
$w_1^{'}$	4	0000111
$w_2^{'}$	3	0000110
$w_3^{'}$	1	00001000

Through the tree merging method, the old tree model can be maximized, saving processing time to fully build a new tree. The core of the algorithm is to ensure that the original Huffman tree is unchanged. In order to reduce the calculation of the subsequent neural network layer, the shortest path node of the inherited Huffman tree is set as the root of the new tree. Finally, find the appropriate word vector for all nodes and all internal nodes θ to maximize the likelihood of training samples.

3 WORD2VEC BASED ON INCREMENTAL LEARNING

After the corpus is collected online, the updated Huffman tree is obtained. Based on the updated Huffman tree, word vector would be generated using traditional model, including the original continuous CBOW and Skip-gram models shown as Fig. 3.

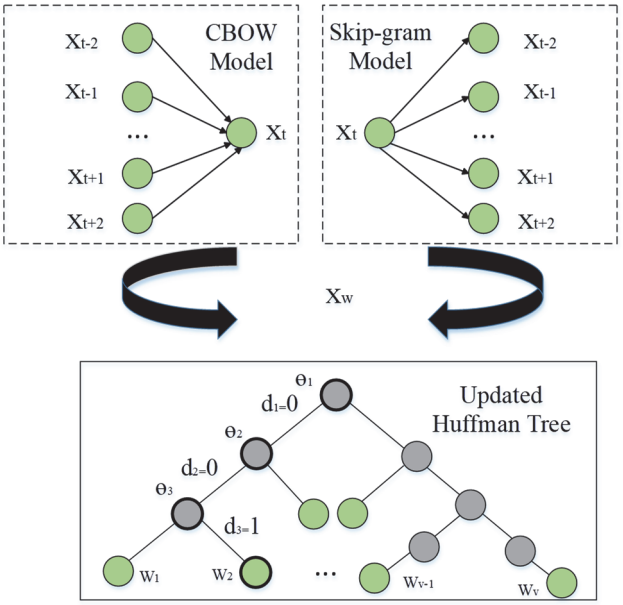


Figure 3 Huffman tree merging schematic diagram

Specific algorithm implementation details can be seen in the introduction of the word2vec principle. Here just generally talk about the difference paying particular attention to the different features in use.

3.1 CBOW Model

In the BBOW method, the central word is predicted by surrounding words, and the gradientascent algorithm is used to predict the surrounding words by using the prediction result of the central word. When the training is completed, each word will be used as the central word, and the word vector of the surrounding words will be adjusted, thus obtaining the word vector of all the words in the whole text. The input layer value is the sum of 2c word vectors around word w and averaged:

$$x_w = \frac{1}{2c}\sum_{i=1}^{2c} x_i \tag{7}$$

Hidden layer neuron vector θ_{j-1}^w and x_w as Eqs. (8) and (9) shown is updated using the gradient ascent method based on Eqs. (5) and (6).

$$\theta_{j-1}^w = \theta_{j-1}^w + \eta\left(1 - d_j^w - \sigma\left(x_w^T \theta_{j-1}^w\right)\right) x_w \tag{8}$$

$$x_w = x_w + \eta \sum_{j=2}^{l_w}\left(1 - d_j^w - \sigma\left(x_w^T \theta_{j-1}^w\right)\right)\theta_{j-1}^w \tag{9}$$

η is the learning rate of the gradient ascending.

3.2 Skip-gram Model

Skip-gram uses a central word to predict the surrounding words. In the Skip-gram mode, gradient ascent is also used to continuously adjust the word vector of the central word according to the prediction results of the surrounding words. After all the texts are traversed, the word vector of all the words of the text is obtained. x_w is the word vector corresponding to the word w. Note that there are $2c$ word vectors around x_w here. Since the contexts are mutual, while expecting $P(x_i \mid x_w)$, $i = 1, 2, \ldots, 2c$ to be maximized, also expecting $P(x_w \mid x_i)$, $i = 1, 2, \ldots, 2c$ to be the largest. Thus in an iterative window, not only a word x_w is updated, but also x_i, i = 1, 2, …, $2c$ total $2c$ words. This way the overall iteration will be more balanced. The specific learning method is the same as Eqs. (8) and (9), and no more details here.

