BBM497: Introduction to Natural Language Processing Lab. **(Due: 01/06/2020)**

Submission Assignment #4

Instructor: Asst. Prof. Dr. Burcu CAN, Necva BÖLÜCÜ *Name:* Deniz Zağlı*, Netid:* 21527668

- **Libraries:**

  One of the libraries I use in this assignment is the json library. Since the poems were created in the dataset json format found, I used the json library to read this file. I used the numpy library to take advantage of the speed of the numpy array. I also used the Numpy library to be able to use some mathematical operations. I used the random library to get random numbers or to select random elements from a list. I also used the random library to select weighted random elements. I used the Dynet library to create a neural network model. I used the Math library to take advantage of the logarithm operation when calculating perplexity. I used the Matplotlib library to graphically visualize the resulting losses.

- **Data Structures:**

  I used a dictionary primarily as a data structure. Dataset was in JSON format. There were poems related to id and id in dataset. In this case, I created a dictionary with an id-poetry match. In short, I used dictionaries while creating the dataset. Likewise, I created a dictionary with word embeds to be a word-vector relationship. Due to the ease of operation, I used the lists when creating the necessary diagram for the trainset. While creating the language model required for the Perplexity account, I again used the dictionary data structure. I used the lists to keep data for a short time in the functions.

- **0.1 Task 1: Feed-Forward Neural Network Language Model (FNN)**

  I used the Dynet library when creating a Feed-Forward Neural Network Language Model. First, I created an empty model. Then I created the Feed-Forward Neural Network Language Model by entering the following parameters:

  Hidden_Layer = 64

  W_hx = model.add_parameters((Hidden_Layer, 50))

  b_x = model.add_parameters(Hidden_Layer)

  W_hy = model.add_parameters((size_of_unique_words, Hidden_Layer))

  b_y = model.add_parameters(size_of_unique_words)

  The number of elements formed as a result of Softmax layer is equal to the number of unique words. I calculate this with the function I specified in the code. As a trainer, I used the Simple Stochastic Gradient Descent algorithm. The Train part consists of the following stages. First of all, we shuffle the bigram train set we created for the train. Next, we look at the vector of the first word in the word embed. If the word is not embedded, a new word embed is created for that word. But with completely random weights. After this selected vector is calculated in the calculation function, softmax is sent to the layer. The loss is calculated as a result of the softmax layer. As a result of this calculated loss, the relevant weights are updated. This process takes place for all the members of the trains. The number of epoch tells us how many times we will use the train set.The loss value in each epoch is displayed as the output of the train section. An example of these loss values shown is as follows:

  Epoch: 1 Epoch Loss: 6.268589081423843

  Epoch: 2 Epoch Loss: 5.743978045124007

  Epoch: 3 Epoch Loss: 5.272123466773051

  ...

- **0.2 Task 2: Poem Generation**

  The stage of producing poetry is actually similar to the train stage. An inital value is required to produce poetry. This value is ¡s¿. After initially giving this value, it is the same as the train stage until the softmax layer. We get the probability values of words that can be the result of Softmax. Of these values, whose weighted averages are 1, we randomly select a word by weight. This word continues until it becomes ¡/s¿. This cycle continues until you create 5 poems. A sample poem consisting of an end of training with 100 sample size, 25 epoch, 64 hidden layer parameters is shown below:

Health-food folks around the sun is

With flickering meaning and a fir-clad glen far behind

And now this that gladdens home dark

Who make alone

When a pun

- **0.3 Task 3: Evaluation**

  To calculate Perplexity I did the following calculation: When we calculate the sum of the logithms in a

$$PP(W) = 2^{-\frac{1}{N}\sum_{i=1}^{N} log_2 P(w_1 w_2 \cdots w_N)}$$

Figure 1: Perplexity

way, we get a normalized value. When we calculate the sum of the logithms in a way, we get a normalized value. The perplexity of the poem I gave as an example in Task 2 is 1.0591107392111967.

