CSE – 538

Natural Language Processing

Prof. Niranjan Balasubramaniam

Assignment 1

Name: Anurag Dutt

SBUID: 113019209

Purpose:

The purpose of the aforementioned assignment was to understand word embeddings using word2vec and implement skipgram methodology to predict neighboring words for a given context. Using our training dataset we learn our word embeddings, and we also use our hyperparameters to better tune our neural network so as to facilitate better contextual understanding of the word space.

The hyperparameters that we explore in our task are as follows:-

1.) Batch Size:-

Batch size is the length of the mini-batch matrix or the number of instances we choose to train our model. Increasing batch size increases the accuracy of our models by a small amount, but this is not necessarily always the case. For eg. in our case when I increase the batch size to 192 the accuracy increases, but when I increase the batch size to 256, the accuracy decreases.

2.)Num_sampled:-

Num sampled basically means the number of negative samples chosen while estimating loss using noise contrastive estimation. This variable has a direct correlation. Increasing it has a great impact on the model (in my experiments), as it increases the accuracy of the matrix, whereas decreasing it, decreases the accuracy of the matrix.

3) Learning_rate:-

Learning rate refers to the step length the model takes while performing gradient descent. There is a clear relation between learning rate and accuracy. Decreasing the learning rate increases the accuracy of the model. However, taking a too small a learning rate may drastically increase the training times and also make the model more susceptible to overfitting. However, taking too small a learning rate will make the model skip important optimal values. We try and keep the learning rate between 0.2 and 1, for optimal training purposes.

4.) Num_skips and Skip Window:-

Num skips refers to the number of samples being taken in each window of estimation. Increasing Num skips increases the number of samples in each window leading to better accuracy. There were many ways in which Num_skips were tested. We can take samples, either from left or right, alternately from left and right (which was in my final submission case) or randomly. Skip window refers to number of samples in the left or right of the center word in the window. Again, increasing skip window increases the accuracy of our prediction. (This is true for the NCE case, however for the cross-entropy case this works in reverse. I have mentioned the reason in the table)

5.) Max_Num_Steps:-

Max num steps increases the number of batches generated, and increasing it should generate more sample space for the model to learn. However, increasing it too much may make the model susceptible to overfitting. While tuning the model, I tried increasing the Max_num_steps to 300001 and 400001, and found that my accuracy decreases. As such I kept my max_num_steps fixed to 200001, which was the default case.

Note: For all cases, embedding size was kept fixed at 128. Note that, once we adjusted the num_sampled hyperparameter to 128, the NCE model conclusively works much better than the standard model.

6.) Cosine Similarity:

While calculating cosine similarity of our example set of 3 words, we can choose to take the average of all the three words or we can take the maximum or minimum of the set. Although I got slightly better results in accuracy while taking the “max” mode, I chose to take average, as average or mean is a much better representation of the example set, rather than max or min.

Results for hyperparameter tuning:-

I did multiple experiments on each of the 5 hyperparameters mentioned above (11 experiments on cross entropy loss and 16 experiments on NCE, to get best case accuracy). Loss values have been formatted the nearest 2nd decimal to format the table. For actual values, please refer to the attached excel sheet.

Cross Entropy Estimation Loss

Cross Entropy

Batch Size	**Skip Window**	**Embedding Size**	**Max Steps**	**Learning Rate**	**Num Skips**	**Loss**	**Accuracy**	**Test case**	**Comments**
128	4	128	200001	1	8	4.83	31.2	Default	This is the default case
128	2	128	200001	0.5	4	4.72	31.4	1	I tried reducing skip window and num skips and this gave improved value for the accuracy. I believe the reason for this was a smaller window gave words which have a much higher association with the center word
128	32	128	200001	0.5	64	4.86	30.2	2	I checked the previous association and when I increase num skips and skip window the accuracy detoriorates. The reason might be because the model picks up more noisy associations from the window and the relevant training instances might be lost
128	4	128	400001	0.5	8	4.82	31.2	3	I also tried increasing the number of steps, but the accuracy remained almost the same. Clearly for our problem the model coverges in 200001 steps, and addition of further max steps is not very helpful