In Skip-gram, each word will be influenced by the surrounding words. When each word is used as the central word, it will be predicted and adjusted $2c$-windows times. Therefore, when the amount of data is small, or the number of occurrences of rare words is small, this multiple adjustment will make the word vector relatively more accurate.

4 THE ANALYSIS OF EXPERIMENTAL RESULTS

To verify the efficiency and effectiveness of incremental training for online word vector generation, experimental evaluation has been performed on real data. Model performance includes many aspects [18], and the experiment in this paper will be carried out in two aspects: the training time and the semantic similarity of the distributed vector. The experimental results are compared with regular word2vec techniques and existing incremental learning based on Huffman tree global updated.

4.1 Training Time and Efficiency

For the near real-time streaming news corpus learning tasks, the new update data size fluctuates little and has been around 10 kB during processing interval time. The collected news data is divided into several sets. 100 kB, 1 MB, 10 MB, 100 MB data is selected as initial training corpus that is the existing corpus in previous sections.

For traditional global training, the old and new corpora are combined as a whole, and run the original CBOW and Skip-gram training models. For the incremental training, based on the initial different size training corpus of the trained model, and run incremental learning algorithm to Non-global update the Huffman tree as well as the parameters and word vectors.

The comparison results can be seen in Fig. 4.The four curves represent global learning based on CBOW, global learning based on Skip-grams, incremental learning based on CBOW and incremental learning based on Skip-grams.

In global leaning mode, as the amount of training data increases, the global training time becomes longer and longer. Although adding 10 kB is relatively small compared to the original training size 100 kB ~ 100 MB, the amount of data involved in training is high. For small sample online learning, incremental mode can greatly reduce incremental training time. Moreover, it is found that Skip-gram is in an order of magnitude slower than CBOW. By comparing CBOW and Skip-gram, it can be seen that for each context word, Skip-gram needs to update the parameters following the path from root to that word. Furthermore, incremental training for both CBOW and Skip-gram benefits from the algorithm and is faster than global training mode.

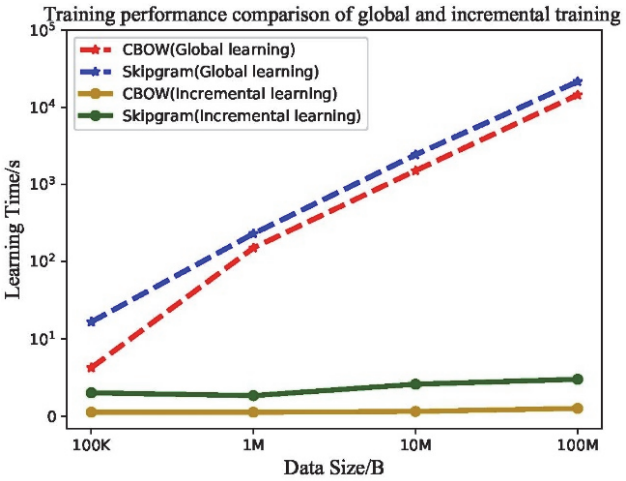


Figure 4 Training time

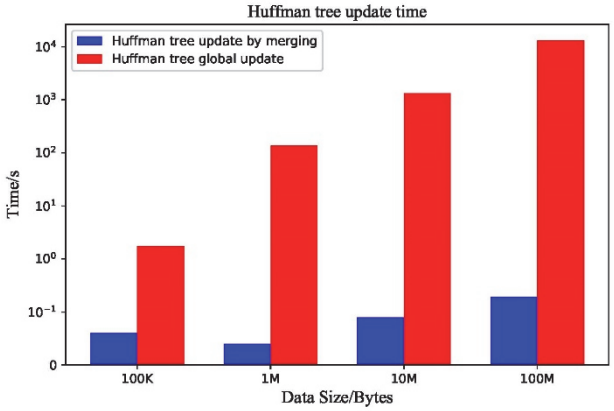


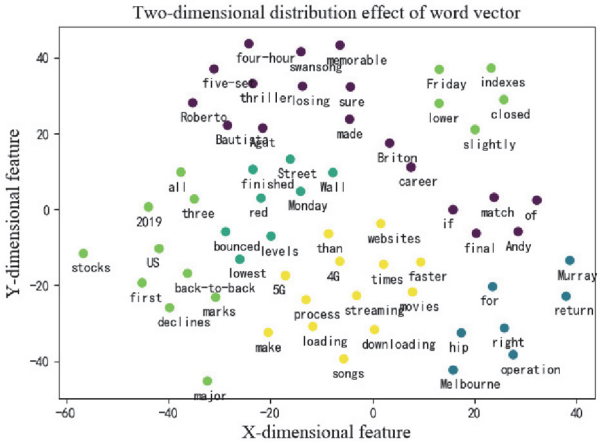
Figure 5 Training time for Huffman tree update

