SG-Drop: Faster Skip-Gram by Dropping Context Words[1]

DongJae Kim, DoangJoo Synn, Jong-Kook Kim
Dept. of Electric and Electronics Engineering, Korea University
dong4810@korea.ac.kr, alansynn@korea.ac.kr, jongkook@korea.ac.kr

ABSTRACT

Many natural language processing (NLP) models utilize pre-trained word embeddings to leverage latent information. One of the most successful word embedding model is the Skip-gram (SG). In this paper, we propose a Skip-gram drop (SG-Drop) model, which is a variation of the SG model. The SG-Drop model is designed to reduce training time efficiently. Furthermore, the SG-Drop allows controlling training time with its hyperparameter. It could train word embedding faster than reducing training epochs while better preserving the quality.

1. INTRODUCTION

Language Model (LM) has a long history in natural language processing and is based on various NLP tasks such as Machine Learning Translation (MLT) tasks [1]. The earlier LM was introduced with statistical LM to cover various linguistic representations by assigning probabilities to a partial word sequence. Nevertheless, the statistical LM has a significant limitation since it needs to compute sequences of conditional probabilities. There were several approaches to solve this problem, such as the N-gram model. They were based on the approximation and still had a fundamental problem, such as dimensionality.

The distributional hypothesis [15], which is the essential idea of distributed representation for words, provides a fundamental theory to integrate semantics into word representations. It describes that the words that occur in the same contexts tend to have similar meanings. The Neural Network Language Model (NNLM), introduced [2][3], enables language modeling in continuous space to avoid the curse of dimensionality based on the distributional hypothesis. The key concept of NNLM [2] was learning a distributed representation for words, which converts the word sequence into a low dimensional vector, the word embeddings. The distributed word representation models perform better than the former approaches, such as N-gram LM, and emerged as an empirical topic.[14]

In 2013, Word2vec [5] [6] proposed two popular word representation models: Continuous Bag-Of-Words (CBOW) and Skip-gram (SG). SG and CBOW share the same fundamental idea that the words similar to each other are likely to have similar co-occurrence of neighbor words. The CBOW model learns the relationship of context and words by predicting the center word with the nearby words, and the SG does vice versa. These two popular word representations do not have non-linear hidden layers. Therefore, the models could process the large datasets much faster than the former NNLM based models. Many studies have shown by adapting word embeddings [4], semantics between center word and context words can be modeled and captured. As a result, the word embeddings have become the mainstream methods of distribution representations and widely used on NLP tasks such as text classification, inference, and knowledge mining.

Though CBOW generates the 1-dimensional average vector from the nearby word vectors, SG predicts the context words given the center words. This main difference makes SG to have more prediction pairs than CBOW. There have been various approaches to improve the word embedding quality from the promising enhancements of word representations. Pennington et al. [7] focused on the global statistical information. Ling et al. [8] suggested improving word embedding leveraging position information. Faruqui et al. [9] introduced a post-processing technique that utilizes semantic lexicons.

There are many corpus types from various sources such as medical corpus, blog corpus, social network corpus. Though multiple corpora can be mixed to learn common semantics, they can be trained independently to learn their unique characteristics. Furthermore, in the era of the internet, the size of the unlabeled text is rapidly growing. If we could train the word embedding model faster, it will be easier to handle those works.

In this work, we describe a simple modification to Word2Vec's Skip-gram model that improves learning speed. Training a single pair of a center word and its nearby word with negative sampling requires multiple vector scalar products and memory accesses. The SG-Drop model aims to drop

[1] This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2016R1D1A1B04933156) This work was supported in part by the National Research Foundation of Korea (NRF) through the Basic Science Research Program funded by the Ministry of Education under Grant 2014R1A1A2059527, and in part by the Information Technology Research Center (ITRC), Ministry of Science and ICT (MSIT), South Korea, through a Support Program under Grant IITP-2020-2018-0-01433, supervised by the Institute for Information and Communications Technology Promotion (IITP)

some context words during the training process. Our goal is to make the original model keeps its simplicity with the higher speed and low memory access to work properly on low-performance hardware. We expect the SG-Drop model to run efficiently on the commodity hardware and be used in various fields such as on-device learning and online sentiment classification [10].

