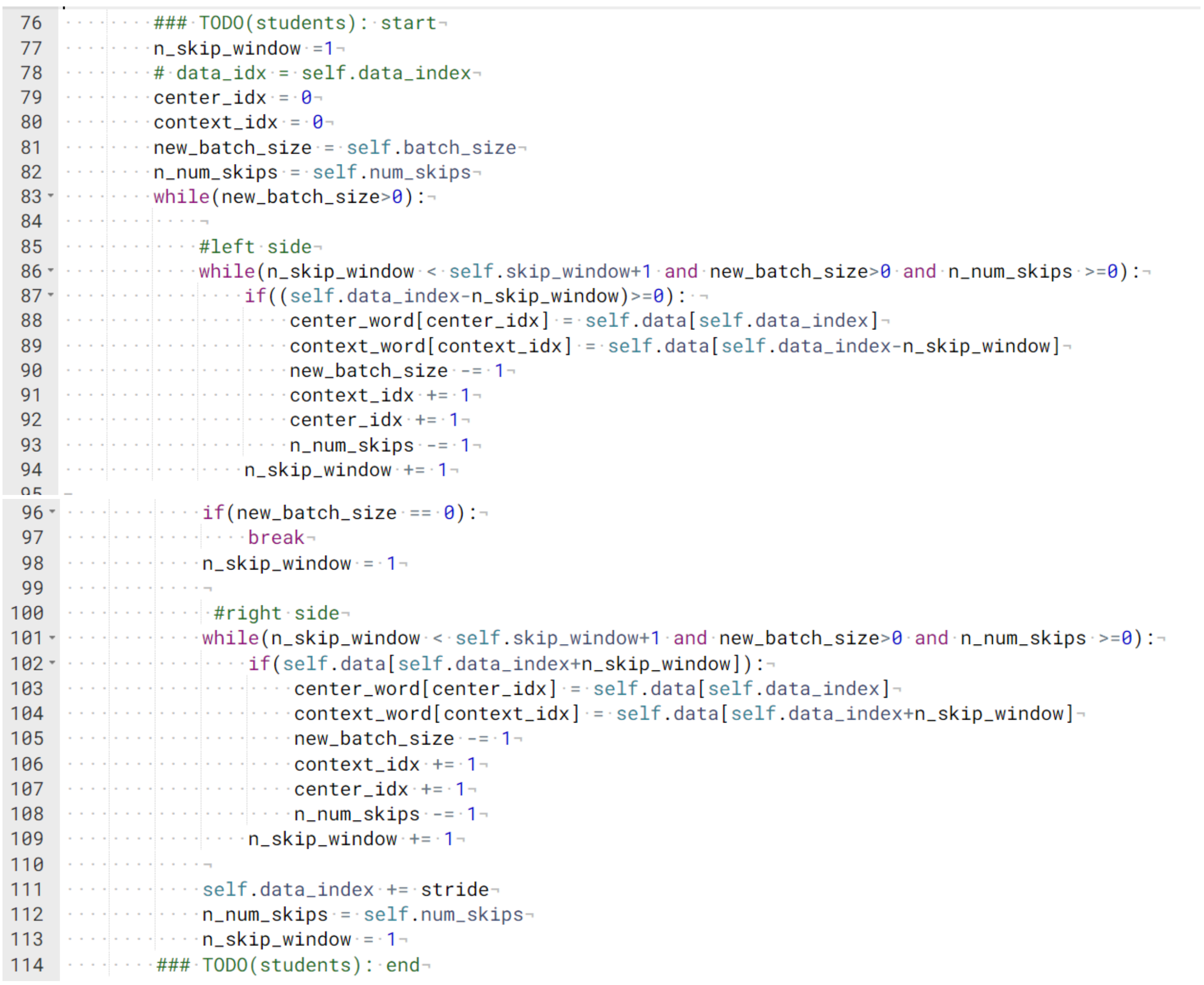
# CSE 354 Assignment 1 Report

JeongYoon Lee (114133199)

March 9, 2022

## Part 1. Explanation of my code implementations for each TO-DO tasks

- 'data.py'



```python
        ### TODO(students): start
        n_skip_window =1
        # data_idx = self.data_index
        center_idx = 0
        context_idx = 0
        new_batch_size = self.batch_size
        n_num_skips = self.num_skips
        while(new_batch_size>0):
            
            #left side
            while(n_skip_window < self.skip_window+1 and new_batch_size>0 and n_num_skips >=0):
                if((self.data_index-n_skip_window)>=0): 
                    center_word[center_idx] = self.data[self.data_index]
                    context_word[context_idx] = self.data[self.data_index-n_skip_window]
                    new_batch_size -= 1
                    context_idx += 1
                    center_idx += 1
                    n_num_skips -= 1
                n_skip_window += 1

            if(new_batch_size == 0):
                break
            n_skip_window = 1
            
             #right side
            while(n_skip_window < self.skip_window+1 and new_batch_size>0 and n_num_skips >=0):
                if(self.data[self.data_index+n_skip_window]):
                    center_word[center_idx] = self.data[self.data_index]
                    context_word[context_idx] = self.data[self.data_index+n_skip_window]
                    new_batch_size -= 1
                    context_idx += 1
                    center_idx += 1
                    n_num_skips -= 1
                n_skip_window += 1
            
            self.data_index += stride
            n_num_skips = self.num_skips
            n_skip_window = 1
        ### TODO(students): end
```

This image is my generate_batch function in data.py.

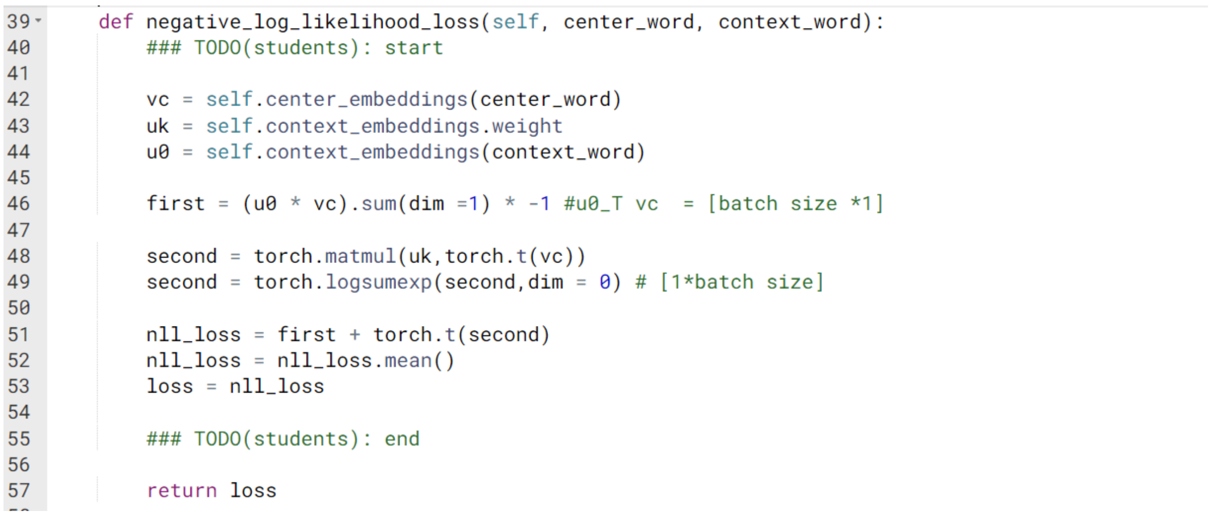
I divided into two parts, which is left side of center word, and right side of center word based on some parameters, for example, skip_window, batch_size, num_skips, data_index and stride.