There are two ways to update the Huffman tree in incremental mode training. One is based on global pattern rebuilt. The other is the Huffman tree online merging method proposed in this paper. Fig. 5 shows the time comparison of two Huffman tree update modes. The tree merging method can greatly reduce Huffman tree coding time. It is worth mentioning that Huffman tree generation time at 1 MB data is less than the time at 100 kB data, because the new incoming 10 kB data and 1 MB data are more repeated, the number of update nodes is less.

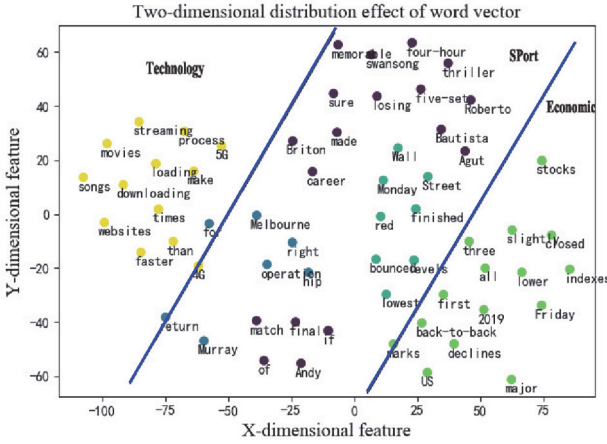
4.2 Word Similarity

The most important purpose of the word2vec model is to generate word vectors that map words to High-dimensional vector spaces [19, 20]. Vector operations between words can also correspond to semantics. Next the word similarity evaluation benchmarks are used to evaluate the correctness of the proposed incremental training algorithm, as shown in Fig. 6.

Both modes are capable of extracting word vector features. In incremental learning mode based on Huffman tree global update, the word vector for all words is generated based on the reconstructed Huffman tree code. Word similarity will change with new incremental corpus. In incremental learning mode based on Huffman tree merging update, the feature distribution of the word vector shows a better effect, and can be better used for advanced applications such as text classification and emotion recognition.



(a) Incremental learning based on Huffman tree global update



(b) Incremental learning based on Huffman tree merging update

Figure 6 Word Similarity

5 CONCLUSION

In the text processing, small sample incremental learning is an important application scenario. For large data volume and high real-time performance requirements, an online word vector model generation method based on incremental Huffman tree merging is proposed.

Maintaining the inherited word Huffman tree in existing word vector model unchanged, a new Huffman tree of incoming words is constructed there is no leaf node identical to the inherited Huffman tree. Then an updated Huffman tree is constructed by a method of node merging. In the node merging method, find the shortest path leaf node of the inherited Huffman tree and use it as the root of the new Huffman tree. Thus based on the existing word vector model, each word still has a unique encoding for the calculation of the hierarchical softmax model. Finally, the generation of incremental word vector model is realized by using neural network on the basis of hierarchical softmax model. The experimental results show that the method could realize the word vector model generation online based on incremental learning with faster time and better performance.

Acknowledgements

This paper is supported by Nanjing Institute of Technology High-level Scientific Research Foundation for the Introduction of Talent (No. YKJ201918), the Natural Science Fondation of the Jiangsu Higher Education Institutions of China (No. 20KJB510049), and partially supported by the National Natural Science Foundation of China (No. 51675259) and Natural Science Foundation Youth Fund of Jiangsu Province of China (No. BK20181018).

6 REFERENCES

[1] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *Proceedings of the International Conference on Learning Representations.*

[2] Church, K. W. (2017). Word2Vec. *Natural Language Engineering, 23*(1), 155-162. https://doi.org/10.1017/S1351324916000334.

[3] Rexha, A., Kröll, M., Dragoni, M., & Kern, R. (2016). Polarity classification for target phrases in tweets:a word2Vec approach. *International Semantic Web Conference*, 217-223. https://doi.org/10.1007/978-3-319-47602-5_40.

