1. Hyper parameter exploration

Tuning hyper parameters is a challenging tasks and requires to perform lot of experiments to find out how values of hyper parameters affects accuracy of models.

I tried different combinations for hyper-parameters and here is my findings of how it affects a final accuracy.

maximum number of steps:

This parameters gives number of times batch data feeded to network and number of times we optimize out loss function. As noticed in results, for initial few steps loss decreases exponentially but after few steps it became stagnant. After this steps model has found local minima. Higher than certain number of epochs doesn't affect final results or loss function. Number of epochs required depends on configuration and loss function.

In my implementation, Cross entropy loss was achieving local minima after around 20000 steps and NCE loss was achieving local maxima after around 120000 steps.

batch size:

Higher batch size leads to finding minima in less number of epochs. Higher number of epochs with high batch size doesn't improve result. Because after some epochs new batches will contain data which model has already seen. I have tried three different values of this parameter, 128,64, 256. Value 128 is giving relatively higher accuracy.

skip windows:

We are using this hyper-parameter to decide how many words in left and right to be considered as a context word. If value of this hyper parameter is really high then we will consider which are not actually in the context. So, keeping high value of this parameter will reduce accuracy. I have tried 3 different values of this parameter, 2,4 and 8. Value of 2 is giving higher accuracy for both models.

num_skips:

This property is used to decide the number of samples we want to draw in a window. We can implement different intellignent algorithms to decide which samples to take from given window. If algorithms is simply taking first num_skips samples then it's better to take all possible samples. I have tried three different values of this parameter, 4,8,16. Value of 8 is giving optimal result. Performance also depends on value of skip_windows.

embedding size:

Embedding_Size hyper-parameter is used to decide embedding size of each word. High embedding size can help in improving results. But having higher embedding size will increase number of parameters. To train higher number of parameters we need really huge data. I tried four different values of embedding size, 64,128, 256 and 300. Value 128 is giving best result. If we had larger data then we can use higher embedding size.

2. Analogy task results and accuracy

I explored several different configurations. Here are five different configurations that I believe help to understand patterns.

Epochs (num of steps)	Batch size	Embedding Size	Skip window	Num skips	Accuracy (cross entropy)	Accuracy (NCE)
200001	128	128	8	4	31.7 Loss : 4.79	31.4 Loss : 1.40
200001	128	300	8	4	33.3% Loss : 4.85	32.8% Loss : 1.44
50001	64	128	8	4	31.3% Loss : 4.16	29.4% Loss : 1.23
150001	**128**	**128**	**4**	**2**	**33.8% Loss : 4.85**	**36.2% Loss : 1.37**
150001	256	256	4	2	32.6% Loss: 5.54	36.1% Loss:1.34
150001	128	128	16	8	31.5% Loss : 4.83	34.1% Loss : 1.40

As we can see in the table, 4th configuration is giving best results. I have already discussed reason behind this accuracy in previous part of this document. Lower skip window helped to generate data which are relevant and not so large batch size is used to ensure training speed of models.

As mentioned in previous part, For higher than 128 embedding size I believe we need more data. Otherwise it wont be able to set weights properly.

3. Top 20 similar words for {first, american, would}

Similar words depends on window size and number of skips. For lower window size closer words in context get higher similarity. For higher window size, it looks for a wider context.

Here is a table for cross entropy loss based similar words.

Word	Similar Words
First	the, to, of, and, in, a, out, more, it, high, with, is, will, over, them, long, only, into, for, him
American	century, late, last, united, early, end, during, age, first, second, mid, empire, until, same, states, following, beginning, west, north, war
Would	and, in, for, with, the, to, of, by, it, is, known, a, that, also, from, being, s, he, but, or

Here is a table for NCE loss based similar words.

Word	Similar Words
First	five, eight, only, more, four, such, see, i, their, called, nine, have, for, many, number, general, links, which, later, between
American	also, has, term, from, way, society, french, still, any, to, been, act, by, that, means, wavefunction, revolution, campaign, thin, cyrix
Would	it, from, also, has, act, gombe, term, messina, achille, strangles, still, been, apok, elevate, kenyatta, ignoring, empedocles, ltr, minsk, french,

As we can see, closest word for first are numbers, which makes sense. Other closest word for first are adjectives.

Closest words for ‘American’ are words like society, French, century. French is other country and phrases like ‘american century’ and ‘american society’ are quite frequent words.

Most of the words used with would are ‘it’, ‘from’, ‘in’ etc. Most of them are prepositions. And ‘it would’ is also really common words.

4. NCE Loss

In cross entropy loss, we have to calculate probability for all words and each iteration and that is really time consuming. Rather we can limit this search by some specific words and this will speed up process and it can also increase accuracy.

Too decide which samples to consider while calculation is decided by some classification process. This classifiers gives binary classification of whether sample is from data distribution or from noise distribution. Every word that can not be in the context is considered as a noise word. Generally classifiers like logistic retraction is used to get probability of each sample being noise. The main advantage of this method is that it’s training time is independent of vocabulary size.

If we want to find out context words for word ‘w’, we create a binary classification problem. We will treat training data as +ve examples and noisy data -ve examples. We will use unigram distribution of the training data as the -ve examples. Noisy samples are always more probable than context words. Let’s assume that noise samples are k times more probable than normal data samples. In such cases we can give probability of some word by probably of sample being a +ve sample plus k times it being a negative sample. This probabilities can be denoted in terms of unigram samples.

Simplifying this equation we can get loss by adding k samples of noise rather than entire vocabulary. So, Its not dependent of vocabulary size. If we increase value of k then training time will be increased.