Efficient Word Vector Representations for Boosted Accuracy in Machine Translation Tasks

A Research Survey

Brian Fakhoury
Boston University
fakhoury@bu.edu

Abstract

The representation of words as vectors is a critical technique in the modern recipes for natural language processing (NLP). In this survey paper, I aim to cover a niche portion of the research on compressed representations through modified learning techniques for these embeddings. We will see how these methods not only lead to smaller embedding and memory spaces, but also higher performance in various NLP tasks. I also cover neighboring approaches to optimizing word embeddings (in the post-processing class of techniques), and use the relative performance and complexity of the approaches in consideration for future work. I leave this survey as an opener into a further modified approach for learning embeddings that will notably perform better in machine translation tasks.

1 Introduction

Word vectors began to take off when Mikolov et al. published his work on Word2Vec [7]. Previously, embeddings have been used in neural language models to some success, but, after Word2Vec, their popularity shot up. Other methods followed Word2Vec, like GloVe and FastText [3], as the use of pretrained word vectors made advancements in the field easier.

Today, much relies on the quality of pretrained embeddings as it is almost impossible for a majority of projects to train their own general embeddings. On the other hand, it is possible to adapt pretrained embeddings to a particular task, but this technique does not work for edge intelligence and other lower-powered scenarios.

To adequately deal with increasing demand for high quality embeddings, a whole class of optimization techniques have popped up that aim to tackle the size and performance of word vectors.

We can group the class of techniques into a few broad distinctions. First is post-processing versus in-line techniques. Post-processing techniques [1][2] simply take the output of a pretrained language model, and apply some sort of algorithm (like PCA) to produce a new set of word vectors. An in-line technique [3][4][5][6] modifies the training process of the language model to produce better word embeddings in the first place.

Another broad distinction we can make is between virtual and physical quantization. When we refer to virtual quantization, we mean the process of forcing a set of values to be constrained to certain pre-determined quanta. For example, instead of allowing a continuous 32-bit floating-point range, we can force all those floating-point numbers to be either of -0.9, -0.8, -0.7, …, 0.8, 0.9. This virtual quantization forces lower precision without changing anything on the compute layer. Physical quantization then would be the literal constraint on the bit level of values. For example, instead of using 32-bit floating point values, we could use 8-bit floating point values.

In the rest of this paper, we will see how these different techniques come into play. We will then dive specifically into the realm of virtual quantization during the language model training process, and explore how the amazing effects of this simple change can have a lasting effect on the NLP field.

We will keep the overarching theme as a relation to language model performance for machine translation between 2 or more languages. This sub-field can benefit greatly from better quality, high integrity word representations.

2 The Overview

We'll briefly cover some naïve techniques following standard compression methods, and use these findings as a basis for modifying the learning process for embeddings.

2.1 PCA Hybrid Approach

In this first approach, Raunak et al. [1] uses a series of PCA transforms to produce a subspace of the original vectors. The algorithm is defined by him as such:

Algorithm 1 The Post-Processing Algorithm, PPA(X, D)

Input: Word Embedding Matrix X, Threshold Parameter D.

1. Subtract the Mean:
X = X - mean(X).
2. Compute the PCA Components:
u_i = PCA(X), where i = 1,2,...,d.
3. Eliminate the Top D Components: ∀ v in X:
$v = v - \sum_{i=1}^{D}(u_i^T \cdot v)u_i$

Output: Post-Processed Word Embedding Matrix X.

end

Algorithm 2 The Dimensionality Reduction Algorithm

Input: Word Embedding Matrix X, New Dimension N, Threshold Parameter D.

1. Apply the Post-Processing Algorithm:
X = PPA(X, D).
2. Transform X Using PCA:
X = PCA(X).
2. Apply the Post-Processing Algorithm:
X = PPA(X, D).

Output: Word Embedding Matrix of Reduced Dimension N: X.

end

By applying PCA as a post-processing technique, this study showed that word vectors do indeed share a large common mean, and that most of the variance of the large vectors can be preserved with only 8 dimensions.

The performance of this technique is quite excellent relative to how easy it is to implement and use. Scores were mostly equal to and greater than scores for the same word vectors before reduction. Thus, this technique should be highly valuable for the purpose of at least reducing memory requirements. It might be a useful endeavor to automatically pick the hyperparameters for the reduced dimension size and threshold of variance as this technique has no universally applicable method, and must instead be tuned for each pre-trained embeddings set.

It's important that we note this technique is agnostic to any semantic meaning, and simply views the vector space in terms of its variance. As we'll see further on, this is the main issue with post-processing techniques, and is what makes in-line techniques better.