128	8	128	300001	0.1	16	4.84	29.9	4	Again in this case I try reducing the learning rate. Our problem converges well with a learning rate of 0.5 or 0.75 and setting it very proves harmful, because of overfitting we would require a much higher number of max steps to converge
256	2	128	200001	0.5	4	5.35	31.4	5	Increasing the max size also doesn't help much. Our accuracy improves slightly as compared to the base case
256	32	128	200001	0.5	64	5.55	30.1	6	We try and increase both the batch size and num skips and skip window. Clearly increasing the skip window doesn't work
128	32	128	400001	0.2	64	4.86	30	7	Same issue as the previous. Even with a higher number of max steps and a much lower learning rate, our model fails to converge optimally
192	32	128	200001	0.75	64	5.26	30.1	15	Same issue as the previous, even a higher learning rate is unhelpfu as our model is training on more noisy associations
192	***2***	***128***	***200001***	***0.75***	***4***	***5.09***	***31.8***	***16***	***This is our best case. We increase the batch size slightly to aid training but not very high. We slightly decrease our learning rate to help the model converge, and decrease our num skips and skip window to tune our***

									model
192	2	128	300001	0.75	4	5.07	31.2	17	Lastly increase the number of steps is not very helpful in our case and our best case accuracy stays at 31.8

Noise Contrastive Estimation Loss

Batch Size	**Skip Window**	**Embedding Size**	**Max Steps**	**Learning Rate**	**Num Skips**	**Loss**	**NCE Accuracy**	**Test case**	**Comments**	**Negativ Sample**
128	4	128	200001	1	8	1.21	30.3	Default	This is the default case	64
128	2	128	200001	0.5	4	1.47	30.7	1		64
128	32	128	200001	0.5	64	1.21	32.5	2		64
128	4	128	400001	0.5	8	1.35	30.6	3		64
128	8	128	300001	0.1	16	1.53	30.6	4		64
256	2	128	200001	0.5	4	1.13	30.7	5		64
256	32	128	200001	0.5	64	1.75	30.6	6		64
128	32	128	400001	0.2	64	1.39	31.2	7	While testing with the negative sample size of 64, we don't getmuch variance in our results. However on testing with negative sample size of 128, our results improve drastically, which we illustrate in the table below	64

Batch Size	**Skip Window**	**Embedding Size**	**Max Steps**	**Learning Rate**	**Num Skips**	**Loss**	**Accuracy**	**Comments**	**Comments**	**Negative Samples**
128	4	128	200001	0.5	8	2.01	31.5	9	This is default case.	128
128	32	128	200001	0.5	64	8.43	33	8	Increasing the num skips and skip windows gives us a much better prediction for NCE and hence we stick to values of 32 and 64 respectively for all the subsequen iterations	128
256	32	128	200001	0.5	64	4.44	29.2	10	We try increasing the batch size, but as expected, we get very noisy predictions.	128
192	32	128	200001	0.5	64	7.05	33.1	11	Hence we reduce the batch size to 192 (although it is greater than 128) and find 192 proves to be the optimal value for our case	128
192	32	128	300001	0.5	64	8.79	31.9	12	We also try increasing the number of max steps, but our accuracy doesn't increase much. I am assuming this might be due to overfitting	128
192	32	128	200001	0.2	64	4.14	30.4	13	We also try reducing the learning rate, however we observe that our model doesn't converge properly in the given steps.	128
192	32	128	200001	1	64	9.46	31.9	14	We keep our learning to a higher value such as 1, which is same as the default problem, however we observe we can tune optimally	128
192	***32***	***128***	***200001***	***0.75***	***64***	***8.81***	***35.6***	***15***	***Hence we reduce the learning rate slightly to 0.75 and observe there is a drastic increase in our accuracy which increase by almost 5.3%, and thus we keep our last case as our best case***	***128***