2. RELATED WORK

2.1 Skip-gram

The Skip-gram model was introduced in popular Word2vec [5] [6]. The SG model learns word embeddings by leveraging the relation between a word and its adjacent words. The SG's objective function is to maximize the average log probability given vocabulary size $|V|$:

$$\mathcal{L}_{SG} = \frac{1}{|V|}\sum_{t=1}^{|V|}\sum_{0<|i|\le c} \log\, p(w_{t+i} \mid w_t)$$

where w_t and w_{t+1} denote a center word and its context word. Though the probability $p(w_{t+i} \mid w_t)$ can be estimated with softmax function, it requires more computation as the vocabulary size increases. Hence, the negative sampling objective is preferred for large $|V|$. Given N as negative samples, negative sampling objective is defined:

$$\log\left(\sigma\left(v'^{T}_{w_{t+i}} v_{w_t}\right)\right) + \sum_{w_n \in N} \log\left(\sigma\left(v'^{T}_{w_n} v_{w_t}\right)\right)$$

where v and v' refer to the input and output vector representation of the corresponding word and σ denotes sigmoid function.

2.2 Memory Efficient Approaches on NLP Tasks

Although deep learning models have shown good accuracy, deploying the models in production and execute on-line learning poses significant memory constraints. For NLP tasks, the typical word embedding vector holds 60M parameters on the embedding matrix, which would cost up to 100Gbs of memory [17]. Shu and Nakayama [17] have shown that their simple sentiment analysis model costs up to 98.8% of its parameters on the entire network's embedding vectors.

The former researches [17][18][19] have focused on compressing the word embedding matrix by hashing or quantization-based approaches. These approaches make the additional process and give additional latency and need to be tuned. Acharya [16] proposed a low-rank projection of the embedding layer using Singular Value Decomposition (SVD). Since the Acharya has shown promising results on sentiment analysis on specific tasks, the low-rank matrix factorization (LMF) reduces the dimensionality and obtains the high compression rate. However, dimensionality reduction means the loss of the embedding layer for certain original information on training.

3. SKIP-GRAM DROP

Reducing train epochs can easily shorten the training time. However, this naive approach is prone to performance

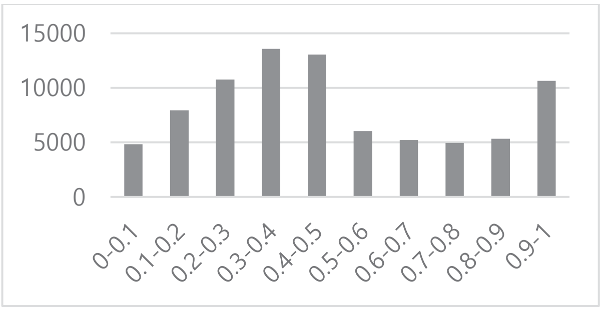

Figure 1. Histogram of prediction values in the SG model when the center word is "ice".

degradation due to the decreased number of training samples. The SG-Drop model suggests attaining faster training by dropping context words, which seem to be relatively less important. The experiment in Section 4.3 is conducted to compare our model to simply reducing epochs.

The SG-Drop model's overall architecture is the same as the SG except that it makes a drop decision for every pair of a center word and its nearby word. Whenever a drop decision is needed, the SG-Drop calculates a probability to drop a given pair with a drop probability function. The drop probability function is a function that returns a probability given a center word and its nearby word as input.

The drop probability function returns a lower probability for the context words, which seem to be more relevant. Since the SG-Drop model aims to improve training speed, the drop probability function should not be a compute-intensive function. If the decision making takes a long time, it could offset the learning time gain by dropping words.

The less critical words are nearby words that are less related to the meaning of the center word.

"I am so hungry that I could eat a horse."

In an example sentence above, the word "hungry" could have "I" and "eat" as context words when the context window size is five. However, they are not equally relevant to the meaning of the word "hungry". Dropping those less relevant words could make dropping effective. Though people can easily find those less critical words, a computer needs a proper measure.

Depending on the distributed hypothesis and the architecture of the SG, we can assume that relevant words are more predictable. Using $\sigma\left(v'^{T}_{w_{t+i}} v_{w_t}\right)$ term, which is part of the SG negative sampling loss, could easily estimate the relatedness. However, dropping less relevant words with low $\sigma\left(v'^{T}_{w_{t+i}} v_{w_t}\right)$ values is dangerous. Figure 1 shows the histogram of $\sigma\left(v'^{T}_{w_{t+i}} v_{w_t}\right)$ when training the SG model on the small corpus. The number of $\sigma\left(v'^{T}_{w_{t+i}} v_{w_t}\right)$ values lower than 0.5 is much larger than the counterpart.