- **Experiments**

  Sample Size

  My first variable is the size of our train set. I experimented with dataset sizes allowed by my hardware. The training set sizes I used for this are 10, 50, 100, 200. The graphics of Loss values and a sample poem produced in each size are shared below. Similarly, the average perplexity of the 5 poems created under each experiment is given:

  Sample Size: 10 (Figure 2)

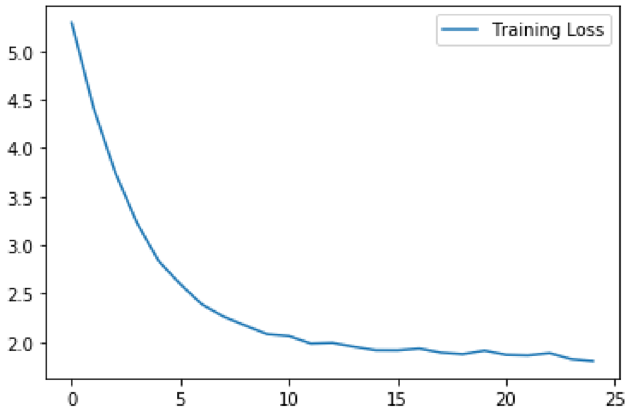

Figure 2: Sample Size: 10

Poem:

o may thy morn was the windows frost

but scarce observ'd the laws

for gold his sword the golden woodlands redden

nbsp wait here's another one chamber

but scarce observ'd the dangers gather robes

Sample Size: 50 (Figure 3)

Poem:

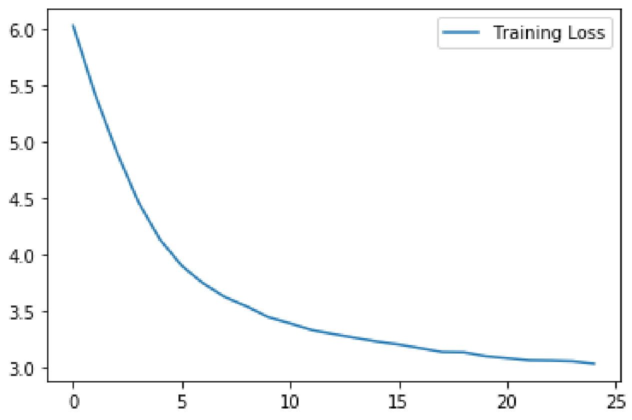

Figure 3: Sample Size: 50

white as sewer pipes burst flooding immeasurable night light

and those near

and sunwise burn your wrong

and nevermore returned nor was a bird once by thee

Sample Size: 100 (Figure 4)

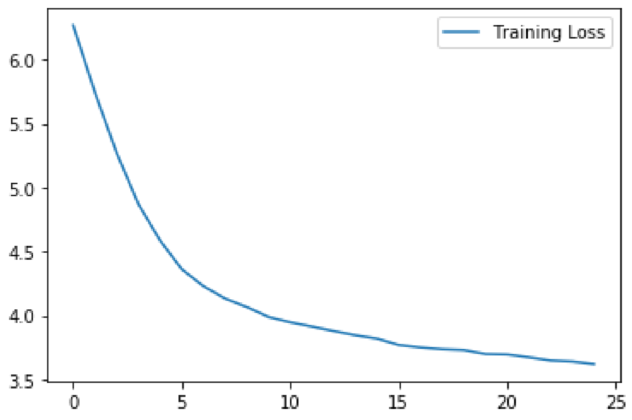

Figure 4: Sample Size: 100

Poem:

oh my shepherd lad

as fair

a fool when cruelty she tells me

flashed on the fixed stars

up

when a warm calabria's rosy mornings smile

disguising oft soul

of woe the wall

nothing really seems to rive what goth and lightly do flow

shakespeare

Sample Size: 200 (Figure 5)

