SI630 HW2 – SANGHO EUM

Task 1

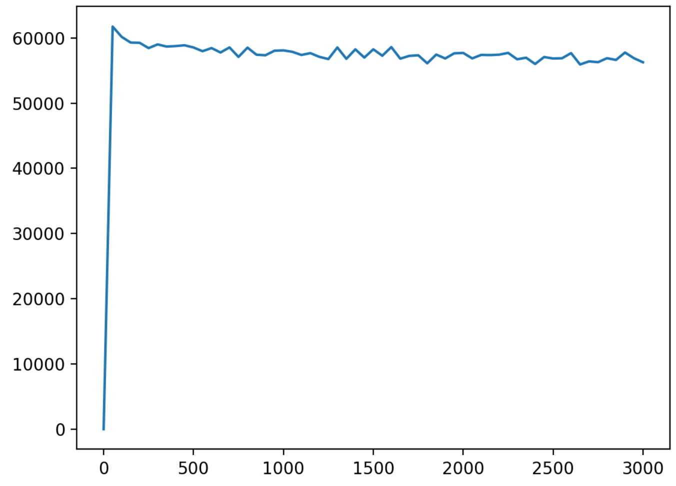
Problem 1. Modify function negativeSampleTable to create the negative sampling table.
-> Refer to the code.

Problem 2. Modify function performDescent() to implement gradient descent. Note that above this function is the line: @jit(nopython=True). This can speed up the gradient de- scent by up to 3x but requires that your implementation uses a subset of data types and shouldn't make function calls to other parts of your code (except sigmoid() because it has also been jit-compiled). You don't have to use @jit and if this complicates your code too much, you can remove it.
-> Refer to the code.

Problem 3. Modify function loadData to remove stop-words from the source data input.
-> Refer to the code.

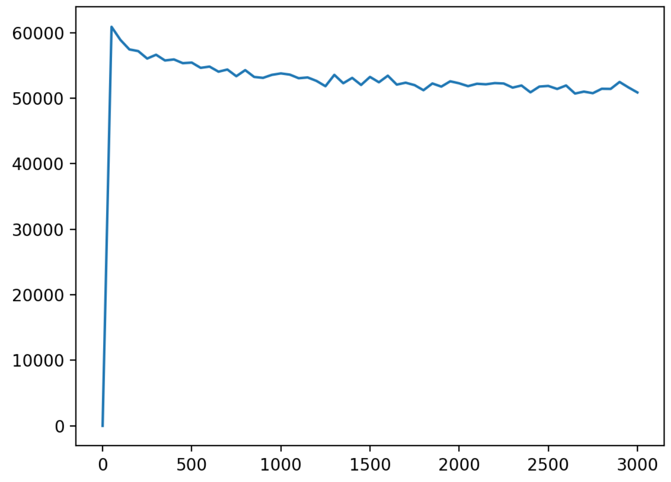
Problem 4. Modify function loadData to convert all words with less than min count oc- currences into <UNK> tokens. Modify function trainer to avoid cases where <UNK> is the input token.
-> Refer to the code.

Problem 5. Train your model with this context window and the default parameters specified at the start. After every 10,000 iterations, store the cumulative negative log-likelihood (Equation 1), reset the cumulative value to 0 and repeat. Plot these results against iteration number.

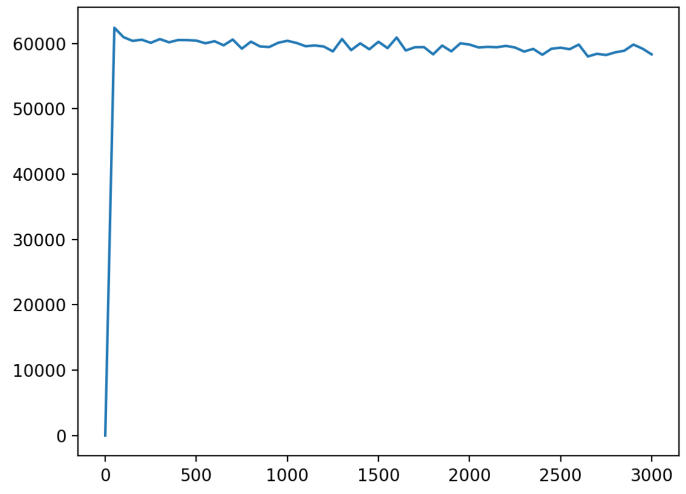


Problem 6. Re-train your network using the context windows of C = [−4, −3, −2, −1] (the four words preceding the input word) and again on the window C = [1, 2, 3, 4] (the four words after the input word). Repeat the analysis of Problem 5 with these context windows. What do you observe?

(1) C = [-4, -3, -2, -1]



(2) C= [1, 2, 3, 4]



-> I don't see much difference from Problem.5 when I used C=[1, 2, 3, 4], but negative likelihood is lower when I used C=[-4, -3, -2, -1]

Task 3

Problem 7. Modify function prediction() so that it takes one argument, the target word, and computes the top 10 most similar words based on absolute cosine similarity.
-> Refer to the code.

Problem 8. Separately preload the models trained in Problem 6 by commenting out the train vectors() line and uncommenting the load model() line. Modify the file names in load model() according to what you named them in Problem 6.
-> Refer to the code.

Problem 9. Repeat the steps in Problem 8 but using the word vectors trained in Problem 5. This output file should be called p9 output.txt.
-> Refer to the code.

Problem 10. Qualitatively looking at the topmost similar words for each target word, do these predicted word seem to be semantically similar to the target word? Describe what you see in 2-3 sentences.
-> In the case of C = [-4, -3, -2, -1], a couple of words seem to be similar, and this case has the lowest negative likelihood.

Task 4

Problem 11. For each word pair in the intrinsic-test.tsv file, create a new file con- taining their cosine similarity according to your model and the pair's instance ID. Upload these predictions to the Kaggle task associated with intrinsic similarity.

-> I got 0.24886.