A Fast Word2Vec implementation on manycore architectures for Text Representation and its applications

Leonardo A. Amorim, Mateus F. Freitas, Chayner C. Barros, Wellington S. Martins

[1] Instituto de Informática – Universidade Federal de Goiás (UFG)
Caixa Postal 131 – 74.001-970 – Goiânia – GO – Brazil

{leonardoafonso,mateusfreitas,chaynercordeiro,wellington}@inf.ufg.br

Abstract. *Word embedding has made possible to work with semantics in any application that works with a text document. Through algorithms that implement this technique, such as Word2Vec, it is possible to discover the similarity between words, paragraphs and even whole documents. However, generating word embedding still has a high computational cost. Some researches have been proposed parallel algorithms in recent years to deal with this problem, but gains in performance have ranged from 2 to 20 times compared to the original implementations. Manycore architectures have been able to scale algorithms in a more performative way. Since the accuracy of the word embeddings generation depends on a large amount of data (Big Data), it is necessary that new scalable parallel algorithms are developed to deal with this large amount of data (billions of words). Scalable parallel algorithms development is one the most complex and difficult tasks so, in this work, we focus on exploiting parallelism in text representation with applications. We work with Word2Vec for text representation.*

1. Introduction

The representation of words by vectors has become state-of-the-art in classification tasks. This is mainly due to the growth of Word2Vec that generates compact representations of words to express the semantic relationships through vector operations. The Word2Vec has been used in various fields of computing such as Sentiment Analysis, Word Clustering, Machine Translation, Named entity recognition and Automated Program Repair [Amorim and et al 2018]. The use of Word2Vec in Information Retrieval may be promising because it adds semantics in document classification and this can enhance the text classification task.

The major Word2Vec bottlenecks are inherent iterative nature of Stochastic Gradient Descent (SGD) algorithm and between the hidden layer and output layer (or Softmax layer). The time taken to generate the word embeddings makes the experiments in the areas cited severely affected by delays in the results. This delay can force the use of small vocabulary and, as a consequence, can impact the accuracy of the generation of word embeddings. There are parallel algorithms for the SGD in manycore architectures, but they do not have good strong scalability. The scalability of the parallel algorithm is important because every day we have hardware with more cores and taking advantage of this capability is a major challenge. In this context, there are two open questions that are going to guide this research:

- How to exploit parallelism in text representation in an efficient and scalable way?

- How to apply the parallel proposals in the context of Automated Program Repair application?

The work is organized as follows. Section 2 provides background to Word embedding model. Section 3 summarizes existing approaches to improving Word2Vec performance. Section 4 presents new ideas to explore parallelism in Word2Vec. Finally, in Section 5 we present our conclusions.

2. Background

Word embedding is the collective name for a set of language modeling and feature learning techniques in natural language processing where words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space relative to the vocabulary size ("continuous space"). Methods to generate this mapping include neural networks, dimensionality reduction on the word co-occurrence matrix [Levy and Goldberg 2014], probabilistic models, and explicit representation in terms of the context in which words appear. Word and phrase embeddings, when used as the underlying input representation, have been shown to boost the performance of NLP tasks such as syntactic parsing, sentiment analysis, automatic document classification. Current methods for generating word embeddings fall into two categories: count-based methods [Blei et al. 2003] (e.g. Latent Semantic Analysis), and predictive methods (e.g. neural probabilistic language models). In this work, we focus on Word2Vec, which is a predictive method with moderate computational costs that permits the processing of large datasets.

Word2Vec learns continuous word embeddings from plain text in an entirely unsupervised way. The model assumes the Distributional Hypothesis [Harris 1954], which states that words that appear in the same contexts tend to share semantic meaning. This allows us to estimate the probability of two words occurring close to each other. With Word2Vec, a neural network is trained with streams of n-grams of words so as to predict the n-th word, given words $[1, ..., n-1]$ or the other way around. The output is a matrix of word vectors or context vectors respectively. Since the neural network used has a simple linear hidden layer, the intensity of correlation between words is directly measured by the inner product between word embeddings. Such a shallow neural network might not produce as precise distributed representations as deep neural networks on relatively small datasets, but they can process data much more efficiently. When operating on large datasets, this approach usually produces better results than using a more sophisticated algorithm [Mikolov et al. 2013].