Poem:

the pain

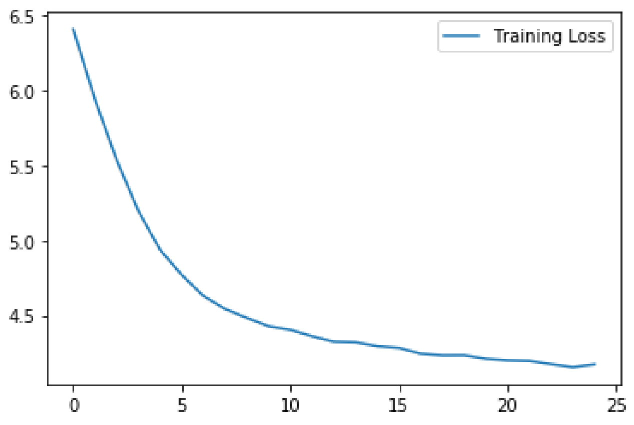

Figure 5: Sample Size: 200

beware

while waves and fall grain in the earth in the the elfin spirits pass: and that treasured counted other casualty

on the round fast

in the rich annoy the smiles him alone aloof feveredly apart

in the lips alone nerved

there be o' lovely

Epoch Size

In this part of the experiment, all variables were kept constant and only epoch was changed.

Epoch Size: 5 (Figure 6)

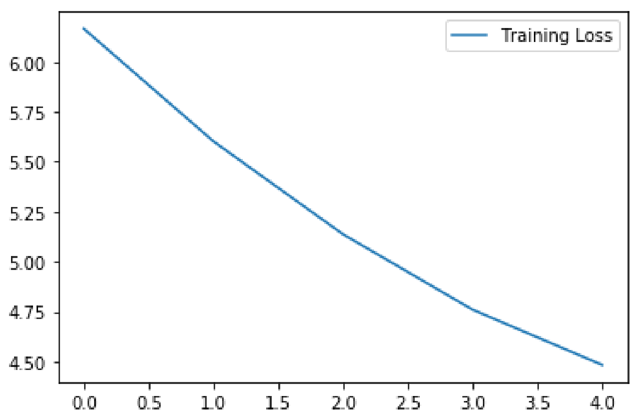

Figure 6: Epoch: 5

Poem:

their countries

called he blink

hot

ever it were to not rise falls

for not dreaming

warton

it not not took give me either i see

he head king

embrace

hoary a thousand in these same news in the dark eyes away wound i always

near

Epoch Size: 15 (Figure 7)

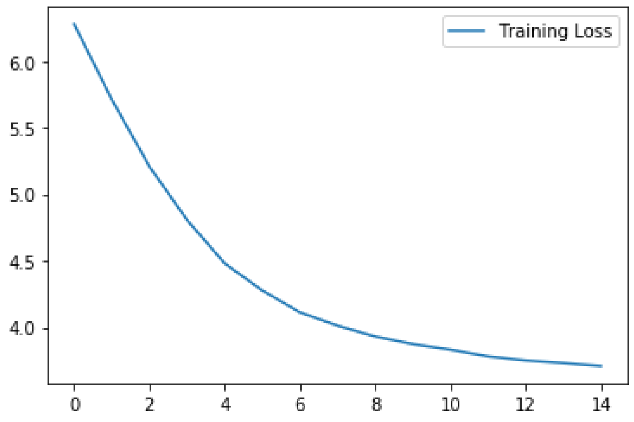

Figure 7: Epoch: 15

Poem:

we float on the first stanza added the open air of napalmand one we clomb to the desert thus

the soft from the bloody brother act v

the waves forced back to the last leafs fall

the very nimble fingers

the elm

the opening mysteries

the hall

the way a those rebell spirits adjudg'd to the great campaigner came certain centuries imperial name

the attack

Epoch Size: 20 (Figure 8)

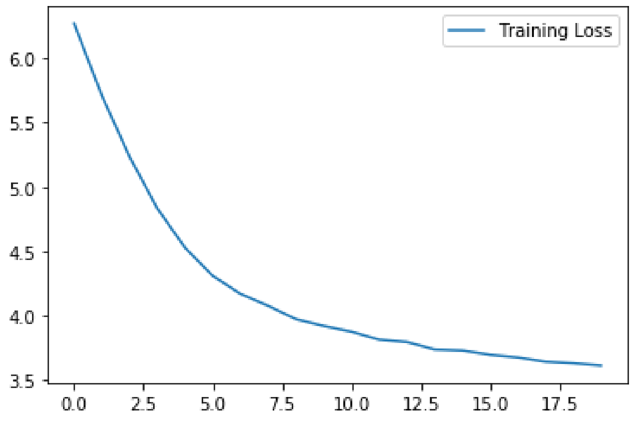

Figure 8: Epoch: 20

Poem:

o earth there's the young narcissus

be gone

then wont let

you the cuyahoga river

who cannot be thou

we quite sure have naught

who builds

who have found it they said it's

a little at life

who lived the budget the mall

Epoch Size: 25 (Figure 9)

