Negative sampling pap

Authors	Tomas Mikolov and team
Class	negativesampling
Created	@Jan 3, 2021 10:40 PM
Materials	https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf
Reviewed	☐
Status	
Type	neurips

Summary

This paper is an extension to the previously presented word2vec paper where CBOW and skip gram methods were introduced. This paper presents methods to improve the quality of the vectors and training speed.

The actual skip-gram formulation uses softmax function,

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$

This formula requires the sum of probabilities of all the words in the corpus in the denominator which is computationally very heavy. This led the authors to propose a new model.

Hierarchical Softmax

This is an alternative to full softmax where instead of evaluating W output nodes only log(W) are evaluated. The words frequencies are converted to Huffman trees and the words are grouped together based on their frequencies.

Negative Sampling

An alternative to the hierarchical softmax is Noise Contrastive Estimation which says that a good model should differentiate data from noise by means of logistic regression. Here the data is taken as

a set of all (words, context) in the corpus and a set of all (words, context) that do not co-occur in the corpus. But if we take one word, then the word might not always be co-occurring hence we take k words. Experiments suggested k in the range 5-20 for small training sets and 2-5 for huge training sets.

Sub-sampling of frequent words

Whenever we are picking frequent words, words like the, a, and repeat a lot. To reduce the importance of such words, a sub-sampling technique is used where

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

t is the threshold around 10^-5 and f(w) is the frequency of the word w.

Important Points

- The model is built on bigrams to accommodate the learning of phrases like "New York", "San Jose" etc.
- The accuracy of the model was 72% when a huge corpus of 33 billion words was used and 66% when 6 billion words are used.
- The word vectors are showing a linear structure that makes it possible to perform precise analogical reasoning using simple vector arithmetics. As the word vectors are trained to predict the surrounding words in the sentence, the vectors can be seen as representing the distribution of the context in which a word appears.
- These values are related logarithmically to the probabilities computed by the output layer, so the sum of two-word vectors is related to the product of the two context distributions. The product works here as the AND function: words that are assigned high probabilities by both word vectors will have a high probability, and the other words will have low probability. Thus, if “Volga River” appears frequently in the same sentence together with the words “Russian” and “river”, the sum of these two-word vectors will result in such a feature vector that is close to the vector of “Volga River”