So, I add the center_word from self.data[self.data_index], and I added adjacent from the center words into the context_word.

If there is a word in the index corresponding to the size of skip_window on both sides based on 'center_word', it is inserted into 'context_word'. The iteration stops when iteration number reaches to given batch_size, and the average of the loss values for each batch is returned.

One batch will be finished if the size of num_skip and batch_size is same. After one batch is finished, we have to set the new center word for the new batch, so we set self.data_index again by adjusting this with stride.

- ‘model.py’



```python
    def negative_log_likelihood_loss(self, center_word, context_word):
        ### TODO(students): start

        vc = self.center_embeddings(center_word)
        uk = self.context_embeddings.weight
        u0 = self.context_embeddings(context_word)

        first = (u0 * vc).sum(dim =1) * -1 #u0_T vc  = [batch size *1]

        second = torch.matmul(uk,torch.t(vc))
        second = torch.logsumexp(second,dim = 0) # [1*batch size]

        nll_loss = first + torch.t(second)
        nll_loss = nll_loss.mean()
        loss = nll_loss

        ### TODO(students): end

        return loss
```

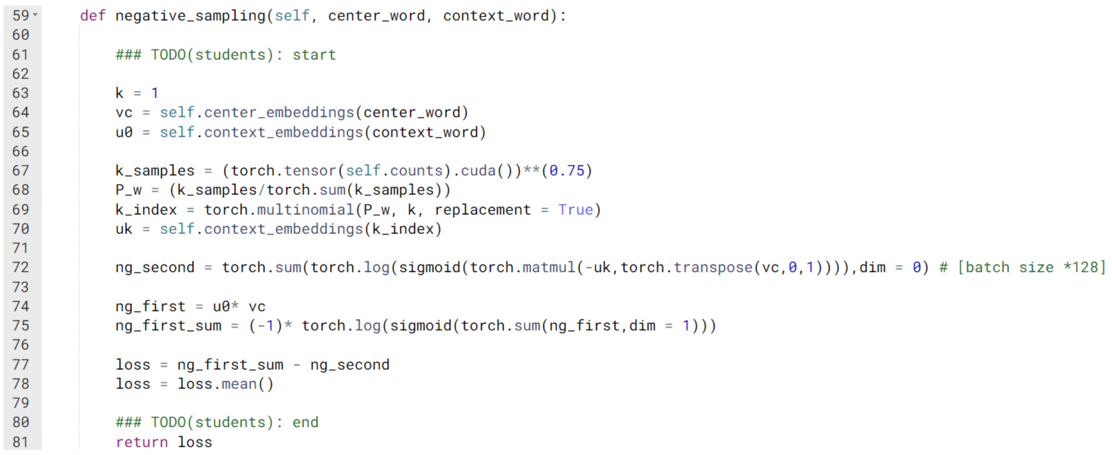
This image is my negative_log_likelihood_loss function in model.py. I used vector calculation instead of using for loop to get more faster model.

$$NLL_{batch}(\Theta) = \sum_{(w_c,w_o)\in batch} \left( -u_o^T v_c + ln \sum_{w_k \in Vocab} exp(u_k^T v_c) \right)$$

From this formula, I set my variable vc to v_c, u0 to u_0, uk to u_k in this equation.

So, my ‘first’ variable is same with this $-u_o^T v_c$ part, and my ‘second’ variable is same with $ln \sum_{w_k \in Vocab} exp(u_k^T v_c)$ this part.

The nll_loss will be the size of batch_size, so I average them and get one loss value for this one loss function. This loss value will help me to find best model between other models when doing tuning.



```python
    def negative_sampling(self, center_word, context_word):

        ### TODO(students): start

        k = 1
        vc = self.center_embeddings(center_word)
        u0 = self.context_embeddings(context_word)

        k_samples = (torch.tensor(self.counts).cuda())**(0.75)
        P_w = (k_samples/torch.sum(k_samples))
        k_index = torch.multinomial(P_w, k, replacement = True)
        uk = self.context_embeddings(k_index)

        ng_second = torch.sum(torch.log(sigmoid(torch.matmul(-uk,torch.transpose(vc,0,1)))),dim = 0) # [batch size *128]

        ng_first = u0* vc
        ng_first_sum = (-1)* torch.log(sigmoid(torch.sum(ng_first,dim = 1)))

        loss = ng_first_sum - ng_second
        loss = loss.mean()

        ### TODO(students): end
        return loss
```

This image is my negative_sampling function in model.py. I also used vector calculation instead of using for loop to get more faster model.

$$-\log\sigma(u_{c-m+j}^{T}\cdot v_c)-\sum_{k=1}^{K}\log\sigma(-\tilde{u}_k^{T}\cdot v_c)$$

From this formula, I do the same thing for setting my variable vc to v_c, u0 to u_(c-m+j), uk to u_k in this equation.

So, my 'ng_first_sum' variable is same with $-\log\sigma(u_{c-m+j}^{T}\cdot v_c)$ this part, and my 'ng_second' variable is $-\sum_{k=1}^{K}\log\sigma(-\tilde{u}_k^{T}\cdot v_c)$ same with this part.

And I average each batch's loss value as I did in the nll model, and return the loss value.

Also, I tried to make Unigram Model to get more values using power of 3/4. This will make this model for better approximation.

## Part 2. Brief description of the Hyper-parameters explored for each of the loss functions.

### - Negative Log likelihood loss

The Negative Log likelihood is used to minimize the negative log likelihood of our data which means the mean squared error between the observed y and our prediction y'. It is tuned by batch_size ans embedding size since we should use word_embedding values for this loss function.

If you look at part 3, you can see that when the batch size is proper, not large, and if there's enough epoch, we can get good training model. The smaller size of window is making better model, and their training output seems good since we can find similar words per each center word.

### - Negative Sampling

The Negative Sampling is that it will not update for all word's weight. Since Word2Vec use softmax after output layer's calculation, the computation is too large. So, it pick some words and calculate probability.

So, we do sampling with some negative examples from noise distribution which is sorted by word's frequency. To get some words from this distribution, we use hyperparameter K.

Then we can set the number of negative samples with K, and then train the way that minimizes the probability that these negative sample will be an output.

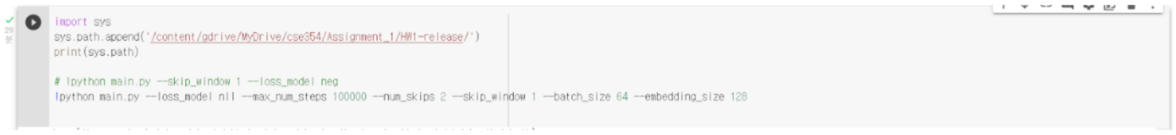
You can find if K value is high, the loss are getting higher. It depends on size of whole dataset, but in our data, K=1 is the best model for negative sampling model. Also, same with negative log likelihood model, the smaller window size is better than large size.

