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

Submission Assignment #4

Instructor: Necva BÖLÜCÜ *Name:* Ahmet Faruk BAYRAK, *Netid:* 21426716

1 Introduction

In this assignment, we were expected to implement a **text generation model** using **deep learning models (bigram level Feed-Forward Neural Network)**. We use **DyNet** deep learning library for implementing FFNN and **GloVe 6B word-embeddings** for the pretrained word vectors.

1.1 Dataset

As a dataset we use a poem dataset called **Uni-Modal Poem** which is a large poem corpus dataset that involves around 93K poems. UniM-Poem is crawled from several publicly online poetry web-sites, such as Poetry Foundation, PoetrySoup, best-poem.net andpo- ets.org.

For the pretrained word vectors, **GloVe 6B word embeddings** is used with different dimensions.

2 Tasks

2.1 Task 1: Build Feed-Forward Neural Network Language Model (FNN)

Feed-Forward Neural Network Language Model is a prediction-based neural language model. In this part we were expected to build a bigram FNN using DyNet library.

2.1.1 Pre-processing

First of all we needed to prepare dataset for training. For pre-processing I added start token (which is called ”bos” in the GloVe) to the starting of the poems and end token (which is called ”eos” in the GloVe) to the ending of the poems. By the help of these tags program knows that the starting and finishing of poems while generating a random poems.

Also in the generating part, we use newline tokens (”`\n`”) for passing a new line, I replaced ”`\n`” with ” `\n` ” to take them as a token.

After that program creates bigram pairs. Since there are almost 93k poems in the dataset it takes so much time to processes whole data. So I took parts of the dataset for training.

Finally, I read pre-trained GloVe vector embeddings and placed them to the dictionary. For **out-of-vocabulary (OOV) words** I used ”unk” (unkown words) token in the GloVe. After that I created lookup parameters on DyNet model and create **lookup table**.

2.1.2 Training

After pre-processing part is done, program starts to train the data and build a bigram FNN model. The feed-forward neural network is defined mathematically as follows:

$$y = b + U \cdot tanh(d + Hx)$$

Where:

Y: is an unnormalized log-probabilities of the unique words

d: is a hidden layer biases
b: is a output layer biases
Hx (W): is a weight matrices between input-hidden layers
U: is a weight matrices between hidden-output layers

In the training part, there are several **parameters**:

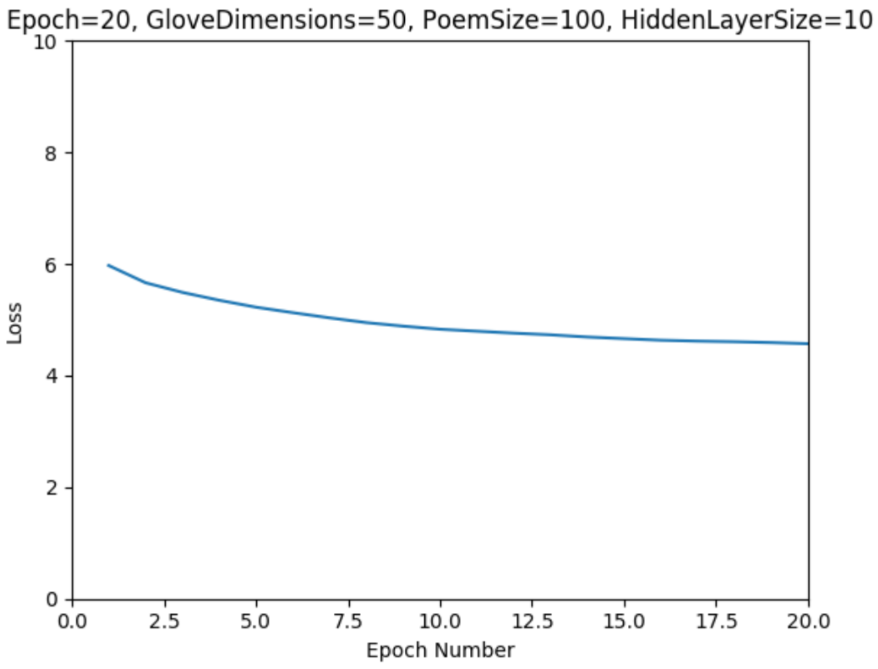
Trainer Algorithm: I selected stochastic gradient descent (SGD) method using backpropagation (BP) algorithm.
GloVe vector dimensions: There are four different dimensions in the GloVe 6B word embeddings which are 50, 100, 200, 300.
Training poem size: Since the processing of whole dataset takes too much time, I used different parts of the dataset such as 100 poems, 500 poems and 1000 poems.
Hidden Layer dimension size: I tried different hidden layer dimension size for getting better result such as 10 and 100.
Epoch number: Since our model learns doing epochs, I tried several different epoch number to determine the number of epochs needed for learning of model. According to my result minimum 5, maximum 10 epochs is enough for learning of model.
Loss Function: I used negative log likelihood (**pickneglogsoftmax** function of DyNet) as a loss function.