[4] Zhang, D., Hua, X., Su, Z., & Xu,Y. (2015). Chinese comments sentiment classification based on word2vec and SVMperf. *Expert Systems with Applications, 42*(4), 1857-1863. https://doi.org/10.1016/j.eswa.2014.09.011.

[5] Chiu, J. P. C. & Nichols, E. (2016). Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics, 4*, 357-370. https://doi.org/10.1162/tacl_a_00104.

[6] Wang, A. & Gao, X. (2019). Multifunctional Product Marketing Using Social Media Based on the Variable-Scale Clustering. *Tehnički vjesnik, 26*(1), 193-200. https://doi.org/10.17559/TV-20181120082714.

[7] Fauzi, M. A. (2018). Word2Vec model for sentiment analysis of product reviews in Indonesian language. *International Journal of Electrical and Computer Engineering, 7*(1), 244-252. https://doi.org/10.11591/ijece.v7i1.pp244-252.

[8] Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532-1543. https://doi.org/10.3115/v1/D14-1162.

[9] Ji, S., Satish, N., Li, S., & Dubey, P. (2019). Parallelizing word2vec in shared and distributed memory. *IEEE Transactions on Parallel and Distributed Systems, 30(9): 2090-2100.* https://doi.org/10.1109/TPDS.2019.2904058.

[10] Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., & Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 3111-3119.

[11] Gutmann, M. U. & Hyvärinen, A. (2012). Noise contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *Journal of Machine Learning Research, 13*(Feb), 307-361. https://doi.org/10.1109/TNNLS.2011.2178124.

[12] Peng, H., Li, J., Song, Y., & Liu, Y. (2017, February). Incrementally learning the hierarchical softmax function for

neural language models. *Thirty-First AAAI Conference on Artificial Intelligence,* 3267-3273.
[13] Medhat, W., Hassan, A., & Korashy, H. (2014). Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journa*l*, 5*(4), 1093-1113. https://doi.org/10.1016/j.asej.2014.04.011.
[14] Liu, B. (2018). Text sentiment analysis based on CBOW model and deep learning in big data environment. *Journal of Ambient Intelligence and Humanized Computing*, 1-8. https://doi.org/10.1007/s12652-018-1095-6.
[15] Liu, P., Qiu, X., & Huang, X. (2015, June). Learning context-sensitive word embeddings with neural tensor skip-gram model. *In Twenty-Fourth International Joint Conference on Artificial Intelligence*.
[16] Wang, X., Du, Y., Li, X., Cao, F., & Su, C. (2019, February). Embedded representation of relation words with visual supervision. *2019 Third IEEE International Conference on Robotic Computing (IRC)*, 409-412. https://doi.org/10.1109/IRC.2019.00075
[17] Yogatama, D., Faruqui, M., Dyer, C., & Smith, N. (2015, June). Learning word representations with hierarchical sparse coding. *In International Conference on Machine Learning*, 87-96.
[18] Burgan, H. I. & Aksoy, H. (2018). Annual flow duration curve model for ungauged basins. *Hydrology Research, 49(5), 1684-1695.* https://doi.org/10.2166/nh.2018.109.
[19] Martinčić-Ipšić, S. & Miličić, T. (2019). The influence of feature representation of text on the performance of document classification. *Applied Sciences, 9*(4), 743. https://doi.org/10.3390/app9040743.
[20] Kiela, D., Hill, F., & Clark, S. (2015). Specializing word embeddings for similarity or relatedness. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2044-2048. https://doi.org/10.18653/v1/D15-1242

Contact information:

Kui QIAN, Senior engineer
(Corresponding author)
School of Automation,
Nanjing Institute of Technology,
No.1 Hongjing Avenue, Jiangning District, Nanjing, Jiangsu Province, China
E-mail: kuiqian@njit.edu.cn

Lei TIAN, Assistant Professor
School of Automation, Nanjing Institute of Technology,
No.1 Hongjing Avenue, Jiangning District, Nanjing, Jiangsu Province, China
E-mail: tianleinjit@163.com

Xiulan WEN, Professor
School of Automation, Nanjing Institute of Technology,
No.1 Hongjing Avenue, Jiangning District, Nanjing, Jiangsu Province, China
E-mail: zdhxwxl@njit.edu.cn

Zhenzhong SONG, Senior engineer
Shanghai Electromechanical Engineering Institute,
No. 3888 Yuanjiang Road, Minhang District, Shanghai, China
E-mail: szzjason@126.com