CSE 538 NATURAL LANGUAGE PROCESSING

ASSIGNMENT 1 Report (SBU ID #111870086, Ayush Garg)

My initial accuracy (Before any hyper-parameter tuning) is 33.2% for Cross Entropy Model and 33.5% for the NCE model. The loss for Cross Entropy model is around 4.83 and for NCE is 1.38. I've used python 3.6 for this assignment.

Initial Steps: First when I implemented the assignment, For the analogy task, I used the difference vectors as suggested in the Question paper. I calculated b-a and compared it with d-c and then based on the cosine similarity predicted the most and least similar pairs. However that gave me an accuracy of only around **29.1% for cross entropy model and 28.7% for NCE model.**

However, then I came across a research paper where they had mentioned an alternative way of calculating the similarity between the vectors. Instead of taking the difference between both pairs and comparing, instead I used a-b+c=d equation. This equation is valid because if we take the difference (in direction) of 2 vectors and add it with the third, then it should give us the 4th vector. And hence I checked the cosine similarity of a-b+c with d and results were very good. My accuracy reached around **33.2% for Cross Entropy and 33.5% for NCE.**

1&2. HYPER PARAMETERS TUNING and DIFFERENT CONFIGURATIONS:

A) **MAX_NUM_STEPS:**
The default number of training steps that have been given to us is 200001. However, we know that the more we train the model, the better. Hence I will try to increase the maximum number of training steps and see what happens.

When I **double the number** of steps to **400001**:
For Cross Entropy Model, There is not much change by doubling the number of steps. There is a difference of only about 0.005 in the loss and accuracy remains the same i.e. **33.2%**. To which i conclude the model has already been sufficiently trained till 200001 steps.

For NCE model too, there was not much difference in the loss as well as accuracy. Loss remained 1.32 and accuracy **33.5%**

When I reduce max_num_steps to **60001**:
For cross entropy model, Reducing the number of training steps **did not have any significant effect** on the loss as accuracy. Loss was and accuracy was

For NCE, reducing the number of steps also did not have any significant impact on accuracy or loss. Loss was 1.12 and accuracy **33.7%**

B) BATCH_SIZE:

The default Batch size that is given to us is 128. So I will try to increase and reduce the batch size and see what effect it has on different models' loss and

When I **increase** the batch_size to **256**:
For cross entropy model, the accuracy **increased a bit to 33.5%**. The loss was around 5.5. Also, the processing was faster since more words were taken at a time for processing.

For NCE model, the loss came out to be 1.38. And accuracy was about **33.4%.**

C) SKIP_WINDOW:

Skip window considers how many words to skip left and right of our central word while running the model. The default skip window given to us is 4. We will try **reduce the skip window to 2** to skip lesser words and check whether that improves our accuracy. Since num_skips should be 2*skip_window, we change **num_skips to 4**
For Cross Entropy model, we observe that the loss significantly reduces to 4.73. Also, the **accuracy significantly improves to 33.9%**

For NCE, we observe that the loss improves to 1.34 and the accuracy significantly **improves to 33.7%**

Due to this improvement, we will try decrease num_steps even more and see what happens.
Now we will put **skip_window=1 and num_skips=2** and see what happens:

For Cross Entropy model, the loss significantly reduces to 4.46. Also, the accuracy significantly increases to **34.1%**.
For NCE model, the loss goes down to as low as 1.01. However, the accuracy decreases to 33.4

D) Reducing the batch size to 64 and skip window=1 and num_skips=2:

For cross Entropy model, the loss significantly reduces to 3.76. And the **accuracy shoots up to as high as 35.6%. Best performance of my model till now.**

For NCE model for this configuration, the loss comes around to be 1.38 and the the accuracy goes up to **35.1%, the best for NCE model** yet.

We conclude this improvement, to the fact that since the number of words to skip is less, same input is used more times to train and generate the batch, hence the overall accuracy is improved.

E) Changing the value of ‘k’ Negative samples and evaluating the effect.

Increasing **Doubling** the number of k negative samples to **128**, I **did not observe any further change** in accuracy. It remained the same as previous.

3. <u>TOP 20 words similar to {first,american,would}</u> <u>(In order most to least similar)</u>

Cross Entropy:

First:
('last', 'second', 'next', 'original', 'final', 'following', 'same', 'third', 'diamondbacks', 'latter', 'entire', 'best', 'throughout', 'main', 'previous', 'gollancz', 'current', 'fatwa', 'phu', 'auditions')
American:
('british', 'german', 'french', 'italian', 'canadian', 'european', 'russian', 'japanese', 'english', 'international', 'irish', 'local', 'australian', 'soviet', 'scottish', 'austrian', 'greek', 'ancient', 'discreet', 'grote')
Would:
('could', 'will', 'can', 'must', 'should', 'might', 'may', 'did', 'cannot', 'does', 'do', 'had', 'ezo', 'we', 'although', 'makes', 'though', 'to', 'came', 'took')

NCE:

First:
('term', 'american', 'before', 'where', 'word', 'against', 'about', 'early', 'at', 'since', 'including', 'how', 'however', 'most', 'all', 'until', 'abuse', 'pierre', 'english', 'civil')
American:
('french', 'english', 'century', 'pierre', 'including', 'might', 'western', 'de', 'revolution', 'civil', 'where', 'how', 'term', 'kropotkin', 'about', 'against', 'especially', 'spirit', 'before', 'since')
Would:
('will', 'could', 'did', 'he', 'can', 'does', 'de', 'might', 'century', 'should', 't', 'no', 'called', 'were', 'known', 'they', 'had', 'kropotkin', 'pierre', 'th')

The similarities I noticed are that most of the similar words returned are similar in tone rather than the context. Also cross entropy gives more similar words.

4.) NCE METHOD LOSS JUSTIFICATION (BASED ON NCE PAPER SECTION 3.1)

Noise contrastive estimation method is a method for fitting unnormalised models where, instead of the whole vocabulary we take some ‘k’ sample negative words and model our distribution based on that. So, basically, we need to train our classifier such that it is able to distinguish between samples from the data and noise distribution. This kind of behaviour makes the NCE model independent of vocabulary size (a very big advantage). We get the accuracy that is on par to methods like cross entropy but for very less computations.

Justification of the method (derivation of the formula):
Suppose, if we want to learn the distribution of words for some context. Now we need to create a binary classification problem with positive samples and k negative samples as training data. Negative samples are k times more frequent.
Then,

$$\mathbf{P\ h\ (D = 1|w, \theta) = P\ h\ \theta\ (w)\ /\ (\ P\ h\ \theta\ (w) + kPn(w)) = \sigma\ (\Delta s\theta(w, h)),}$$

Where, σ is the logistic function and $\Delta s\theta(w, h) = s\theta(w, h) - \log(kPn(w)$ is the difference between probability distribution of positive samples target words and the negative ones.
Hence we are (kind of) taking an unnormalised model and recovering a perfectly normalised one
This is the final loss equation that we get,

$$\mathbf{\partial/\partial\theta\ J\ h,w(\theta) = (1 - \sigma(\Delta s\theta(w, h)))\ \partial/\partial\theta\ \log P\ h\ \theta\ (w) - \textstyle\sum\ (i=1...k)\ [\sigma\ (\Delta s\theta(xi\ , h))\ \partial/\partial\theta\ \log P\ h\ \theta(xi)]}$$

The main difference between this and cross entropy is the k negative samples that we take. As we increase the value of k and it reaches to vocabulary size, it becomes the same as cross entropy.
Due to these k samples, we are able to reduce computation cost without compromising accuracy and that is the biggest advantage of NCE over cross entropy where we consider whole vocabulary size and the computation cost is more.

Even in our implementation, we get the NCE loss values to be around 1.3 on average whereas for cross entropy it is 4.8 on average. Add to that the fact that the NCE ran faster and includes less computational cost, why NCE is such as popular model for word embeddings.
NCE shares similarities with non probabilistic models.
The final loss equation of NCE is:

$$\mathbf{J(\theta, Batch) = \textstyle\sum\ (wo..wc)\in Batch - [\log P\ r(D = 1, wo|wc) + \textstyle\sum\ x\in V\text{^}k\ \log(1 - P\ r(D = 1, wx|wc))}$$

where,

$$\mathbf{P\ r(D = 1, wo|wc) = \sigma\ (s(wo, wc) - \log\ [kP\ r(wo)])}$$
$$\mathbf{P\ r(D = 1, wx|wc) = \sigma\ (s(wx, wc) - \log\ [kP\ r(wx)])}$$
$$\mathbf{\sigma(x) = 1/(1 + e\text{^}{-x})\ and\ s(wo, wc) = (uc\text{^}T\ uo) + bo}$$

Explanation of the loss equation:
For each input word (denoted by uo), we apply sigmoid function and then compute the logistic function of its difference with the k times unigram probability. Remember to add a bias term when computing ucuo. This bias will keep on adjusting in each step.
Then, finally, for each input word, we take k input words, compute their sum and add that to the prob. Distribution of each word and take the log. And that’s how we calculate the loss function.
Main difference to the cross entropy function is the bias term and the k negative sample we add.