MODEL
https://drive.google.com/drive/folders/1we2OS9PDYwNUbqTALfJOxBOroxSh1qBH?usp=sharing

Configuration

1. Cross Entropy

batch_size	skip_window	num_skips	max_num sampled	Learning rate	Average Loss	Word Analogy Score
128	2	4	70000	1.0	4.72	33.7

2. NCE

batch_size	skip_window	num_skips	max_num sampled	Learning rate	Average Loss	Word Analogy Score
128	4	4	150000	1.0	1.42	34.0

1. Batch Generation

Imp Variables -

- n - counter to count no. of samples added to batch, goes from [0,batch_size]
- data_index - center word index around which samples are generated (this is global and is NOT
- reset on every batch)
- ret_idx - index of batch and label which needs to be returned
- count - counter from [0, num_skips]

Everytime batch generation is called, a counter starts from data_index. num_skip number of words are taken around each center word and pairs are inserted in batch and label. Max skip_num of numbers are taken from either side of the center word. The loop is broken as soon as n reaches batch_size.

When the next batch starts it starts from where the last batch left, so that different batches are generated every time batch generation is called.

CORNER CASES -

1. If data_index ==0 we can't take any word to its left. Therefore make data_index = skip_window
2. If data_index reaches end of vocabulary reset it to 0+skip_window so that it can generate the batches again.

II. Cross Entropy Loss

A - We use scalar multiplication to multiply rows of input and true_w matrix. Then we use reduce_sum along row to create a (batch_size X 1) matrix.

B - Here we need to calculate not just one element but sum of all rows of true_w with a particular row of input (context word). Therefore matrix multiplication would be easier. We take transpose of true_w and then dot product with inputs. The we take the exponent and then to sum over all output words reduce sum along rows. Finally we return the loss after taking log.

III. NCE LOSS

1.First we convert numpy array to tensors, to speed up tf processing.

2. We generate required matrices of small values (10^-10) to add then to log functions. This is done because if at any point in the algorithm value reaches 0 then the log of that would be negative infinity, which we do not want.

3. We use tf.nn.embedding_lookup to get word embeddings from their ids

4. We then write the equations verifying dimensions at each step. Finally, we return a [batch_size X 1] matrix denoting nce loss for each pair

IV WORD ANALOGY

Since words can be represented in N dimensional space as vectors, we can apply all vector operations to word embeddings.

1. File parsing - We first parse out all the example and option pairs out of the given file.

2. The relation between 2 vectors can be represented by their difference. Thus we compute the difference of of pair words to get the relation between them. The idea is to compute a single relation from all the example pairs.

3. To check that all the example pairs contribute the same to the final relation, we make the difference vector into a unit vector, by dividing it with its length.

4. We add all the unit vectors to get a single vector depicting the final relation.

5. We then iterate through the option pairs and keep 2 counters. 1 for minimum value and 1 for max value. We make the difference vector of option pair a unit vector in a similar fashion. Then we compute their correlation by taking their element wise product and sum the array. If the product is low that means there is minimum relation between the two words and vice-versa.

6. We finally write the output to a file in the required format.