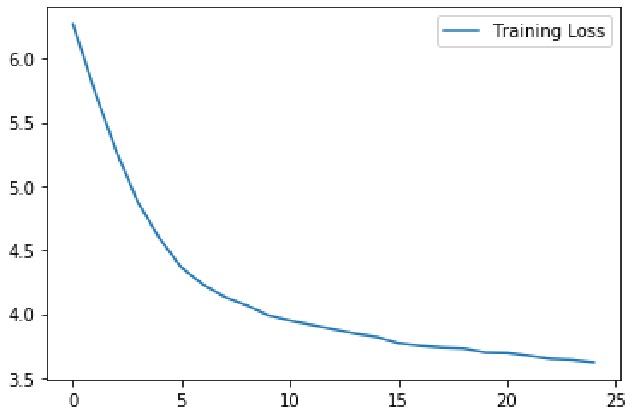

Figure 9: Epoch: 25

Poem:

oh my shepherd lad

as fair

a fool when cruelty she tells me

flashed on the fixed stars

up

when a warm calabria's rosy mornings smile

disguising oft soul

of woe the wall

nothing really seems to rive what goth and lightly do flow

shakespeare

Hidden Layer Size

In this part, I observed the effect of Hidden Layer on learning. Keeping all other parameters constant, I changed the number of Hidden Layer.

Hidden Layer: 16 (Figure 10)

Poem:

here the fair inside

into the happiness and silver

what or sides a hint reflette e aside than by destroy

che

and what dare gallogly

what for pike and sleep they suffer as don't look

boosey

Perplexity for Poem: 8.988084611133699

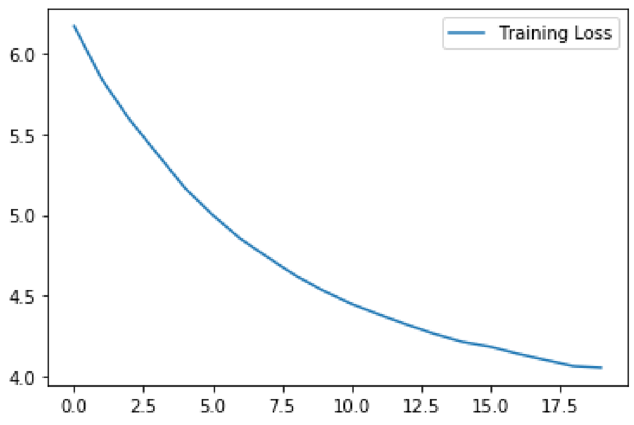

Figure 10: Hidden Layer: 16

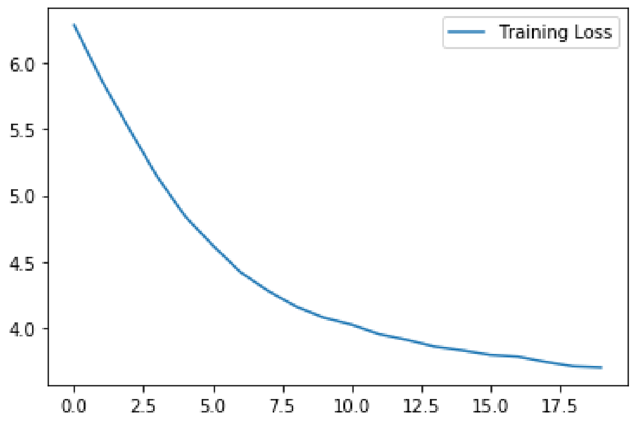

Figure 11: Hidden Layer: 32

Hidden Layer: 32 (Figure 11)

what could your apartment now i feared him it shall always
we the ground are portions of lovelessness eye
the moist matter would say it keers dark of as i fall
is not ever into their ale
(and aye to love we rays
to kept by side by side by rove
creep up from their names with black and the cream home me sunlight art my apartment now it eve
the time one exits art fairer than the picture