## Part 3. Discuss the hyperparameters experimented and how the average loss varied across each combination.

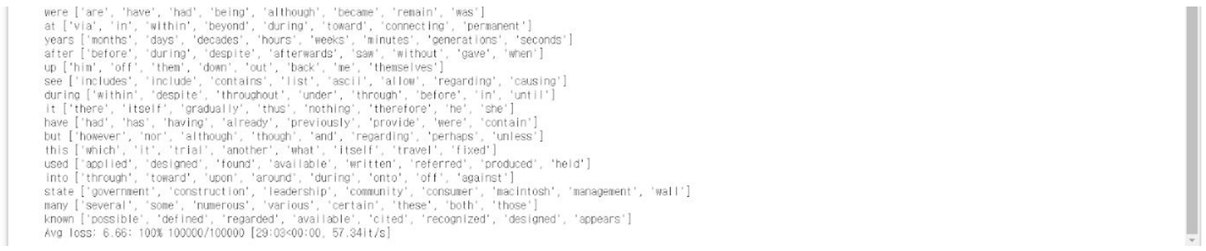
- **Negative Log likelihood loss**

  **1)** Max_num_steps: 100000, num_skips: 2, skip_window: 1, batch_size : 64, Embedding Size : 128 → **Best NLL Model!**

  >>> Avg_loss = 6.66



```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
!python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 128
```



```
were ['are', 'have', 'had', 'being', 'although', 'became', 'remain', 'was']
at ['via', 'in', 'within', 'beyond', 'during', 'toward', 'connecting', 'permanent']
years ['months', 'days', 'decades', 'hours', 'weeks', 'minutes', 'generations', 'seconds']
after ['before', 'during', 'despite', 'afterwards', 'saw', 'without', 'gave', 'when']
up ['him', 'off', 'them', 'down', 'out', 'back', 'me', 'themselves']
see ['includes', 'include', 'contains', 'list', 'ascii', 'allow', 'regarding', 'causing']
during ['within', 'despite', 'throughout', 'under', 'through', 'before', 'in', 'until']
it ['there', 'itself', 'gradually', 'thus', 'nothing', 'therefore', 'he', 'she']
have ['had', 'has', 'having', 'already', 'previously', 'provide', 'were', 'contain']
but ['however', 'nor', 'although', 'though', 'and', 'regarding', 'perhaps', 'unless']
this ['which', 'it', 'trial', 'another', 'what', 'itself', 'travel', 'fixed']
used ['applied', 'designed', 'found', 'available', 'written', 'referred', 'produced', 'held']
into ['through', 'toward', 'upon', 'around', 'during', 'onto', 'off', 'against']
state ['government', 'construction', 'leadership', 'community', 'consumer', 'macintosh', 'management', 'wall']
many ['several', 'some', 'numerous', 'various', 'certain', 'these', 'both', 'those']
known ['possible', 'defined', 'regarded', 'available', 'cited', 'recognized', 'designed', 'appears']
Avg loss: 6.66: 100% 100000/100000 [29:03<00:00, 57.34it/s]
```

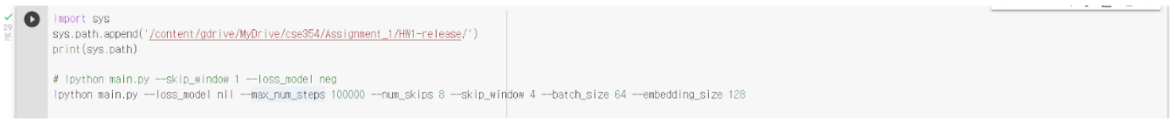
Since this nll model has smallest average loss value, so we can say this can be a best nll model among our training models. Also, the training time is smaller than other models, so we can guess this can be an optimal model. But we should look at some validation data and test data to figure out there's any other problems like overfitting.

But it seems this model trains well if you look at the output of this model which is second image. For example, for the center word 'at', we can see that there's similar words in context word like 'via' , 'in' , 'within', 'beyond' and for the center word 'many', we can find some context words like 'several', 'some', 'numerous' which is quite similar between center word and context words.

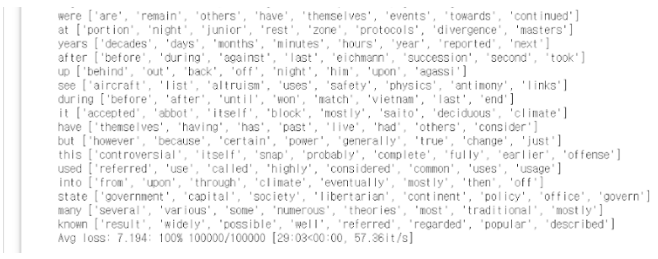
2) Max_num_steps: 100000, num_skips: 8, skip_window: 4, batch_size : 64, Embedding Size : 128

   >>> Avg_loss = 7.194

   When the size of window are large, the loss value is also high than other models.



```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
!python main.py --loss_model nll --max_num_steps 100000 --num_skips 8 --skip_window 4 --batch_size 64 --embedding_size 128
```

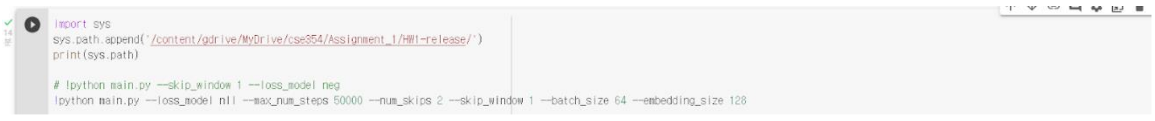


```
were ['are', 'remain', 'others', 'have', 'themselves', 'events', 'towards', 'continued']
at ['portion', 'night', 'junior', 'rest', 'zone', 'protocols', 'divergence', 'masters']
years ['decades', 'days', 'months', 'minutes', 'hours', 'year', 'reported', 'next']
after ['before', 'during', 'against', 'last', 'eichmann', 'succession', 'second', 'took']
up ['behind', 'out', 'back', 'off', 'night', 'him', 'upon', 'agassi']
see ['aircraft', 'list', 'altruism', 'uses', 'safety', 'physics', 'antimony', 'links']
during ['before', 'after', 'until', 'won', 'match', 'vietnam', 'last', 'end']
it ['accepted', 'abbot', 'itself', 'block', 'mostly', 'saito', 'deciduous', 'climate']
have ['themselves', 'having', 'has', 'past', 'live', 'had', 'others', 'consider']
but ['however', 'because', 'certain', 'power', 'generally', 'true', 'change', 'just']
this ['controversial', 'itself', 'snap', 'probably', 'complete', 'fully', 'earlier', 'offense']
used ['referred', 'use', 'called', 'highly', 'considered', 'common', 'uses', 'usage']
into ['from', 'upon', 'through', 'climate', 'eventually', 'mostly', 'then', 'off']
state ['government', 'capital', 'society', 'libertarian', 'continent', 'policy', 'office', 'govern']
many ['several', 'various', 'some', 'numerous', 'theories', 'most', 'traditional', 'mostly']
known ['result', 'widely', 'possible', 'well', 'referred', 'regarded', 'popular', 'described']
Avg loss: 7.194: 100% 100000/100000 [29:03<00:00, 57.36it/s]
```