2.2 More on reduced dimensions

Much of the PCA hybrid approach is extended from the work of Mu et al. [2], where they show that removing the dominant directions from the word vectors better captures meaning and invariance. We've already gone over this from Raunak et al., but I want to point out this paper for its findings on what the top dominant directions actually represent:

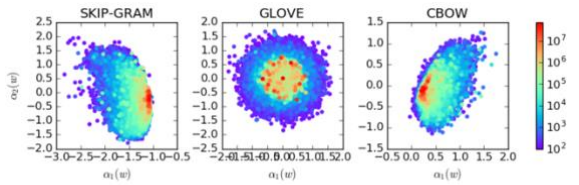

Figure 2: The top two PCA directions (i.e, $\alpha_1(w)$ and $\alpha_2(w)$) encode frequency.

Mu et al found that the top two directions in the PCA were encoding frequency of the words in the relative context. Furthermore, by removing these directions plus a few more that do not help in semantic meaning, they could fix a measure of "isotropy" which is a measure of a vector having the same value when measured in different directions. They formulated an approximate (2nd order) isotropy measure as:

$$I(\{v(w)\}) \approx \frac{|\mathcal{V}| + \min_{\|c\|=1} 1_{|\mathcal{V}|}^{\top} Vc + \min_{\|c\|=1} \frac{1}{2} c^{\top} V^{\top} Vc}{|\mathcal{V}| + \max_{\|c\|=1} 1_{|\mathcal{V}|}^{\top} Vc + \max_{\|c\|=1} \frac{1}{2} c^{\top} V^{\top} Vc} = \frac{|\mathcal{V}| - \|1_{|\mathcal{V}|}^{\top} V\| + \frac{1}{2}\sigma_{\min}^2}{|\mathcal{V}| + \|1_{|\mathcal{V}|}^{\top} V\| + \frac{1}{2}\sigma_{\max}^2}$$

By optimizing this value towards 1, they proved that the dimensionality reduction was effective in preserving information purely by relation, and not by frequency or other measures that a language model might learn from the unsupervised process.

2.3 Quantization in Neural Networks

Quantization has been successfully used in neural networks to decrease model size and improve learning outcomes [4].

Specifically, quantization refers to the process of limiting the values of real numbers such that the value can only take up some certain quanta. Already, Cai et al. have shown the impact that quantization can have on a deep neural network by limiting floating point values in the activation function to just 2 bits. By using quantization functions in a network, model size is greatly reduced, and learning performance can come out better with the right implementation. This is because quantization can act as a regularizer, as the derivatives of the activation functions will most likely need to clip values.

2.4 Limited Memory Embeddings

Ling et al. [5] showed that limiting the precision of weights can be applied to the training of embeddings. Specifically, their study shows that similar performance can be achieved by using 8 (or even 4) bits, instead of the standard 64, for the weight connections in the last dense layer of the network. This translates to word vectors that are an order of magnitude smaller without any change to the topology of the network. Here's the data on how this method performed on a dependency parsing task:

Bits	CBOW	Skipgram
Original	88.58%	88.15%
Binary	89.25%	88.41%
4-bits	87.56%	86.46%
6-bits	88.62%	87.98%
8-bits	88.63%	88.16%

At 4 bits, the memory usage is reduced by 16 times. These results were mimicked for a dozen of the popular baseline NLP tasks. This only leaves room for advanced MT tasks to be tested.

2.5 Advanced Quantization – Word2Bits

This is by far the most exciting step in quantized word vectors. In this study [6], Lam shows that 1 bit and 2 bit virtually quantized word vectors can actually replace the 32-bit counterparts. To start, many 32-bit word vectors are often 300+ dimensions, and take up many gigabytes of memory, which is not ideal for a pretrained embedding space. By combining recent efforts in quantized neural networks, Lam's study shows how memory usage can be reduced by up to 32x on the same tests.

First, a quantized loss function was used instead of the standard CBOW loss function from Word2Vec. This:

$$J(u_o, \hat{v}_c) = -log(\sigma(u_o^T \hat{v}_c)) - \sum_{i=1}^{k} log(\sigma(-u_i^T \hat{v}_c))$$

Turns into this:

$$J_{quantized}(u_o^{(q)}, \hat{v}_c^{(q)}) = -log(\sigma((u_o^{(q)})^T \hat{v}_c^{(q)})) - \sum_{i=1}^{k} log(\sigma((-u_i^{(q)})^T \hat{v}_c^{(q)}))$$

This loss function intuitively minimizes distance of similar words, and maximizes distance of non-similar words. To virtually quantize the values, two quantization functions were used. One quantization function used 1 bit, and the other 2 bits:

$$Q_1(x) = \begin{cases} \frac{1}{3} & x \geq 0 \\ -\frac{1}{3} & x < 0 \end{cases}$$

$$Q_2(x) = \begin{cases} \frac{3}{4} & x > \frac{1}{2} \\ \frac{1}{4} & 0 \leq x \leq \frac{1}{2} \\ -\frac{1}{4} & -\frac{1}{2} \leq x < 0 \\ -\frac{3}{4} & x < -\frac{1}{2} \end{cases}$$

Since these functions are not continuous, the identity function is used as the derivative, straight-through-estimator by Geoffrey Hinton:

$$\frac{\partial Q_{bitlevel}(x)}{\partial x} = I$$

The method then follows that full precision vectors are used in the network still, but the quantized output is what is being optimized. This approach maximized information flow through the network and targeted a reduced memory space for the final embedding.

