CSE 538 Natural Language Processing HW1

Student: Paras Avkirkar/112685014

1. Hyper-Parameters Explored

Batch-Size:
The tuned model converged faster or slower depending on the batch-size. Changing batch-size affected average loss computed.

Skip-Window/Num-skips:
The similarity between the words will majorly get affected by the window-size. Since the window-size determines the type of batches that are going to get generated, changes in the window-size changes the underlying probability distribution of the center-word and outer word pairs.. If we increase the window size then the center word is going to impact more neighbors. If we decrease the window size then the center word would not be influencing words at longer distances in the context.

Learning Rate:
A small value of learning rate would mean that the model will take time to converge, but the loss may decrease monotonically. A higher value of learning rate means the updates to the parameters in the model may make the loss overshoot.

Embedding size:
It determines the richness of latent features that the model is going to learn on a given probability distribution of words. A large embedding size may introduce sparsity in vectors while gaining richness in the model, while a smaller embedding size may not capture specific semantics. Mikolov in 'Distributed Representation of Words and Phrases.' showed that embedding size of 300 performs well on data-set in order of millions

Number of steps:
If the number of steps are large and enough for a given learning rate then the model may converge sooner. But this means for the residual steps the model will keep moving around minima. If the number of steps are small and not enough for a given learning rate then the model will not converge at optimal minima.

Number of negative samples (k):
Negative sampling through NCE treats the objective of word prediction as Binary Logistic Classification where we try to maximize the log-likelihood of true target word against a center word and minimizes the log-likelihood of noise target word against a center word. Less number of negative samples would increase training time vs more negative samples. But the value of 'k' must be decided on the basis of size of data-set. Mikolov recommends value 'k' in order of 5-20 for smaller data-sets and 2-5 for large data-sets.

2. Experiments/Model Tuning

Citation:

1. https://stats.stackexchange.com/questions/140811/how-large-should-the-batch-size-be-for-stochastic-gradient-descent
2. https://stackoverflow.com/questions/22272370/word2vec-effect-of-window-size-used
3. https://arxiv.org/pdf/1310.4546.pdf (Distributed Representation of Words and Phrases)

While tuning, we tweaked one parameter at a time assuming independence between the rest of the parameters (except skip_window, num_skips). In the end, the best model was decided based by those values which were giving better overall accuracy in each parameter.

Baseline model with Cross-Entropy Loss function:
Trained on default parameters:

batch_size=128 skip_window=4 num_skips=8 num_sampled=64 max_num_steps=400001 learning_rate=1.0

Performance:

Least illustrative accuracy: 30.3 %, Most illustrative accuracy: 32.6 %, Overall Accuracy: 31.5%

Since the pre-trained model provided was trained with Noise Contrastive Estimation loss. **We trained Cross entropy loss based model by training from Scratch without loading pre-trained model**
Following are the experiments:

Experiment #	**Hyper parameters**	**Least**	**Most**	**Overall**
1	batch_size=64	30.9	31.9	31.4
Observations	The tuned model converged sooner than the baseline model. Although analogy accuracy came close to Base-line model. Sample nearest words were not close to each other. Example: digit word like ‘zero’ doesn’t get close to other digit words ‘one’, ‘two’, ‘three’ ,etc			
2	batch_size=256	**31.6**	**31.6**	**31.6**
Observations	The tuned model converged slower than the one with batch_size=64 but faster than Base-Line model. Slightly perform better than Base-Line			
3	skip_window=2 num_skips=4	28.2	33.7	31.0
Observations	Sample nearest words were poor than base-line model as expected when window size is reduced. Since the distributional probability on a center word changes drastically from its neighbours			
4	skip_window=8 num_skips=16	**31.5**	**31.6**	**31.6**

Observations	Sample nearest words were improved. E.g: All articles in the sample ‘a’, ‘an’, ‘the’ were close to each other			
5	learning_rate=2.0	**34.2**	**32.3**	**33.3**
Observations	After increasing learning_rate loss keeps overshooting between 4.8x and 4.9x and doesn’t stabilize. Nearest words logged are completely non-deterministic: For e.g: neighbours of word ‘six ‘-> ‘mcdonagh’, ‘randomized’ It seems the model somehow over-fitted as evident from high accuracy			
6 **Best Model**	embedding_size=256, num_steps=900001	**30.3**	**32.2**	**31.2**
Observations	Nearest words logged are somewhat closer: E.g: numbers group -> zero, six, eight, nine Prepositions group -> in, under, on			

Baseline model with Noise-Contrastive Estimation Loss:

Trained on default parameters:

batch_size=128 skip_window=4 num_skips=8 num_sampled=64 max_num_steps=200001 learning_rate=1.0

Performance:

Least illustrative accuracy: 32.4 %, Most illustrative accuracy: 35.7 %, Overall Accuracy: 34.0%

Tuned on pre-trained model to test tuned parameters (except embedding size)