The two neural network models used by Word2Vec are known as continuous bag-of-words (CBOW) and Skip-gram. Rather than predicting a word conditioned on its predecessor, as in a traditional bi-gram language model, the CBOW model predicts the current word based on the context, while the Skip-gram model predicts the neighborhood words given the current word. The training of these models has to compute normalization terms, which has complexity $\mathcal{O}(|\mathcal{V}|)$, where $|\mathcal{V}|$ is the vocabulary size. To reduce this high computational cost, Word2Vec uses fast training algorithms: hierarchical softmax and negative sampling. Hierarchical softmax makes use of a Huffman tree representation of the vocabulary, which saves calculations, at the potential loss of some accuracy. On the other hand, negative sampling avoids using the words observed next to one another in

the training data as positive examples, and instead sample random words from the corpus and present to the network as negative examples. Word2Vec remains a popular choice for building word vectors due to their efficiency and simplicity. Although these algorithms are widely used, generating word embeddings are still too costly, which impacts negatively on the time for conducting experiments in both Information Retrieval and Machine Learning applications.

3. Related Works

The first state-of-the-art algorithms for Word embeddings including Word2vec have been parallelized for multi-core CPU architectures but are based on vector-vector operations that are memory-bandwidth intensive and do not efficiently use computational resources. The original Word2Vec was implemented this way.

To reduce generation time for word vectors, [Liu, 2014] implemented the optimization CBOW model in CUDA. He made a in-warp approach to CUDA architecture. A WARP works with one word and updates the hidden layer of the artificial neural network in shared memory. Each WARP of thread updates the weights for one word of the sentence. He uses one in-warp to handle one data (one word), and 32 threads manage the parallelism in one data in in-warp [Liu 2014].

In [Ji et al. 2016], it was shown that the reuse of various data structures in the algorithm through the use of minibatching, allows one to express the problem using matrix multiply operations producing good strong scalability. In [Taylor et al. 2016], it was explored an unconventional training method to train networks without gradient descent steps.

In [Gupta and Khare 2017], they present BlazingText, a high performance distributed word2vec implementation that leverages massive parallelism provided by modern GPUs. They exploit GPU parallelism to discover the right balance between throughput and accuracy. This work replicated the matrices and processed disjoint parts of sentences in each GPU. Their work uses one thread block per sentence. Since the corpus is too large to fit in the GPU memory, this work uses disk stream to the GPU and several sentences are batch transferred to decrease the cost of data transfer between CPU and GPU. And at regular intervals, their algorithm produces a combination of matrix weights (one coming from each GPU) and transmits the new weights (new matrices) for each GPU to continue its iteration. According to the authors, there was no significant fall in accuracy with multi-GPU implementation. Their proposed implementation achieves near-linear scalability across multiple GPUs.

In [Shim et al. 2017], they propose a fast approximation method of a softmax function with a very large vocabulary using singular value decomposition (SVD). SVD-softmax targets fast and accurate probability estimation of the topmost probable words during inference of neural network language models. The proposed method transforms the weight matrix used in the calculation of the output vector by using SVD. The approximate probability of each word can be estimated with only a small part of the weight matrix using a few large singular values and the corresponding elements for most of the words.

In [Simonton and Alaghband 2017], the authors propose new methods to increase the speed of the Word2Vec on multi-core shared memory CPU systems, and on modern

NVIDIA GPUs with CUDA. They perform this on multi-core CPUs by batching training operations to increase thread locality and to reduce accesses to shared memory. They propose new heterogeneous NVIDIA GPU CUDA implementations of both the skip gram hierarchical softmax and negative sampling techniques that utilize shared memory registers and in-warp shuffle operations for maximized performance.

Grave et al. propose an approximate strategy to train neural network based language models over vast vocabularies efficiently. Their approach, called adaptive softmax, avoids the linear dependency on the vocabulary size by exploiting the unbalanced word distribution to form clusters that explicitly minimize the expectation of computational complexity. Their approach further reduces the computational cost by employing the specificities of modern architectures and matrix-matrix vector operations[Grave et al. 2016].