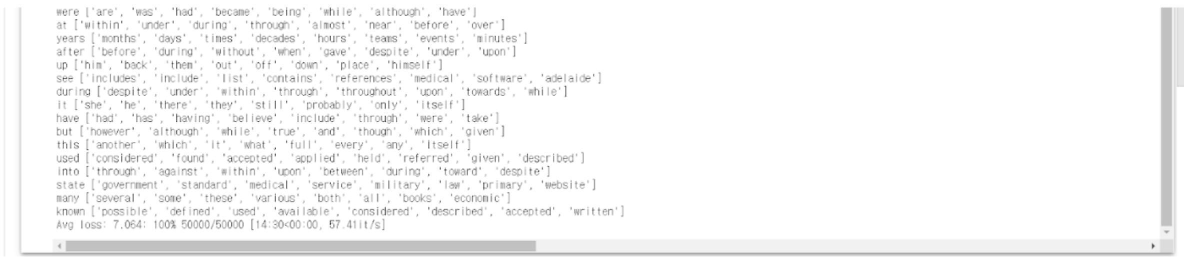
3) Max_num_steps: 50000, num_skips: 2, skip_window: 1, batch_size : 64, Embedding Size : 128

   >>> Avg_loss = 7.064

   We can find that we need more epochs to get more small loss value in this model.



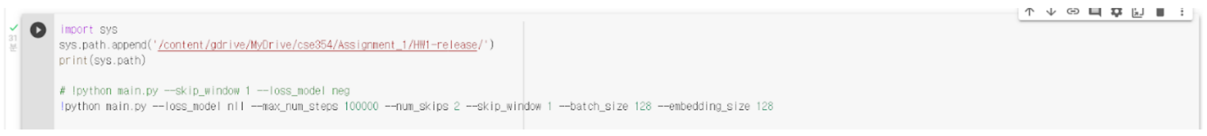
```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
!python main.py --loss_model nll --max_num_steps 50000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 128
```



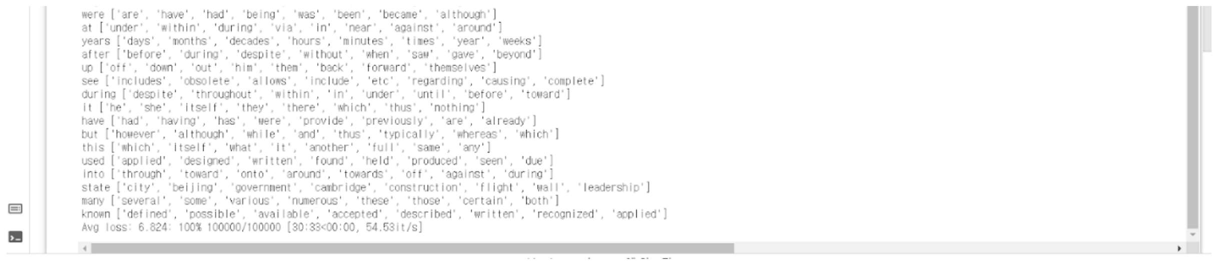
```
were ['are', 'was', 'had', 'became', 'being', 'while', 'although', 'have']
at ['within', 'under', 'during', 'through', 'almost', 'near', 'before', 'over']
years ['months', 'days', 'times', 'decades', 'hours', 'teams', 'events', 'minutes']
after ['before', 'during', 'without', 'when', 'gave', 'despite', 'under', 'upon']
up ['him', 'back', 'them', 'out', 'off', 'down', 'place', 'himself']
see ['includes', 'include', 'list', 'contains', 'references', 'medical', 'software', 'adelaide']
during ['despite', 'under', 'within', 'through', 'throughout', 'upon', 'towards', 'while']
it ['she', 'he', 'there', 'they', 'still', 'probably', 'only', 'itself']
have ['had', 'has', 'having', 'believe', 'include', 'through', 'were', 'take']
but ['however', 'although', 'while', 'true', 'and', 'though', 'which', 'given']
this ['another', 'which', 'it', 'what', 'full', 'every', 'any', 'itself']
used ['considered', 'found', 'accepted', 'applied', 'held', 'referred', 'given', 'described']
into ['through', 'against', 'within', 'upon', 'between', 'during', 'toward', 'despite']
state ['government', 'standard', 'medical', 'service', 'military', 'law', 'primary', 'website']
many ['several', 'some', 'these', 'various', 'both', 'all', 'books', 'economic']
known ['possible', 'defined', 'used', 'available', 'considered', 'described', 'accepted', 'written']
Avg loss: 7.064: 100% 50000/50000 [14:30<00:00, 57.41it/s]
```

4) Max_num_steps: 100000, num_skips: 2, skip_window: 1, batch_size : 128, Embedding Size : 128

   >>> Avg_loss = 6.824



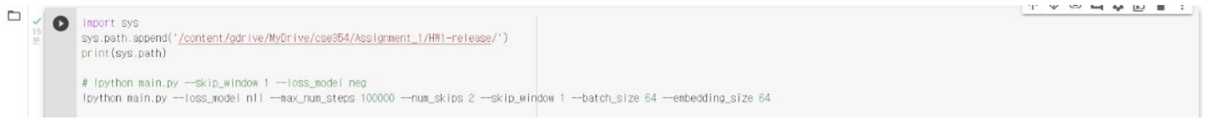
```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
!python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 128 --embedding_size 128
```



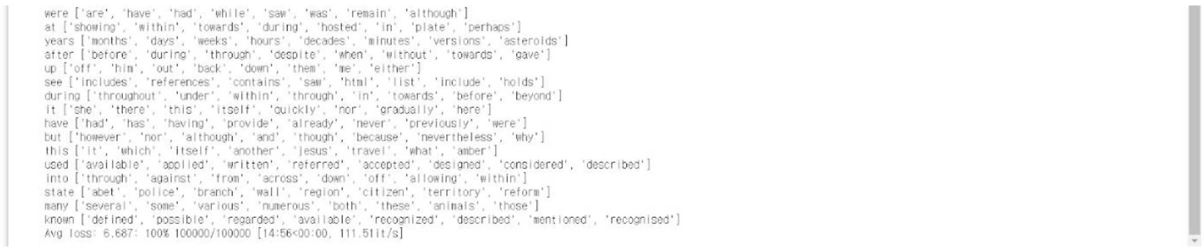
```
were ['are', 'have', 'had', 'being', 'was', 'been', 'became', 'although']
at ['under', 'within', 'during', 'via', 'in', 'near', 'against', 'around']
years ['days', 'months', 'decades', 'hours', 'minutes', 'times', 'year', 'weeks']
after ['before', 'during', 'despite', 'without', 'when', 'saw', 'gave', 'beyond']
up ['off', 'down', 'out', 'him', 'them', 'back', 'forward', 'themselves']
see ['includes', 'obsolete', 'allows', 'include', 'etc', 'regarding', 'causing', 'complete']
during ['despite', 'throughout', 'within', 'in', 'under', 'until', 'before', 'toward']
it ['he', 'she', 'itself', 'they', 'there', 'which', 'thus', 'nothing']
have ['had', 'having', 'has', 'were', 'provide', 'previously', 'are', 'already']
but ['however', 'although', 'while', 'and', 'thus', 'typically', 'whereas', 'which']
this ['which', 'itself', 'what', 'it', 'another', 'full', 'same', 'any']
used ['applied', 'designed', 'written', 'found', 'held', 'produced', 'seen', 'due']
into ['through', 'toward', 'onto', 'around', 'towards', 'off', 'against', 'during']
state ['city', 'beijing', 'government', 'cambridge', 'construction', 'flight', 'wall', 'leadership']
many ['several', 'some', 'various', 'numerous', 'these', 'those', 'certain', 'both']
known ['defined', 'possible', 'available', 'accepted', 'described', 'written', 'recognized', 'applied']
Avg loss: 6.824: 100% 100000/100000 [30:33<00:00, 54.53it/s]
```