Experiment #	**Hyper parameters**	**Least**	**Most**	**Overall**
1	batch_size = 32	29.8	31.4	30.6
Observation	Accuracy is pathetic as evident by loss which did not converge around 1.x			
2	skip_window=2 num_skips=4	31.8	35.3	33.6
Observation	A smaller window size impacted heavily more on least illustrative accuracy as compared to Base-Line model. Because of change in underlying distribution of center-context word pairs due to window-size			
3	skip_window=8 num_skips=16	**32.2**	**34.4**	**33.3**
Observation	Increase in window size improved accuracy for Most illustrative pairs than base-line model but did not impacted much in Overall accuracy			

4	num_sampled=32	33.3	35.1	34.2
Observation	Lowering value of k reduce training time and slightly improved overall-accuracy than base-line			
5	num_sampled=8	**32.5**	**35.8**	**34.1**
Observation	Loss converged at 0.21x but accuracy heavily improved. May be the model over-fitted.			
6	num_steps=400001	32.2	33.3	34.4
Observation	Loss converged at 1.41 (same as base-line model) Increasing steps, didn't help much			
8 **Best-Model**	batch_size = 128 skip_window = 8 num_skips = 16 num_sampled = 8 max_num_steps = 200001 learning_rate = 1.0	**32.6**	**36.5**	**34.6**
Observation	A combination of better performing values for each parameter, improved overall-model			

3. **Top 20 Nearest words (Excluding Self)**

Cross-Entropy

Query word	**Neighbor words**
first	**most**, same, an, spit, a, **second**, metallic, microphone, neuropathologist, attendant, fingernail, **last**, veered, federalist, vacate, paixhans, zora, next, hopewell, regulatory
american	**french**, **english**, maharaj, **hollander**, focal, montanari, **british**, concatenation, bocce, cheaper, ecusa, berlinguer, forking, atomists, **australian**, monogamous, basket, infractions, musaf, mathrm
would	**will**, **may**, **can**, **could**, **had**, **was**, **did**, were, is, does, **should** must, might, has, are, agitator, pele, garr, communicator, unbuilt

1. first -> most, second, last **(first is kind of related to opposite of last, or close to word 'second')**
2. american -> french, english, hollander, british, australian **(nationality is the concept)**
3. would -> will, may, can, could, had, was, did, should **(verbs close to would)**

NCE

Query word	**NCE Top 20 nearest words (excluding self)**
first	**name**, **last**, following, during, same, original, of, **second**, end, **most**, book, after, until, united, th, city, was, **beginning**, **best**,
american	**german, british, italian**, d, **russian**, understood, its, **european**, **irish**, **actor**, might, december, heterodox, **war**, **player**, **english**, **canadian**, international, autres, writer
would	**will, could**, said, india, must, we, do, does, did, you, they, not, who, families, if, may, so, i, believed, even

1. first -> name, last **(first-name, last-name)**
2. american -> french, english, hollander, british, australian **(nationality is the concept)**
3. would -> will, may, can, could, had, was, did, should **(verbs close to would)**

4. Summary: Noise Contrastive Estimation

1. During calculation of likelihood for a given target word ‘w’ and context word ‘h’,
   $P_{\Theta}^{h}(w) = exp(s\_\Theta(w,h)/\sum' \left(exp\left(s_{\Theta}(w',h)\right)\right)$,
   here the term in denominator we need to calculate over entire vocabulary size which is computationally expensive
2. Noise Contrastive Estimation transforms a neural language modeling problem into probabilistic binary classification problem**. The intuition is to train a classifier to learn to distinguish between a target word taken from noise distribution against a target word taken from true underlying probability distribution for a given center word.**
3. The major advantage of using NCE is that we don’t need to normalize using $\sum \; exp\left(s_{\Theta}(w,h)\right)$ for each word in our vocabulary. Instead, we just need to use $exp\left(s_{\Theta}(w,h)\right)$ for a given context and target word for a set of k-negative samples. In this manner we reduce our training time
4. The likelihood function we try to maximize is $P^{h}(D=1|w,\Theta)$, probability that for a given input word ‘h’, the target word ‘w’ is taken from a true distribution. Also, since we are negative sampling for k negative samples, we need to maximize k * $P^{h}(D=0|w,\Theta)$ :: **the probability that for a given input word ‘h’ , the target word ‘w’ was taken from a noise distribution, here ‘k’ is the number of negative samples we pick from noise distribution**
5. The final maximum likelihood function we will optimize is
   a. $J^{h}(\Theta) = E_t[logP^{h}(D=1|,w,\Theta)] + kE_n[logP^{h}(D=0|w,\Theta)]$
   Here**, E_t is expectation over true distribution and E_n is expectation over noise distribution**
   As usual we take logarithm over probability of true/noise distribution to avoid dealing with products of small infinitesimal values in likelihood function.
   So, now we will maximize our **expectation over log-likelihood function**
6. The intuition behind above log likelihood function is to maximize the probability that a target word w for a context word h is taken from true distribution and the probability chosen k negative target words taken from noise distribution