Perplexity for Poem: 7.875672905492629

Hidden Layer: 64 (Figure 12)

Poem:

o earth there's the young narcissus
be gone
then wont let
you the cuyahoga river
who cannot be thou
we quite sure have naught
who builds

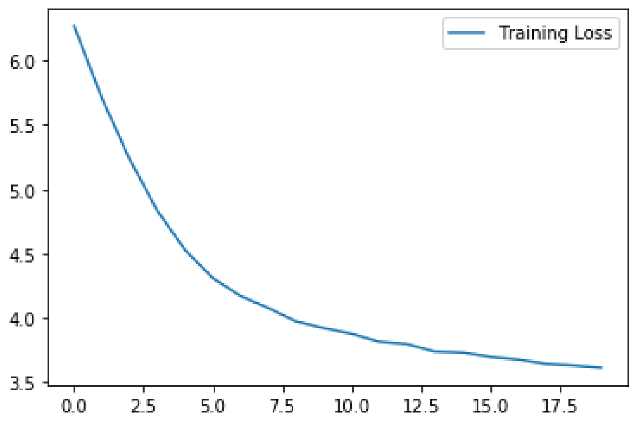

Figure 12: Hidden Layer: 64

who have found it they said it's

a little at life

who lived the budget the mall

Perplexity for Poem: 9.623146730027491

Hidden Layer: 128 (Figure 13)

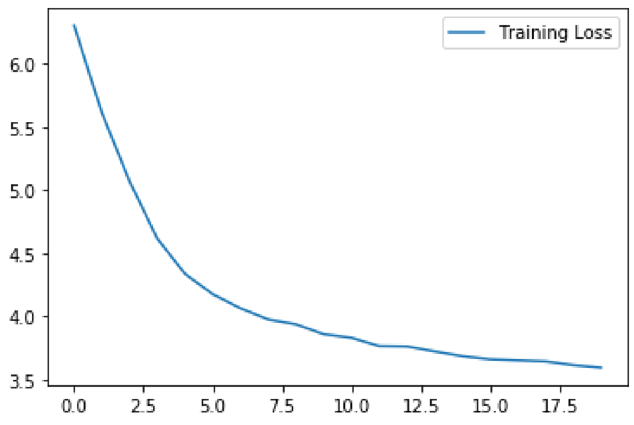

Figure 13: Hidden Layer: 128

Poem:

sal

remembering those now i measure

we ebbed with rapture crown'd

though was a trial

my rhymin'

the tablet stone

for saluting you be we part life is i seal them showed sublime

that is i curse ne'er made out

che fu mal seme per la coculla

Perplexity for Poem: 6.987586617862193

Note:

During the experiments, I found the perplexities of the first 3 values wrong due to the error in the peroplexity function. For this reason, I only calculated the perplexities of the poems in the Last Hidden Layer experiment. However, the perplexity function works correctly.

- **Experimental Results**

  If we interpret the results of the experiment, let me start with the effect of dataset size on learning. Dataset is an important element for NLP tasks. It was observed that few words were learned in low datasets. We can understand this from the resulting poem. Even if loss are low, perplexity is high. Loss is low because the number of unique words that make up the softmax layer is low. In other words, the machine guesses among fewer words. but as the dataset grows, the number of unique words to guess as a result of softmax layer increases. This means that the loss increases. But we observe more accurate poems and lower perplexity values. With the increase in dataset, the model needs to be developed. In the model I created for this project, we see that the average loss is fixed at 3. Another observed factor in experiments is the number of epoch. The effect of the epoch number on learning seems to be certain. The low number of epoch makes learning insufficient. Too much can result in overfit. As we will see in the graph, on average, we see that the loss is fixed at 3. Although I tried a maximum of 25 epoches, there was no decrease after 20 epochs. Another factor I observed in experiments is the number of hidden layers. As the number of hidden layers increases, the model gets deeper and there is an increase in learning. But there is a trade like this. As the number of hidden layers increases, the train time becomes longer. I experimented up to 128 hidden layers. During these experiments, I observed a continuous decrease in loss and an increase in learning. There is a decrease in perplexities.