Report

Hyper Parameter Tuning:

Parameter Description:

```
# Hyper Parameters to config
  batch_size = 128  # No of training examples in each batch
  embedding_size = 128  # Dimension of the embedding vector.
  skip_window = 4      # How many words to consider left and right.
  num_skips = 8      # How many times to reuse an input to generate a label.

# maximum training step
 max_num_steps  = 200001
```

Nce Model:

1st Model :

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
128	4	8	200001	34.6%	1.5033708277868485

2nd Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
128	2	4	200001	34.4%	1.4537452065239598

3rd Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
128	8	16	200001	33.9%	1.3783387459615477

4thModel:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
256	2	4	200001	33.9%	1.50916780224667

5th Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
64	1	2	200001	34.4%	1.279767415623586

6thModel:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
64	2	4	200001	34.8%	1.26403406615339

7th Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
64	2	4	800001	34.4%	1.4631933213499595

8th Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
64	2	4	400001	34.5%	1.2442161393751876

Cross Entropy Model:

1st Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
128	4	8	200001	33.4%	4.824304129600525

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
128	2	4	200001	33.6%	4.70401112203598

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
128	8	16	200001	34.1%	4.835974419879913

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
64	2	4	800001	33.5%	3.926882362318039

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
64	8	16	400001	33.3%	4.144266467761994

6th Model:

Batch Size	Skip Window	Num Skips	Max Num Steps	Accuracy	Loss
256	8	16	200001	33.6%	5.534155613517761

Observations:

1.For NCE model, decreasing the batch size results in good loss as well as accuracy for the word analogy task. On the other hand, if we increase the batch size, we are not seeing any improvement in the accuracy or loss. Although, decreasing the batch size improves the model, the time taken to train the model increases too. For small batch size, we are getting better results by removing repeated context in the examples as we are taking random context word for each target word.

2. Increasing the Skip Window doesn't help with the improvement in the NCE model loss. The best results we got was at skip window = 2, taking 2 words on the left and on the right. Anymore than 2, the word doesn't have significant relation to the target word. Although, taking skip window = 1 did give the comparable result against the model where we took skip window =2.

3.Increasing the iteration for NCE model didn't help much with the improvement in the accuracy for the analogy task. Although, we did see a decrease in the loss function for 4 millions iterations. Since, i was using pretrained models, where i have already run the nce loss function for a

couple of millions iterations. We didn’t see much improvement in the accuracy.
4.For Cross Entropy Model, increasing the skip window from 4 to 8 got us better model than for 8. We did see an improvement in the accuracy for the task but the loss function remained same for both the configuration. Also, decreasing the skip window did decrease the loss function and an improvement in the accuracy against the default configuration but better improvement was seen at higher values.
5.Increasing the number of iterations for cross entropy model didn’t help much with the accuracy of the analogy task but had a significant improvement in the loss function. Most probably, the specific set of examples gave a lower accuracy than other models. Also, increasing the batch size increased the loss function.

Learnings:

1. For Nce Model, it is better to have a small batch size and skip window for better model.
2. For Cross Entropy Model, it is better to have large skip window.
3. The decrease in loss function doesn’t mean increase in the accuracy of the task.
4. The number of iterations helps in decreasing the loss function but we don’t see any significant improvement in accuracy for the task by increasing the iterations to 8 millions or 4 millions.
5. The increase in batch size didn’t improve the model for cross entropy.

Top 20 Similar Words :

Best Cross Entropy Model (3^{rd}):

Accuracy : 34.1%

first	american	would
colloids, coast, main, callings, entire, report, cyphertext, triola, court, end, latter, beginning, same, until, most, best, name, original, following, last	taoist, menezes, peli, pharmacy, freedoms, reburial, european, fanfic, french, oncifelis, bezout, snowy, rascal, tyre, barzani, its, borges, eu, italian, german	shown, seen, deceptive, householder, argued, you, needing, been, believed, do, did, seems, does, should, we, said, could, will, might, must

Best NCE Model (6^{th}) :

Accuracy: 34.8%

first	american	would
efe, moment, use, end, late, year, way, part, battle, term, same, english, next, best, original, following, name, book, second, last	although, including, anti, william, all, general, how, early, while, so, during, china, abuse, before, later, ideas, since, others, law, history	others, no, had, robert, during, william, before, abuse, does, became, so, general, since, did, links, might, see, called, t, can

I am seeing that for both the models, there are some words coming in both of the above table.

Word Analogy Prediction file using best NCE & Cross Entropy Model:

1. nce_loss_model_analogy_prediction.txt
2. cross_entropy_analogy_prediction.txt

Noise Contrastive Estimation

We are interested in highly scalable models that can be trained on billion-word datasets with vocabularies of hundreds of thousands of words within a few days on a single core, which rules out most traditional neural language models

NCE is based on the reduction of density estimation to probabilistic binary classification. It is based on the idea is to train a logistic regression classifier to discriminate between samples from the distribution and from some noise distribution, based on the ratio of probabilities of the sample under the model and the noise distribution.The main advantage of NCE is that it allows us to fit
models that are not explicitly normalized making the training time independent of vocabulary size. The perplexity of Neural probabilistic language models trained using this approach has been shown comparable to those trained with maximum likelihood learning but at a fraction of the computational cost.
Suppose, we would like to learn the distribution of words for a specific context c, denoted by P^c. For this, we will create an auxiliary binary classification problem, treating the data from training as positive and from noise as negative P^k. We will choose unigram distribution of the training data as the noise distribution. If we assume that the noise samples are k times more frequent than data samples, the probability that the given samples came from the data is :

$P^c(D =1 |w) = P^c(w) / (P^c(w) + k * P^k(w)) = \sigma (\Delta s (w, h))$

Where $\sigma (\Delta s (w, h))$ is the logistic function and $\Delta s (w, h) = s(w,h) - \log(k P^k(w))$ is the difference in the scores of word w under the model and the (scaled) noise distribution. The scaling factor k in front of Pn(w) accounts for the fact that noise samples are k times more frequent than data samples.
We fit the model by maximizing the log-posterior probability of the correct labels D averaged over the data and noise samples:

$J^h (\theta) = E [\log P^h (D = 1|w, \theta) + k*E_{Pn} [\log P^h (D = 0|w, \theta)]$

$\quad = E_{P h d} [\log \sigma (\Delta s_\theta(w, h))] + kE_{Pn} [\log (1 - \sigma (\Delta s_\theta(w, h)))]$, In practice, the expectation over the noise distribution is approximated by sampling. Thus, we estimate the contribution of a word / context pair w, h to the gradient of Eq. 8 by generating k noise samples {xi} and computing :

$$\partial J^{h,w}(\theta) = (1 - \sigma(\Delta s_\theta(w, h)))\ \partial \log P^{h}{}_{\theta}(w) - \sum_{i=1}^{k} [\sigma(\Delta s_\theta(x_i, h))\ \partial \log P^{h}{}_{\theta}(x_i)]$$

$-$ $\qquad\qquad\qquad\qquad$ $\partial\theta$ $\qquad\qquad\qquad\qquad$ $\partial\theta$

$\partial\theta$

As we increase the number of noise samples k, this estimate approaches the likelihood gradient of the normalized model, allowing us to trade off computation cost against estimation accuracy.