We use a simple trick to find less related words. The SG-Drop model selects a random negative sample word w_n and regards context words with high $\sigma({v_{W_n}}^{T}\, {v'}_{W_{t+i}})$ as less relevant words. Hence, the drop probability function returns drop probability with the following equation:

$$MDP + (1 - MDP)\ \sigma({v_{W_t}}^T\ v'_{W_{t+i}})$$

where minimum drop probability (MDP) is a hyperparameter. The MDP is introduced to control training speed while without bias. Furthermore, this method is SG friendly because the SG selects random negative samples for every training step. Thus, we could reuse a negative sample to save computation resources.

The final decision is made with the calculated probability, and a randomly generated number ranged between zero and one with a uniform distribution. The purpose of the probabilistic drop is to avoid a bias that can be created by wrong relatedness evaluation. Also, to guarantee to learn from all possible pairs in the corpus at least once, we conduct the first epoch of training without context word drops. Since the SG-DROP model trains word embeddings without a drop on the first epoch, we can assume that relatedness is trained in word vectors through the co-occurrence in the corpus. Thus, we could effectively drop context words with word vectors trained during the first epoch.

4. EXPERIMENT

4.1 Experiment Setup

For model evaluation, we created a large corpus and a small corpus for training. We created a large corpus with the following procedures.

1. The plain text was extracted from the latest Wikipedia dump to generate the large corpus.
2. The plain text was processed to remove Wikipedia tags and to break into sentences.
3. Sentences with less than ten words were filtered to remove too broken sentences during the Wikipedia tag removal process.

The small corpus was created by a 1% random sampling of the sentences in the large corpus. We conducted all the training process on Ubuntu 20.04 installed system with 32 core AMD Threadripper 3970X processor.

Hyperparameter	Value
Dimension	200
Widow Size	5
Negative Samples	5
Epoch	5
Learning Rate	0.025

Table 1. Commonly used hyperparameters

4.2 Baseline and Dataset

As a baseline model, the SG model [5] [6] was used. Table 1 describes commonly used hyperparameter values for the SG and the SG-Drop model. To evaluate trained word embeddings, we performed word similarity evaluation tasks. Following word similarity datasets were used in our experiments: Simlex-999[11], Wordsim-353[12], MEN-3000[13].

	Simlex	Wordsim	MEN	Time(s)
$SG_{Epoch=5}$	33.35	65.95	63.28	98.1
$SGD_{MDP=0}$	32.68	65.40	62.29	81.3(17.13%)
$SG_{Epoch=4}$	32.21	63.66	60.76	76.9(21.61%)
$SGD_{MDP=0.25}$	31.73	64.93	60.16	67.5(31.19%)
$SG_{Epoch=3}$	31.18	61.64	57.09	58.4(40.47%)
$SGD_{MDP=0.5}$	30.81	62.33	57.77	52.6(46.38%)
$SG_{Epoch=2}$	29.03	56.75	51.37	39.5(59.73%)
$SGD_{MDP=0.75}$	29.67	59.64	54.29	37.4(61.88%)
$SG_{Epoch=1}$	24.48	47.6	42.74	20.8(78.8%)

Table 2. Result of the word similarity evaluation on the small corpus. Percentage in time column denotes relative speedup.

4.3 Epoch and Skip-gram Drop

The table 2 shows the result of word similarity evaluation tasks. The SGD in the table is the abbreviation of the SG-Drop model. The SG model, trained with five epochs, shows the best result because it does not sacrifice training samples. The SG-Drop model trained with MDP=0 shows minor performance reduction with about 17% speedup. The SGD models with MDP = 0.5 and 0.75 have faster training time with better overall performance than the SG models trained with epoch = 3 and 2. Thus, the SG-Drop model is a more efficient way to reduce training time than reducing the training epoch. Also, The SGD models tend to show less performance degradation on the Wordsim dataset.

	Simlex	Wordsim	MEN	Time(s)
$SG_{win=10}$	33.33	67.62	67.05	165.2
$SGD_{win=10}$	32.77	67.65	66.17	135.5(17.98%)
$SG_{win=15}$	32.62	68.81	68.79	223.3
$SGD_{win=15}$	32.54	68.49	68.22	182.3(18.32%)

Table 3. Result of evaluation on different window sizes.

4.3 Window Size

In this experiment, we trained the SG and the SGD model with window size 10 and 15. The table 3 illustrates the result. If the window size increases, the number of less relevant words in the context window is likely to increase. As a result, the overall performance loss is reduced with increased speedup. It is observed that the SGD model works better with various context window size.