The speedup value of the parallel state-of-art versions ranges from 2 to 20 times. Therefore, we will present in the next section some ideas that have the potential to improve the scalability of Word2Vec.

4. New ideas to explore parallelism in Word2Vec

To gain more performance and scalability of Word2Vec in many-core architectures, we speculate that the following experiments can accomplish promising results:

- Regarding implementation in Multi-GPU, we can develop new schemes to distribute the gradient for Artificial Neural Network used by Word2Vec. These schemes can be inspired by articles of Deep Learning in Distributed Systems [Dean et al. 2012];
- Search for a data structure that better exploits the features of manycore architectures (that allows exploring more parallelism i.e without leaving idle processors) in order to replace the Huffman tree in the Hierarchical Softmax;
- Group words that are most likely to co-occur in order to generate vectors for word classes [Shen et al. 2017]. As a consequence, the number of vectors that need to be updated will decrease. It is a similar effect when reducing matrices using SVD. This idea explores the use of K-Nearest Neighbors algorithm and uniform sampling to approximate the softmax function and achieve $\mathcal{O}(\sqrt{V})$ runtime[1];
- Explore matrix batch multiplication operations for CBOW Architecture in Multi-GPU.

5. Conclusion

The proposals of the related works can be summarized in reducing the weight matrices of the Word2Vec model; perform matrix multiplication operations rather than vector-matrix; perform operations on batch matrices in order to hide latency; to exploit low-level GPU resources (operations using in-warp) in order to increase scalability and reduce memory access. Although parallel implementations for Word2Vec or Softmax have had good performance gains, scalability can still be improved. Therefore, we expect to acquire novel and faster Word2Vec algorithm to generate word embeddings able to surpass current state of the art baselines.

[1]http://cs231n.stanford.edu/reports/2017/pdfs/130.pdf

References

Amorim, L. and et al (2018). A new word embedding approach to evaluate potential fixes for automated program repair. In *The International Joint Conference on Neural Networks*. IEEE 2018.

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022.

Dean, J., Corrado, G. S., Monga, R., Chen, K., Devin, M., Le, Q. V., Mao, M. Z., Ranzato, M., Senior, A., Tucker, P., Yang, K., and Ng, A. Y. (2012). Large scale distributed deep networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1223–1231, USA. Curran Associates Inc.

Grave, E., Joulin, A., Cissé, M., Grangier, D., and Jégou, H. (2016). Efficient softmax approximation for gpus. *CoRR*, abs/1609.04309.

Gupta, S. and Khare, V. (2017). Blazingtext: Scaling and accelerating word2vec using multiple gpus. In *Proceedings of the Machine Learning on HPC Environments*, MLHPC'17, pages 6:1–6:5, New York, NY, USA. ACM.

Harris, Z. (1954). Distributional structure. *Word*, 10(23):146–162.

Ji, S., Satish, N., Li, S., and Dubey, P. (2016). Parallelizing word2vec in shared and distributed memory.

Levy, O. and Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q., editors, *Advances in Neural Information Processing Systems 27*, pages 2177–2185. Curran Associates, Inc.

Liu, R. (2014). Unpublished work, retrieved from the author's github web page https://libraries.io/github/fengchenhpc on 2018-08-15.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

Shen, Y., Tan, S., Pal, C. J., and Courville, A. C. (2017). Self-organized hierarchical softmax. *CoRR*, abs/1707.08588.

Shim, K., Lee, M., Choi, I., Boo, Y., and Sung, W. (2017). Svd-softmax: Fast softmax approximation on large vocabulary neural networks. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R., editors, *Advances in Neural Information Processing Systems 30*, pages 5463–5473. Curran Associates, Inc.

Simonton, T. M. and Alaghband, G. (2017). Efficient and accurate word2vec implementations in gpu and shared-memory multicore architectures. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7.

Taylor, G., Burmeister, R., Xu, Z., Singh, B., Patel, A., and Goldstein, T. (2016). Training neural networks without gradients: A scalable admm approach.