5) Max_num_steps: 100000, num_skips: 2, skip_window: 1, batch_size : 64, Embedding Size : 64

>>> Avg_loss = 6.687



```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
!python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 64
```

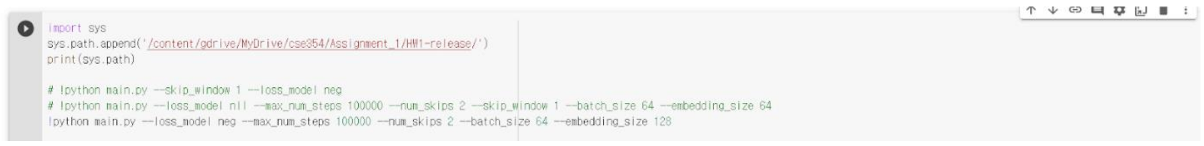


```
were ['are', 'have', 'had', 'while', 'saw', 'was', 'remain', 'although']
at ['showing', 'within', 'towards', 'during', 'hosted', 'in', 'plate', 'perhaps']
years ['months', 'days', 'weeks', 'hours', 'decades', 'minutes', 'versions', 'asteroids']
after ['before', 'during', 'through', 'despite', 'when', 'without', 'towards', 'gave']
up ['off', 'him', 'out', 'back', 'down', 'them', 'me', 'either']
see ['includes', 'references', 'contains', 'saw', 'html', 'list', 'include', 'holds']
during ['throughout', 'under', 'within', 'through', 'in', 'towards', 'before', 'beyond']
it ['she', 'there', 'this', 'itself', 'quickly', 'nor', 'gradually', 'here']
have ['had', 'has', 'having', 'provide', 'already', 'never', 'previously', 'were']
but ['however', 'nor', 'although', 'and', 'though', 'because', 'nevertheless', 'why']
this ['it', 'which', 'itself', 'another', 'jesus', 'travel', 'what', 'amber']
used ['available', 'applied', 'written', 'referred', 'accepted', 'designed', 'considered', 'described']
into ['through', 'against', 'from', 'across', 'down', 'off', 'allowing', 'within']
state ['abet', 'police', 'branch', 'wall', 'region', 'citizen', 'territory', 'reform']
many ['several', 'some', 'various', 'numerous', 'both', 'these', 'animals', 'those']
known ['defined', 'possible', 'regarded', 'available', 'recognized', 'described', 'mentioned', 'recognised']
Avg loss: 6.687: 100% 100000/100000 [14:56<00:00, 111.51it/s]
```

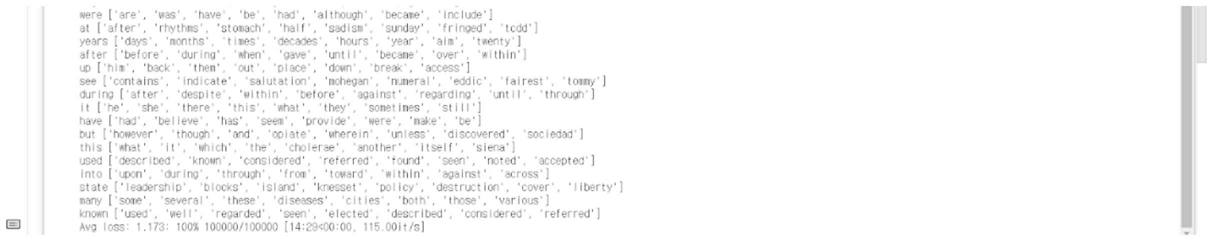
- **Negative Sampling**

**1)** Max_num_steps: 100000, num_skips: 2, batch_size : 64, Embedding Size : 128, K : 1 → **Best NEG Model!**

>>> Avg_loss = 1.173



```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
# !python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 64
!python main.py --loss_model neg --max_num_steps 100000 --num_skips 2 --batch_size 64 --embedding_size 128
```



```
were ['are', 'was', 'have', 'be', 'had', 'although', 'became', 'include']
at ['after', 'rhythms', 'stomach', 'half', 'sadism', 'sunday', 'fringed', 'tcdd']
years ['days', 'months', 'times', 'decades', 'hours', 'year', 'aim', 'twenty']
after ['before', 'during', 'when', 'gave', 'until', 'became', 'over', 'within']
up ['him', 'back', 'them', 'out', 'place', 'down', 'break', 'access']
see ['contains', 'indicate', 'salutation', 'mohegan', 'numeral', 'eddic', 'fairest', 'tommy']
during ['after', 'despite', 'within', 'before', 'against', 'regarding', 'until', 'through']
it ['he', 'she', 'there', 'this', 'what', 'they', 'sometimes', 'still']
have ['had', 'believe', 'has', 'seem', 'provide', 'were', 'make', 'be']
but ['however', 'though', 'and', 'opiate', 'wherein', 'unless', 'discovered', 'sociedad']
this ['what', 'it', 'which', 'the', 'cholerae', 'another', 'itself', 'siena']
used ['described', 'known', 'considered', 'referred', 'found', 'seen', 'noted', 'accepted']
into ['upon', 'during', 'through', 'from', 'toward', 'within', 'against', 'across']
state ['leadership', 'blocks', 'island', 'knesset', 'policy', 'destruction', 'cover', 'liberty']
many ['some', 'several', 'these', 'diseases', 'cities', 'both', 'those', 'various']
known ['used', 'well', 'regarded', 'seen', 'elected', 'described', 'considered', 'referred']
Avg loss: 1.173: 100% 100000/100000 [14:29<00:00, 115.00it/s]
```

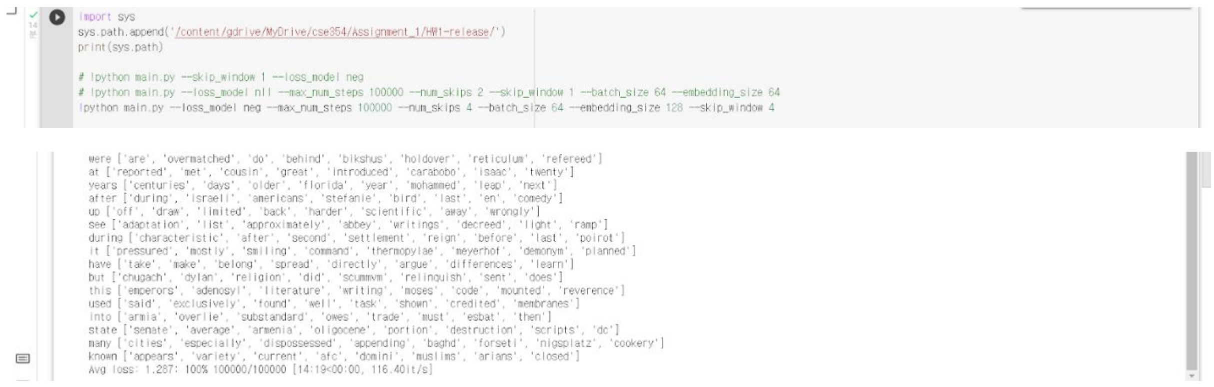
Since this neg model has smallest average loss value, so we can say this can be a best neg model among our training models. Also, the training time is not that bad compared to other models, so we can guess this can be an optimal model using negative sampling as loss function. But same as nll model, we should look at some validation data and test data to figure out if there’s any other problems like overfitting.

It also seems training well if you look at the output of this model which is second image. For example, for the center word ‘into’, we can see that there’s similar words in context word like ‘upon’ , ‘during’ , ‘through’, ‘toward’ and for the center word ‘were’, we can find some context words like ‘are’, ‘was’, ‘be’ which is quite similar between center word and context words.

But I think there are also some wrong trained words like ‘state’ with context words ‘leadership’, ‘blocks’, ‘island’. I think my best nll model’s output for ‘state’ center word is much better.

2) Max_num_steps: 100000, num_skips: 4, batch_size : 64, Embedding Size : 128, K : 1 , * skip_window : 4

>>> Avg_loss = 1.287

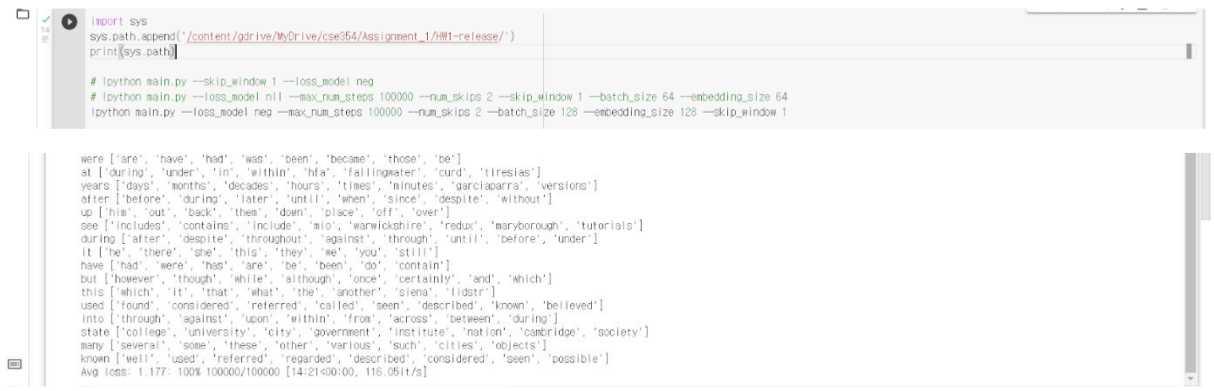


```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
# !python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 64
!python main.py --loss_model neg --max_num_steps 100000 --num_skips 4 --batch_size 64 --embedding_size 128 --skip_window 4
```

```
were ['are', 'overmatched', 'do', 'behind', 'bikshus', 'holdover', 'reticulum', 'refereed']
at ['reported', 'met', 'cousin', 'great', 'introduced', 'carabobo', 'isaac', 'twenty']
years ['centuries', 'days', 'older', 'florida', 'year', 'mohammed', 'leap', 'next']
after ['during', 'israeli', 'americans', 'stefanie', 'bird', 'last', 'en', 'comedy']
up ['off', 'draw', 'limited', 'back', 'harder', 'scientific', 'away', 'wrongly']
see ['adaptation', 'list', 'approximately', 'abbey', 'writings', 'decreed', 'light', 'ramp']
during ['characteristic', 'after', 'second', 'settlement', 'reign', 'before', 'last', 'poirot']
it ['pressured', 'mostly', 'smiling', 'command', 'thermopylae', 'meyerhof', 'demonym', 'planned']
have ['take', 'make', 'belong', 'spread', 'directly', 'argue', 'differences', 'learn']
but ['chugach', 'dylan', 'religion', 'did', 'scummvm', 'relinquish', 'sent', 'does']
this ['emperors', 'adenosyl', 'literature', 'writing', 'moses', 'code', 'mounted', 'reverence']
used ['said', 'exclusively', 'found', 'well', 'task', 'shown', 'credited', 'membranes']
into ['armia', 'overlie', 'substandard', 'owes', 'trade', 'must', 'esbat', 'then']
state ['senate', 'average', 'armenia', 'oligocene', 'portion', 'destruction', 'scripts', 'dc']
many ['cities', 'especially', 'dispossessed', 'appending', 'baghd', 'forseti', 'nigsplatz', 'cookery']
known ['appears', 'variety', 'current', 'afc', 'domini', 'muslims', 'arians', 'closed']
Avg loss: 1.287: 100% 100000/100000 [14:19<00:00, 116.40it/s]
```

3) Max_num_steps: 50000, num_skips: 2, batch_size : 128, Embedding Size : 128, K : 1

>>> Avg_loss = 1.177

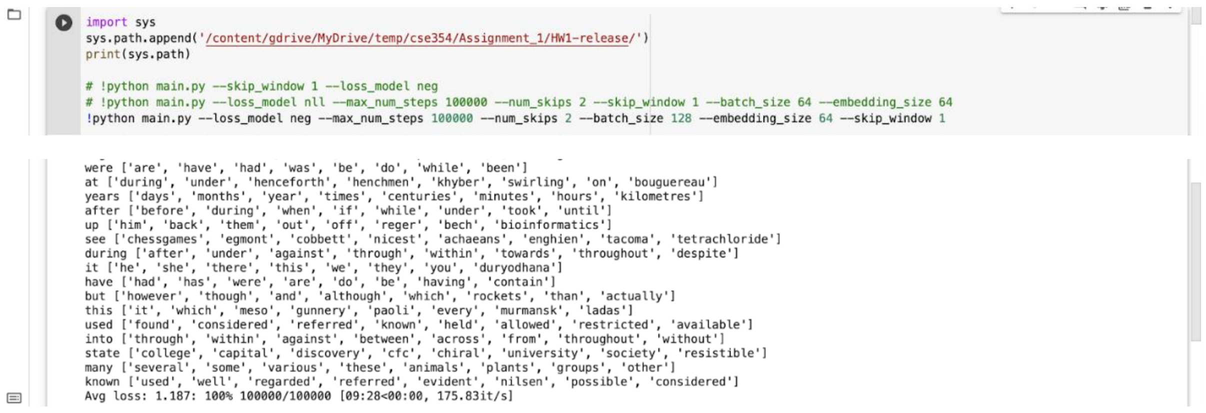


```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
# !python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 64
!python main.py --loss_model neg --max_num_steps 100000 --num_skips 2 --batch_size 128 --embedding_size 128 --skip_window 1
```

```
were ['are', 'have', 'had', 'was', 'been', 'became', 'those', 'be']
at ['during', 'under', 'in', 'within', 'hfa', 'fallingwater', 'curd', 'tiresias']
years ['days', 'months', 'decades', 'hours', 'times', 'minutes', 'garciaparra', 'versions']
after ['before', 'during', 'later', 'until', 'when', 'since', 'despite', 'without']
up ['him', 'out', 'back', 'them', 'down', 'place', 'off', 'over']
see ['includes', 'contains', 'include', 'mio', 'warwickshire', 'redux', 'maryborough', 'tutorials']
during ['after', 'despite', 'throughout', 'against', 'through', 'until', 'before', 'under']
it ['he', 'there', 'she', 'this', 'they', 'we', 'you', 'still']
have ['had', 'were', 'has', 'are', 'be', 'been', 'do', 'contain']
but ['however', 'though', 'while', 'although', 'once', 'certainly', 'and', 'which']
this ['which', 'it', 'that', 'what', 'the', 'another', 'siena', 'lidstr']
used ['found', 'considered', 'referred', 'called', 'seen', 'described', 'known', 'believed']
into ['through', 'against', 'upon', 'within', 'from', 'across', 'between', 'during']
state ['college', 'university', 'city', 'government', 'institute', 'nation', 'cambridge', 'society']
many ['several', 'some', 'these', 'other', 'various', 'such', 'cities', 'objects']
known ['well', 'used', 'referred', 'regarded', 'described', 'considered', 'seen', 'possible']
Avg loss: 1.177: 100% 100000/100000 [14:21<00:00, 116.05it/s]
```

