A Quick Summary: Distributed Representations of Words and Phrases and their Compositionality

Original Paper: https://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality.pdf

22nd Feb 2019

1 Ideas:

The Skip Gram model predicts context words from a center word and assigns an embedding for each word. In this paper, the authors show how to improve the quality of the embeddings through various methods as well as reducing the time required to compute these embeddings.

(a) Hierarchical Softmax: an alternative to compute probabilities. Refer to the summary of the paper "Hierarchical Probabilistic Neural Network Language Model".

(b) Negative Sampling: During every word that we go through during the training of skip-gram, calculating the log likelihood requires us to compute the dot product between that word's embedding and the embedding of all the other words in the corpus. This is very slow. Negative Sampling essentially changes this objective function to solve this problem.

(c) Subsampling of Frequent Words: Words like "the" are too common, and they may co-occur with "rare" words such as "zebra" more than the co-occurance of "zebra" and "stripes", although the latter pair is much more rich semantically. Thus, this method aims to solve this problem through discarding each word with a probability, based on how common it is.

2 Explanations:

(a) Refer to the summary of the paper "Hierarchical Probabilistic Neural Network Language Model".

(b) In the skip-gram model, given that we are at timestep t looking at word c, and denoting the center vectors as v and the context vectors as u, the "true" probability for a particular context word "o" at that timestep ought to be:

$$p(o|c) = \frac{exp(u_o^T v_c)}{\sum_{w=1}^{V} exp(u_w^T v_c)}$$

Note: we are interested in calculating probability since we want to **maximize** it. Now, in the formula given above, we have to calculate the denominator, which is too huge. So, instead of taking the logarithm of that as our objective function at time t, we take the following instead:

$$J_t(\theta) = log(\sigma(u_o^T v_c)) + \sum_{i=1}^{k} \mathbb{E}_{w \sim P(w)}[log(1 - \sigma(u_i^T v_c))]$$

The intuition behind the above is as follows: we want to maximize the probability of seeing the words that appeared around the actual word, and minimize the probability of seeing other words that don't.

(c) We will discard each word with probability

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

Where $f(w_i)$ is the count, and t is a chosen threshold (say 10^{-5}.

3 Results:

(a) HS with 10^{-5} subsampling is the best.

4 Notes:

(a) Subsampling seems to increase performance by quite a bit.