Word2vec: A Brief Summary

Matt McCarty (mdm12)

October 2020

1 Introduction

Word2vec is a family of vector space models developed by researchers at Google.[1] The family of vector space models relies on several techniques (neural networks, skip-gram modeling, hierarchical softmax, subsampling and others) to achieve a word embedding in the vector space model that is more robust than the traditional bag-of-words representation.[2] The researchers discover that this sophisticated vector space model is capable of producing rich relationships between words that resemble an algebra. One example the researchers provide is vec("Madrid")-vec("Spain")+vec("France") is closer to vec("Paris") than any other word vector.[3]

2 Body

There are a number of techniques that are provided within the word2vec framework, the source code of which is publicly available[1]. While the details of all the features cannot be provided in a brief summary, the highlights of the major features will be provided below.

2.1 The Skip-gram Model

The Skip-gram model uses the context of the words around a given word to improve the given word's vector space representation. Given a sequence of training words $w_1, w_2, \ldots, w_T$, the objective function of the Skip-gram model is

[1] code.google.com/p/word2vec

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log\, p(w_{t+j}|w_t)$$

where c is the size of the training context. The larger c is, the more context words are used for a given word w_t, which improves accuracy. Of course, larger c's also increase the training time, so this trade-off has to be carefully considered by the practitioner. The conditional probability $p(w_{t+j}|w_t)$ is defined with the softmax function

$$p(w_O|w_I) = \frac{\exp({v'_{w_O}}^T v_{w_I})}{\sum_{w=1}^{W}\exp({v'_w}^T v_{w_I})}$$

where v_w and v'_w are the input and output vector representations of w, and W is the number of words in the vocabulary. This formula is computationally impractical since the cost of computing $\log\, p(w_O|w_I)$ is proportional to W, or the size of the whole vocabulary. Therefore, the researchers propose using the hierarchical softmax in place of the normal softmax.[2]

2.2 Hierarchical Softmax

The hierarchical softmax approximation of the full softmax in the context of neural networks was proposed first by Morin and Mengio.[4] Hierarchical softmax utilizes a binary tree representation of the output layer of the neural network, where the W output nodes are the leaves of the tree. The non-leaf nodes specify the relative probabilities of the two child nodes. With this structure, the probability distribution can be obtained with only $log_2(W)$ nodes being evaluated, instead of W nodes.

2.3 Subsampling Frequent Words

Similar to the IDF weighting that is done in many vector space models, word2vec subsamples frequent words with the formula

$$P(w_i) = 1 - \sqrt{\frac{t}{f(w_i)}}$$

where $f(w_i)$ is the frequency of the word w_i and t is a chosen threshold, which the researchers suggest should be around 10^{-5}.[2] This function (especially with the given constant) aggressively punishes frequent words.

2.4 Learning Phrases

Many phrases have a meaning which is different than the simple composition of the meanings of the words in the phrase (such as Philadelphia Eagles). To handle such phrases, word2vec supports using multiple words as a single token. Other word combinations which do not have any additional meaning such as 'this is' remain unchanged.[2]

3 Conclusion

There are several limitations to the simple bag-of-words representation in traditional vector space models. For example, they do not consider the context of the word within a document, and they are incapable of handling phrases whose individual words have a different meaning than their composition (such as Boston Globe). The word2vec family of vector space models improve on this naive approach, and in doing so, create a rich structure in which the distance (implemented as cosine distance) between word vectors reflect deep semantic relationships.

References

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *ICLR Workshop*, 2013.

[2] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 2013.

[3] Tomas Mikolov, Wen-tau Yih and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. *In Proceedings of NAACL HLT*, 2013.

[4] Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the international workshop on artificial intelligence and statistics*, pages 246–252, 2005