4) Max_num_steps: 100000, num_skips: 2, batch_size : 128, Embedding Size : 64, K : 1

>>> Avg_loss = 1.187

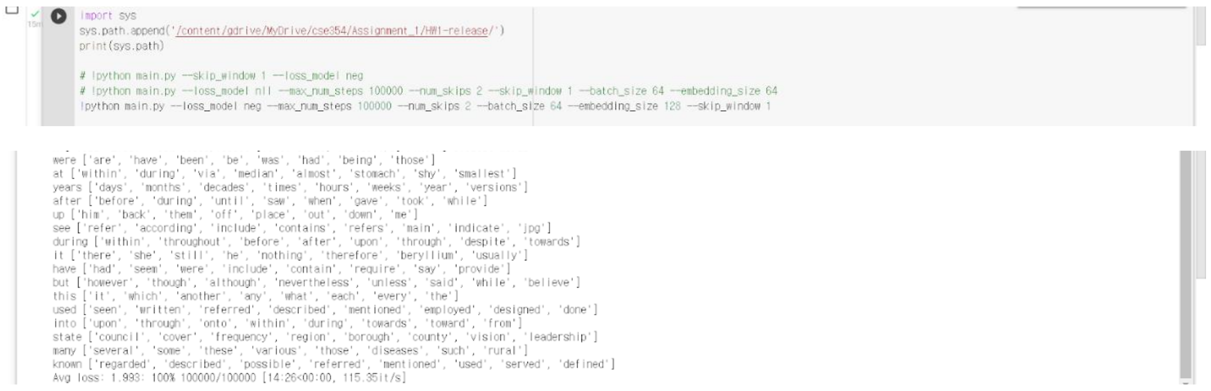


```
import sys
sys.path.append('/content/gdrive/MyDrive/temp/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
# !python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 64
!python main.py --loss_model neg --max_num_steps 100000 --num_skips 2 --batch_size 128 --embedding_size 64 --skip_window 1
```

```
were ['are', 'have', 'had', 'was', 'be', 'do', 'while', 'been']
at ['during', 'under', 'henceforth', 'henchmen', 'khyber', 'swirling', 'on', 'bouguereau']
years ['days', 'months', 'year', 'times', 'centuries', 'minutes', 'hours', 'kilometres']
after ['before', 'during', 'when', 'if', 'while', 'under', 'took', 'until']
up ['him', 'back', 'them', 'out', 'off', 'reger', 'bech', 'bioinformatics']
see ['chessgames', 'egmont', 'cobbett', 'nicest', 'achaeans', 'enghien', 'tacoma', 'tetrachloride']
during ['after', 'under', 'against', 'through', 'within', 'towards', 'throughout', 'despite']
it ['he', 'she', 'there', 'this', 'we', 'they', 'you', 'duryodhana']
have ['had', 'has', 'were', 'are', 'do', 'be', 'having', 'contain']
but ['however', 'though', 'and', 'although', 'which', 'rockets', 'than', 'actually']
this ['it', 'which', 'meso', 'gunnery', 'paoli', 'every', 'murmansk', 'ladas']
used ['found', 'considered', 'referred', 'known', 'held', 'allowed', 'restricted', 'available']
into ['through', 'within', 'against', 'between', 'across', 'from', 'throughout', 'without']
state ['college', 'capital', 'discovery', 'cfc', 'chiral', 'university', 'society', 'resistible']
many ['several', 'some', 'various', 'these', 'animals', 'plants', 'groups', 'other']
known ['used', 'well', 'regarded', 'referred', 'evident', 'nilsen', 'possible', 'considered']
Avg loss: 1.187: 100% 100000/100000 [09:28<00:00, 175.83it/s]
```

5) Max_num_steps: 100000, num_skips: 2, batch_size : 64, Embedding Size : 128, K : 3

>>> Avg_loss = 1.993



```
import sys
sys.path.append('/content/gdrive/MyDrive/cse354/Assignment_1/HW1-release/')
print(sys.path)

# !python main.py --skip_window 1 --loss_model neg
# !python main.py --loss_model nll --max_num_steps 100000 --num_skips 2 --skip_window 1 --batch_size 64 --embedding_size 64
!python main.py --loss_model neg --max_num_steps 100000 --num_skips 2 --batch_size 64 --embedding_size 128 --skip_window 1
```

```
were ['are', 'have', 'been', 'be', 'was', 'had', 'being', 'those']
at ['within', 'during', 'via', 'median', 'almost', 'stomach', 'shy', 'smallest']
years ['days', 'months', 'decades', 'times', 'hours', 'weeks', 'year', 'versions']
after ['before', 'during', 'until', 'saw', 'when', 'gave', 'took', 'while']
up ['him', 'back', 'them', 'off', 'place', 'out', 'down', 'me']
see ['refer', 'according', 'include', 'contains', 'refers', 'main', 'indicate', 'jpg']
during ['within', 'throughout', 'before', 'after', 'upon', 'through', 'despite', 'towards']
it ['there', 'she', 'still', 'he', 'nothing', 'therefore', 'beryllium', 'usually']
have ['had', 'seem', 'were', 'include', 'contain', 'require', 'say', 'provide']
but ['however', 'though', 'although', 'nevertheless', 'unless', 'said', 'while', 'believe']
this ['it', 'which', 'another', 'any', 'what', 'each', 'every', 'the']
used ['seen', 'written', 'referred', 'described', 'mentioned', 'employed', 'designed', 'done']
into ['upon', 'through', 'onto', 'within', 'during', 'towards', 'toward', 'from']
state ['council', 'cover', 'frequency', 'region', 'borough', 'county', 'vision', 'leadership']
many ['several', 'some', 'these', 'various', 'those', 'diseases', 'such', 'rural']
known ['regarded', 'described', 'possible', 'referred', 'mentioned', 'used', 'served', 'defined']
Avg loss: 1.993: 100% 100000/100000 [14:26<00:00, 115.35it/s]
```

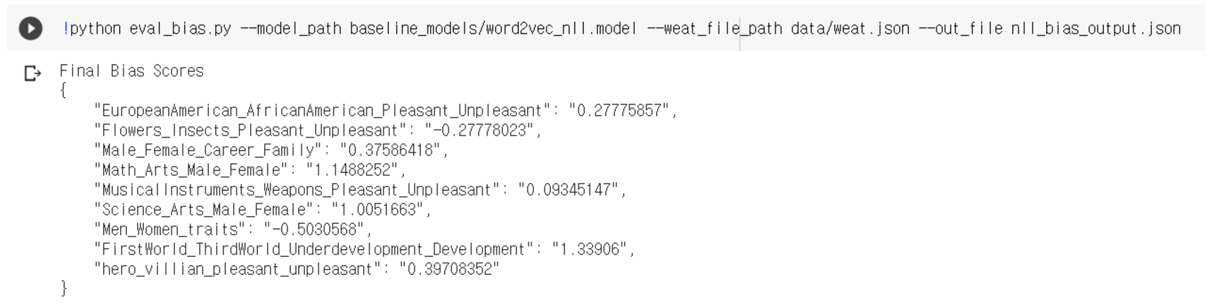
So, based on these results for each model, it would be better to train more than 100000 epochs with smaller window size, and the optimal batch size is 64, and embedding size is 128 to get the small loss value.

For the negative sampling, we can find that the model with K=1 is better than other large K values.

## Part 4. Summary of the comparison in bias scores (for the weat.json output) obtained for the best model as obtained with loss function of NLL and negative sampling. Justification for the bias score along with the attributes and targets used must be mentioned.

*WEAT Test [-2,2]

- ### Negative Log likelihood loss



```
!python eval_bias.py --model_path baseline_models/word2vec_nll.model --weat_file_path data/weat.json --out_file nll_bias_output.json
```

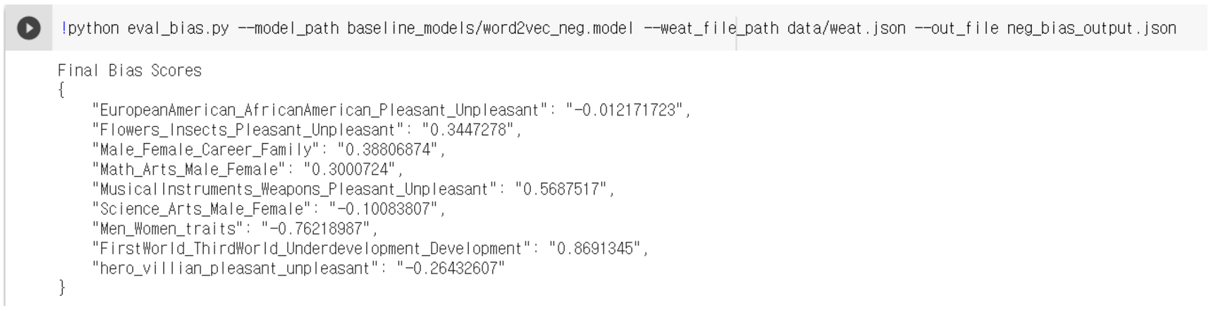
```
Final Bias Scores
{
    "EuropeanAmerican_AfricanAmerican_Pleasant_Unpleasant": "0.27775857",
    "Flowers_Insects_Pleasant_Unpleasant": "-0.27778023",
    "Male_Female_Career_Family": "0.37586418",
    "Math_Arts_Male_Female": "1.1488252",
    "MusicalInstruments_Weapons_Pleasant_Unpleasant": "0.09345147",
    "Science_Arts_Male_Female": "1.0051663",
    "Men_Women_traits": "-0.5030568",
    "FirstWorld_ThirdWorld_Underdevelopment_Development": "1.33906",
    "hero_villian_pleasant_unpleasant": "0.39708352"
}
```

Through this test, we can find some result for the 9 tests as below.

I treated if scores are less than -1 or larger than 1, they are strongly negative or strongly positive.

1. Positive (0.27775857) : European American are more associated with pleasant words
2. Negative (-0.27778023) : Insects are more associated with pleasant words
3. Positive (0.37586418) : Career words are more associated with male names.
4. Strongly Positive (1.1488252) : Make attributes are more associated with Math words.
5. Positive (0.09345147) : Musical instruments are more associated with Pleasant words.
6. Strongly Positive (1.0051663) : Science words are more associated with male attributes words.
7. Negative (-0.5030568) : Women names are more associated with positive traits words.
8. Strongly Positive (1.33906) : Development is more associated with First world words.
9. Positive (0.39708352) : Hero is more associated with pleasant words.

- ### Negative Sampling



```
!python eval_bias.py --model_path baseline_models/word2vec_neg.model --weat_file_path data/weat.json --out_file neg_bias_output.json
```

```
Final Bias Scores
{
    "EuropeanAmerican_AfricanAmerican_Pleasant_Unpleasant": "-0.012171723",
    "Flowers_Insects_Pleasant_Unpleasant": "0.3447278",
    "Male_Female_Career_Family": "0.38806874",
    "Math_Arts_Male_Female": "0.3000724",
    "MusicalInstruments_Weapons_Pleasant_Unpleasant": "0.5687517",
    "Science_Arts_Male_Female": "-0.10083807",
    "Men_Women_traits": "-0.76218987",
    "FirstWorld_ThirdWorld_Underdevelopment_Development": "0.8691345",
    "hero_villian_pleasant_unpleasant": "-0.26432607"
}
```

1. Negative (-0.012171723) : African American are more associated with pleasant words
2. Positive (0.3447278) : Flowers are more associated with pleasant words

3. Positive (0.38806874) : Career words are more associated with male names.
4. Positive (0.3000724) : Make attributes are more associated with Math words.
5. Positive (0.5687517) : Musical instruments are more associated with Pleasant words.
6. Strongly Negative (-0.10083807) : Arts words are more associated with male attributes words.
7. Negative (-0.76218987) : Men names are more associated with positive traits words.
8. Positive (0.8691345) : Development is more associated with First world words.
9. Negative (-0.26432607) : Villain is more associated with pleasant words.

For my models, through the WEAT test, it seems there are some different results for words. Especially test 6 is almost reversed between those two models.

For our normal thinking, for the test 2, flowers are usually more associated with pleasant words than insects. So, I think for this test, neg model seems better. However, for the test 9, we usually think that Hero word is more associated with pleasant words, but through the neg model, they result Villain is more associated with pleasant words.

This result can show that we need more accurate training. Since we are only look at training loss, we should consider other factors like overfitting, and preprocessing training dataset.

Many tests are same with both nll and neg models, but some tests like test 2 or test 9 result different, so it looks interesting to me. Next time, I want to try to preprocess train dataset and tune with other hyperparameters too.