The results on an advanced task set like DrQA SQuAD are impressive, while maintaining an 8x or more memory reduction, and achieving higher scores:

Word Vector Type	Bits per parameter	Dimension	Bytes per word	F1
Full Precision	32	200	800	75.25
	32	400	1600	75.28
	32	800	3200	75.31
	32	1000	4000	9.99
Quantized	1	800	100	76.64
	1	1000	125	76.84
	1	1200	150	76.50
	2	400	100	**77.04**
	2	800	200	76.12
	2	1000	250	75.66

3 Discussion

Compressing representations of language models serves more than to just reduce memory consumption. By quantizing a word vector into a few bits, the parameters of a network are regularized and are markedly a more succinct representation of the word, and its relationship to other words can be captured better.

I think there are two paths forward for this specific research. First, the Word2Bits approach could benefit from a PCA-based approach [1]. Second, regardless if the dimensionality reduction works, the Word2Bits should be tuned for machine translation tasks, as capturing invariance between two language models is critical.

In fact, dimensionality reduction might be more or less unimportant when we have powerful quantization techniques. Especially at the 1-bit quantization level, adding or removing dimensions has a far more reduced effect on the data.

This hypothesis can be easily tested by directly comparing quantization with post-processing dimensionality reduction against quantization on Word2Vec alone.

Furthermore, the concept of limiting the memory space for embeddings could possibly mimic human utility of language in that the exact meaning of a word is not so discrete; rather, a word's definition is not so strictly tied up, and thus a highly parameterized model may be overfitting to the exact usage of a word rather than its context. While this problem has been improved on with modern pre-trained language models like BERT, it remains an issue for static vector representations like Word2Vec.

Therefore, validation of quantized word vectors may lie in the fact that the word's meaning is somewhat lost in this context, and that forced loss of meaning provides deep learning models using a pretrained embedding that was quantized a more fluid latent space for the task at hand.

4 Conclusion

In conclusion, there's been many exciting jumps in the field of word vectors. Since their introduction, incremental improvements have showcased just how powerful the approach actually is.

I would like to propose further research, summarized from the discussion section as: combined PCA + Word2Bits and Word2Bits on large machine translation tasks.

Furthermore, it would be beneficial to attempt to analyze what the quantization is dropping in relevance to improving testing scores.

I predict machine translation will benefit especially from the simple quantization function used in Word2Bits, and I hope that similar quantization functions may yield even better results.

Finally, I expect quantization to be powerful enough in regularization that post-processing techniques will be more or less useless. The cost function is already made to maximize variance across the dataset in the given space, and paired with quantization, the hyperparameter for vector size should be sufficient to yield maximal results.

References

[1] Raunak, Vikas, et al. "Effective Dimensionality Reduction for Word Embeddings." Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019), Association for Computational Linguistics, 2019, pp. 235–43. DOI.org (Crossref), doi:10.18653/v1/W19-4328.

[2] Mu, Jiaqi, et al. "All-but-the-Top: Simple and Effective Postprocessing for Word Representations." ArXiv:1702.01417 [Cs, Stat], Feb. 2017. arXiv.org, http://arxiv.org/abs/1702.01417.

[3] Pennington, Jeffrey, et al. "Glove: Global Vectors for Word Representation." Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 2014, pp. 1532–43. DOI.org (Crossref), doi:10.3115/v1/D14-1162.

[4] Cai, Zhaowei, et al. "Deep Learning with Low Precision by Half-Wave Gaussian Quantization." 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), IEEE, 2017, pp. 5406–14. DOI.org (Crossref), doi:10.1109/CVPR.2017.574.

[5] Ling, Shaoshi, et al. "Word Embeddings with Limited Memory." Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Association for Computational Linguistics, 2016, pp. 387–92. DOI.org (Crossref), doi:10.18653/v1/P16-2063.

[6] Lam, Maximilian. Word2Bits - Quantized Word Vectors. Mar. 2018. arxiv.org, https://arxiv.org/abs/1803.05651v3.

[7] Mikolov, Tomas, et al. "Distributed Representations of Words and Phrases and Their Compositionality." Advances in Neural Information Processing Systems 26, edited by C. J. C. Burges et al., Curran Associates, Inc., 2013, pp. 3111–3119. Neural Information Processing Systems, http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf.