QUES I

Max_num_steps -

- Gives the number of times the model is trained on a minibatch.
- Keeping the max_num_steps too low (<10000) underfits and model and the loss doesn't get to its minimum.
- Keeping max_num_steps too high overfits the data. Although the loss might be less than before, but the model will not give a high score in the word analogy task.
- It is best to stop training the model once there is no significant decrease in score over a large number of steps. This value comes between 50,000 to 1,80,000 depending on other params.
- Also, a high max_num_steps increases the training time of the model.

Batch_size -

- This given the number of training examples in one batch on which the model is trained. After every batch the gradient descent makes a step towards the minima.
- If the batch_size is too small (or tends towards 1) the gradient descent will take too many large steps in arbitrary direction before it converges. Thus it will take an arbitrary path before converging.
- If the batch_size is too large , gradient descent will make very small steps but in the correct direction. However, it may require more epochs to reach the minima.
- Often, a mid value works best (around 2^7 - 2^8)

Skip_window -

- Tells how many words to consider left and right of the center word.
- If skip_window is too low say 1 then only relation of adjacent words would be captured in the dataset.
- As we more far away from the center word, the relation between words decreases. Therefore, a very high skip_window is not likely to be very helpful.

Num_skips -

- This tells the no. of pairs where a particular word will be the center word.
- Making num_skips high will cause repetition of the same word in the dataset too many times and may cause the model to overfit.
- Low num_skips may not generate enough data for the algorithm to train or may not capture all the necessary relations from the context.

Num_samples -

- This tells the number of words used to negative sample a word. This parameter is only used in NCE and not in cross entropy, because in cross entropy we use the entire vocabulary to get the probability distribution of the output word.

- Making num_samples very low will cause the NCE loss function to not fully capture the meaning of the word, as it will have very new negative samples to compare the output word against.
- Making num_samples very high of close to vocabulary size will make the training very slow and will defeat the very purpose for which NCE was created.

Learning_rate

- This is a scalar quantity which decides how big steps gradient descent takes towards the minima.
- Making alpha (learning rate) too small will make the training slow as it will take very small steps.
- Making alpha too large can make the gradient descent skip over the minima and oscillate around it, unable to converge.

QUES 2

CROSS ENTROPY MODEL						
batch_size	skip_window	num_skips	max_num sampled	Learning rate	Average Loss	Word Analogy Score
128	4	8	200000	1.0	3.72	32.6
128	4	8	70000	1.0	3.78	33.2
128	2	4	70000	1.0	4.72	33.7
192	2	4	50000	1.0	5.11	33.3
192	2	4	70000	0.5	5.10	33.2
64	2	4	50000	1.0	4.06	33.0
126	3	6	50000	1.0	4.78	33.4

Trends and Learnings -

1. The loss converges before 200000 epocs. After a certain point it loss rate becomes very low and also starts increasing sometimes in an irregular fashion.
2. Decreasing the number of skips and skip window although increases the average loss, but give a better score on word analogy task.
3. Around 128 batch size works best. Reducing doesn't work well, increasing also doesn't help much.

4. Learning rate and max_num sampled go hand in hand. If you are reducing learning rate u need to increase the no. of epochs.
5. Lower loss does not mean you will get a better score. Below a certain point lower loss means the model is overfitting which gives a bad score.

Noise Contrastive Estimation						
batch_size	skip_window	num_skips	num sampled	Learning rate	Average Loss	Word Analogy Score
128	4	8	200000	1.0	1.17	31.5
256	2	4	70000	0.5	1.22	33.5
128	4	4	150000	1.0	1.42	34.0
128	1	2	50000	1.0	1.02	33.4
64	2	4	80000	1.0	2.58	32.9
128	2	4	1200000	1.0	1.39	33.7
192	2	4	65000	1.0	1.25	33.8

Trends and Learning -

1. The loss converges before 200000 epocs. NCE coverges slower than cross entropy although. Therefore around 150000 epochs gives a better result.
2. Decreasing the number of skips and skip window although increases the average loss, but give a better score on word analogy task.
3. Around 128 and silghtly obove that work fine. Reducing doesn't work well.
4. Lower loss does not mean you will get a better score. Below a certain point lower loss means the model is overfitting which gives a bad score. Often lossed around 1.0 - 1.1 gave bad results.
5. NCE is faster i.e takes less time per epoch. The reason being it samples only k element instead of vocab size.

QUES 3) TOP 20 WOrds

	Cross Entropy			NCE		
	First	American	Would	First	American	Would
1	last	german	could	work	about	i
2	same	british	will	however	how	t
3	best	english	can	while	war	see
4	name	french	did	book	where	did
5	original	italian	must	war	century	called
6	most	canadian	may	during	so	through
7	end	international	might	before	since	american
8	following	european	does	all	human	will
9	second	russian	should	early	western	so
10	main	irish	to	term	at	only
11	latter	reject	had	most	th	links
12	largest	itself	do	where	during	until
13	next	its	we	king	might	could
14	until	ancient	said	until	before	using
15	entire	eu	seems	history	later	joseph
16	work	controlled	made	at	would	where
17	final	cidade	argued	since	modern	might
18	current	official	under	century	more	since
19	because	spanish	has	law	others	than
20	music	follow	been	time	law	more

QUES 4) NCE SUMMARY

1. **Problem with Cross Entropy loss function**

Predicting the next word given a context word is just like predicting a class in a multi-class classification problem. The output neurons of a cross entropy loss function represent a normalized probability as their output.
To calculate such a probability we need to divide the probability of output and context word occurring together by the sum of probability of all words in vocabulary occurring together with the target word.
Thus the complexity to calculate output for one word is of the order O(V) where |V| = vocabulary size. Therefore, this method gets very slow for large vocabulary sizes.

2. **The idea of negative sampling**

To reduce the computation cost of cross entropy loss function we come up with the idea of negative sampling. The main idea is that instead of calculating the actual probability distributions of the output pair, we will now just learn to distinguish between a true pair and a false pair. Therefore, basically reducing a multi classification problem to probabilistic binary classification
We can take k random words from the vocabulary and pair them with the context word. Now these k pairs form up the 'noise' pairs which our model needs to distinguish the true pairs from.

Thus, the complexity for a single output has reduced from O(V) to O(k) , where k is the number of negative samples we take.

3. **The details of NCE**

In NCE we use unigram distribution of the training data as the noise distribution, which has been experimentally verified to work well with language models.

$$p(D, w|c) = \begin{cases} (k/1+k) * p^k & (D=0) \\ (1/1+k) * p(w|c) & (D=1) \end{cases}$$

Here, D=1 part shows that a single sample is drawn from the true distribution .
And D=0 part indicates that the k number of samples are drawn from noise distribution.

Considering, the D=1 part,

$$\begin{aligned} p(D=1, w|c) &= p(w|c) / (p(w|c) + k * p^k) \\ &= 1/(1 + k * p^k / p(w|c)) \\ &= \sigma(\log p(w|c) - \log(k * p^k)) = \sigma(\Delta s) \end{aligned}$$

In order to maximize the probability of a word coming from the true distribution, we need to maximize the, delta s part .
This means our classifier needs to discriminate words from noisy and real distributions.

Similarly, we can calculate the probability for (D==0) and combining both the eqn for (D=0 and D=1) we get the final equation for total loss J.

$$J(theta, batch) = -\sum(log(P(D=1, w_c|w) + \sum log(P(D=0, w_x|w))$$

$$J(theta, batch) = -\sum(log(P(D=1, w_c|w) + \sum log(1 - P(D=1, w_x|w))$$

References

https://www.cs.toronto.edu/~amnih/papers/wordreps.pdf
https://blog.zakjost.com/post/nce-intro/