Word Embeddings with Limited Memory

Anonymous ACL submission

Abstract

This paper presents a systematic evaluation of learning or find word embeddings with limited memory. We study the effect of limited precision data representation and computation on word embedding. Our results show that word embedding can use only 8 bit wide fixed-point number representation with no degradeation in the word similarity task. And we also demonstrate methods to directly training these limited precision data representation.

1 Introduction

First paragraph: For each evidence, add some reference (looking the bib files ccg,cited,newref first and then search google).

What is word embedding (1-2 sentences). Compared to many current NLP systems and techniques treat words as atomic units, word embedding is a set of language modeling and feature learning techniques where words or phrases from the vocabulary are mapped to vectors of real numbers in a low-dimensional space.

They are useful, why (dense representation, generalization, benjio's NNLM JMLR paper) and applications (features for parsing, tagging, etc.). The word embedding provides the measure of similarity betweem words. " By training a neural network language model, one obtains not just the model itself, but also the learned word representations, which may be used for other, potentially unrelated, tasks. This has been used to good effect, for example in (Collobert and Weston,2008; Turian et al., 2010) where induced word representations are used with sophisticated classifiers to improve performance in many NLP tasks." "Distributed representations of words have been shown to benefit NLP tasks like parsing (Lazaridouet al., 2013; Bansal et al., 2014), named entity recognition (Guo et al., 2014), and sentimentanalysis (Socher et al., 2013)."

However, word embedding will be effective when large number of dimension is generated. Thus, loading all the word embedding into memory takes a lot of time, or even not feasible for limited memory machines. For example,200 dimensional vector for 3 million words and phase takes up 1.5 GB memory.

Second paragraph: explain how much bits indeed we need to represent a vector. We show that 4bit-8bit number representation is enough to achieve same result on word similary task as orginal one. For 8 bit representation in 100 dimemsion, every word is array of integer in range1-256 of size 100, and we can futher compressed this vector using some technique to achieve best space-effeciency.

Third paragraph:

In this paper, we present several ways to generate word embeddings with limited memory. The first method is to directly map original 64 bit wide fixed-point into 8 bit value. Secondly, low wide fixed point vector can be directly trained. It significantly reduce memory usage during training, enabling larger models to fit within the given memory capacity.

Fourth paragaph: The rest of this paper is organized as following. This paper will first introduce vector truncation technique and show the serval word simliarity task preformance on these truncated vector.

2 Related Work

First paragraph: Brief summary about what you want to review in this selection, e.g., word embedding methods, and model compression in neural networks. Word2vec is an efficient implementation of the continuous bag-of-words and skip-gram architectures for computing vector representation of words. (Mikolov et al., 2013a)(Mikolov et al., 2013b)

Second paragraph: Word embedding.

Third paragraph: Model compression. (not sure whether this is the right name, you can search it). `http://arxiv.org/pdf/1502.02551.pdf` deep learning with limited numerical precision introduce stochastic rounding in deep networks training using only 16 bit wide fixed-point representation. Sparse Overcomplete Word Vector Representations that transform word vectors into sparse (and optionally binary) vectors. ”The resulting representations are more similar to the interpretable features typically used in NLP, though they are discovered automatically from raw corpora. Because the vectors are highly sparse, they are computationally easy to work with.” `http://www.cs.cmu.edu/~mfaruqui/papers/acl15-overcomplete.pdf`

3 Method

In this section, we introduce our word embedding algorithms when the memory is limited. We first introduce a brute-force approach to directly truncate the word embeddings with expected dimensions. Then we introduce our way to learn the word embedding by considering limited memory.

3.1 Truncation

Directly projection 64 bit float into range 1-256. Lets say we want to use 8 bit to represent any value is the vector 8 bit = 256 number. The idea is to map all the original value to these 256 number. Since we found that most of vector value is in range from -3 to 3. Every real value is divided by a scale factor to get into range -128 to 127 (if bigger than 127 set it to 127 and if smaller than -128 set it to -128). When do computation, times the scale factor to truncated value to estimate original value.

3.2 Learning with Limited Memory

Since in word2vec, update value of vector is learning rate times gradient of maximum likelihood which is usually very small to make difference in small bit wide number representation. We have three technique to solve it. 1. We still use 8 bit wide vector for training but use seperate table to store update value.

2. We found 20 bits is enough to coverage these value. So we can use 20 bits wide number vector for training and then use truncation method above to compressed to 8 bits.

3. Stochastic rounding Stochastic rounding: The probability of rounding x to floorx is proportional to the proximity of x to floor*x:

Stochastic rounding is an unbiased rounding scheme and possesses the desirable property that the expected rounding error is zero, i.e. E (Round (x,hIL, FLi)) = x

4 Experiments

4.1 Datasets

(1) AnnotatedPPDB,a subset of phrase pairs from PPDB which are annotated according to how strongly they represent a paraphrase relationship, and (2) MLParaphrase,a re-annotation of the bigram similarity dataset from Mitchell and Lapata (2010), again annotated for strength of paraphrase relationship SimLex-999 is a gold standard resource for the evaluation of models that learn the meaning of words and concepts.

SimLex-999 provides a way of measuring how well models capture similarity, rather than relatedness or association. The scores in SimLex-999 therefore differ from other well-known evaluation datasets such as WordSim-353 (Finkelstein et al. 2002). The following two example pairs illustrate the difference - note that clothes are not similar to closets (different materials, function etc.), even though they are very much related:

Shyam simliarity tasks.”Evaluation of Word Vector Representations by Subspace Alignment” Qvec. the evaluation score depends on how well the word vector dimensions align to a matrix of features manually crafted from lexical resources. The evaluation score is shown to correlate strongly with performance in downstream tasks (cf. Tsvetkov et al, 2015 for details and results). QVEC is model agnostic and thus can be used for evaluating word vectors produced by any given model.

4.2 Compared Algorithms

Cosine distance. Spearsman correlation.

4.3 Results and Discussion

After using truncation method, the embedding file size reduces significantly. The performance on the several test dataset remains almost same until mapping orginal to 4 bits wide number representation.

5 Conclusion

References

[Mikolov et al.2013a] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. 2013a. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119.

[Mikolov et al.2013b] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. 2013b. Linguistic regularities in continuous space word representations. In *NAACL-HLT*, pages 746–751.