Words nearest to first:

Cross-Entropy Loss:

term, book, word, most, during, early, before, time, being, work, use, american, until, especially, up, state, theory, against, law, about

Noise Contrastive Estimation:

later, eight, some, illusions, time, would, goods, no, including, general, insult, s, unite, four, property, myself, proudhon, mind, seven, was

Words nearest to American:

Cross-Entropy Loss

english, would, proudhon, modern, free, french, called, joseph, term, including, against, western, pierre, revolution, might, ancient, how, being, especially, political

Noise Contrastive Estimation:

later, first, de, that, so, goods, was, some, economics, prisons, social, used, property, law, a, insult, movement, saw, time, general

Words Nearest to would:

Cross-Entropy Loss

i, him, proudhon, could, joseph, called, american, using, pierre, will, labor, can, through, within, free, my, groups, stirner, t, against

Noise Contrastive Estimation:

including, time, illusions, ego, over, france, mind, eight, s, some, no, unite, accepted, products, insult, goods, myself, warren, general, groups

Noise Contrastive Estimation

In a standard and conventional word association model, the significance of the target word and the context word is obtained by training on the softmax functions when a normal cross entropy loss is calculated. However the process involves calculating over nxn problem and also involves exponentiating the log likehood inputs which is very expensive. Evaluating log likelihood of over the entire vocabulary is a very expensive pro NCE uses a sampling based approach to drastically reduce the computational complexity by converting this to a 2n problem. As per the NCE methodology we draw samples from the true distribution and also some noise distribution. Using logistic regression, we try and discriminate between our aforementioned drawn samples.

To understand a normal neural network, we have hidden layers based on the size of the vocabulary and a large vocabulary leads to very large computational costs. The idea behind NCE is to learn the proportion of true data in proportion to noise. The key idea behind using NCE is that instead of using the entire vocabulary, we can train on k negative samples which decreases the training time considerably. Interestingly, if we increase k enough, we start approaching the softmax function.

NCE convert the multi-class classification problem to a binomial class-classification problem, thus reducing computational costs by a large amount. Basically, we take the probability of P = 0 in the second term (we take 1 minus P), we are essentially trying reduce the noise, thereby increasing likelihood od a successful prediction and maximizing the chances of positive likelihood. So to put this the first term in the objective likelihood function is trying to increase the probability of the target word and the second is trying to reduce the probability of noise. Hence the training doesn’t depend on the entire vocabulary given to us.

Understanding the mathematics behind the NCE method, the first step is to sample the data from the true distribution and the noise (unigram):

$$P_r(D=1, \omega_o / \omega_c) = \frac{K}{1+K.} q_{\omega_o}$$

$$P_r(D=0, \omega_o / \omega_c) = \frac{1}{1+K.} p(\omega_o / \omega_c).$$

$$\text{or } 1 - P_r(D=1, \omega_o \mid \omega_c) = \frac{1}{1+K} p(\omega_o / \omega_c).$$

$$P_r(D=1 / \omega_o, \omega_c) = \frac{P_r(\omega_o \mid \omega_c)}{P_r(\omega_o \mid \omega_c) + K \times q(\omega_o)}.$$

We use this to calculate the sample from true distribution and then take the sigmoid as mentioned by the paper, essentially building a classification problem with K noise terms and thus probability becomes:

$$P_r(D=1 / \omega_o., \omega_c) = \sigma\left(s(\omega_o, \omega_c) - \log(K q_{\omega_o})\right)$$

The loss function, which is essentially the negative of log likelihood of the probability distribution shown above (q essentially is our negative sampling distribution with K representing the list of negative words):

$$J(\theta) = -\sum_{w_o, w_c \in D} \left(\log\left(\sigma\left(s(w_o, w_c) - \log(k q_{w_o}) \right) \right) \right)$$

$$+ \sum_{x \in V_k} \log\left(1 - \sigma\left((w_o, w_c) - \log(k q_{w_o}) \right) \right) \Bigg) \Bigg)$$