	Simlex	Wordsim	MEN	Time(hour)
SG	36.88	71.65	74.81	3.05
$SGD_{MDP=0}$	36.87	71.29	74.79	2.48(18.69%)
$SGD_{MDP=0.75}$	36.26	70.33	74.28	1.12(63.28%)

Table 4. Result of word similarity evaluation on the large corpus.

4.5 Corpus Size

Table 4 shows the similarity evaluation tasks on the large corpus. In a large corpus, the SG-Drop with MDP = 0 showed merely no performance reduction except the Wordsim dataset.

We also trained the SG-Drop with MDP=0.75 to further explore its efficiency on the large corpus. As the result shows, the SG-Drop model suffers a smaller decline than on the corpus composed of fewer words.

5. CONCLUSION

In this paper, we presented the simple augmentation, which boosts training speed with reasonable performance reduction. Besides, the SG-Drop allows users to control speedup with the hyperparameter MDP. Experimental data showed that the SG-Drop model efficiently reduces training time than reducing training epochs and performing well with various environments. We expect our model to be applied to services where fast learning time is required or NLP research on low computing power devices such as mobile devices.

Furthermore, our model could be more mobile device friendly. While the SG-Drop model provides faster training experience, it still needs a parameter size proportional to vocabulary size and dimension. As a result, mobile devices still have difficulty locating the embedding matrix on its memory. However, applying quantization or LMF methods described in section 2.1 could solve this issue. Our future work targets to our model more mobile compatible on mobile devices.

REFERENCES

[1] Kun Jing and Jungang Xu. "A Survey on Neural Network Language Models" arXiv preprint arXiv:1906.03591, 2019.

[2] Yoshua Bengio, Re´jean Ducharme, Pascal Vincent, and Christian Janvin. "A neural probabilistic language model" J. Mach. Learn. Res., 3:1137–1155. 2003.

[3] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: Deep neural networks with multitask learning" In Proceedings of the 25th International Conference on Machine Learning, ICML '08, pages 160–167, New York, NY, USA. ACM, 2008.

[4] Wang, S., Zhou, W. & Jiang, C. A survey of word embeddings based on deep learning. Computing 102, 717–740, 2020

[5] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space", Proceedings of the International Conference on Learning Representations (ICLR 2013), Scottsdale, 2013,

[6] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality" Advances in Neural Information Processing Systems 26 (NIPS 2013), Lake Tahoe, 2013, pp.3111-3119

[7] Pennington J, Socher R, Manning CD, "Glove: global vectors for word representation.", In Empirical methods in natural language processing (EMNLP), pp 1532–1543, 2014

[8] Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of Word2Vec for syntax problems. In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1299– 1304, Denver, Colorado. Association for Computational Linguistics.

[9] Manual Faruqui, Jesse Dodge, Sujay Kumar Jauhar, Chris Dyer, Eduard Hovy, and Noah A. Smith, "Retrofitting word vectors to semantic lexicons." In Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1606–1615, 2015

[10] Barkha Bansal, Sangeet Srivastava, "Sentiment classification of online consumer reviews using word vector representations", Procedia Computer Science, Volume 132, 2018

[11] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin, "Placing search in context: The concept revisited", ACM Transaction on Information Systems, Volume 20, Issue 1, 2002, pp.116–131

[12] E. Bruni, N. K. Tran, and M. Baroni, "Multimodal distributional semantics" Journal of Artificial Intelligence Research, Volume 49, Issue 1, 2014, pp.1–47

[13] F. Hill, R. Reichart, and A. Korhonen, "Simlex-999: Evaluating semantic models with (genuine) similarity estimation", Computational Linguistics, Volume 41, Issue 4, 2015, pp.665–695

[14] Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, ACL '10. Association for Computational Linguistics.

[15] Harris, Z. (1954). Distributional structure. Word, 10(23): 146-162.

[16] Acharya, Anish & Goel, Rahul & Metallinou, Angeliki & Dhillon, Inderjit. (2019). Online Embedding Compression for Text Classification using Low-Rank Matrix Factorization.

[17] Shu, R., and Nakayama, H. 2017. Compressing word embeddings via deep compositional code learning. arXiv preprint arXiv:1711.01068.

[18] Joulin, A.; Grave, E.; Bojanowski, P.; Douze, M.; Jegou, H. and Mikolov, T. 2016. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651.

[19] Raunak, V. 2017. Effective dimensionality reduction for word embeddings. arXiv preprint arXiv:1708.03629.