2.1.3 Experiments

As I mentioned before, I trained different models with different parameters. In this section I will show my loss results with parameters.

1) First Model:

Poem Size: 100 poems
GloVe Dimension: 50
Hidden Layer Dimension: 10
Epoch number: 20



My loss results:

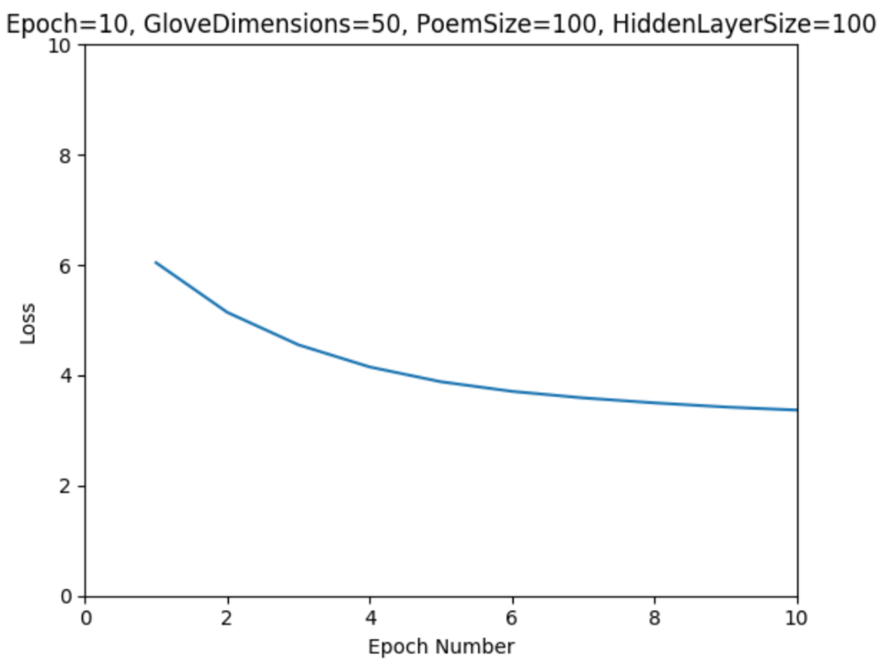
- Epoch 1. loss = 5.973706
- Epoch 2. loss = 5.664620

- Epoch 3. loss = 5.491951
- Epoch 4. loss = 5.350689
- Epoch 5. loss = 5.226334
- Epoch 6. loss = 5.128166
- Epoch 7. loss = 5.034868
- Epoch 8. loss = 4.949514
- Epoch 9. loss = 4.886314

As it shown in the loss results graph, we can say that program learns on 5 or 10 epochs. So I printed 10 epochs and tried with 10 epochs after that.

2) Second Model:

Poem Size: 100 poems
GloVe Dimension: 50
Hidden Layer Dimension: 100
Epoch number: 10



My loss results:

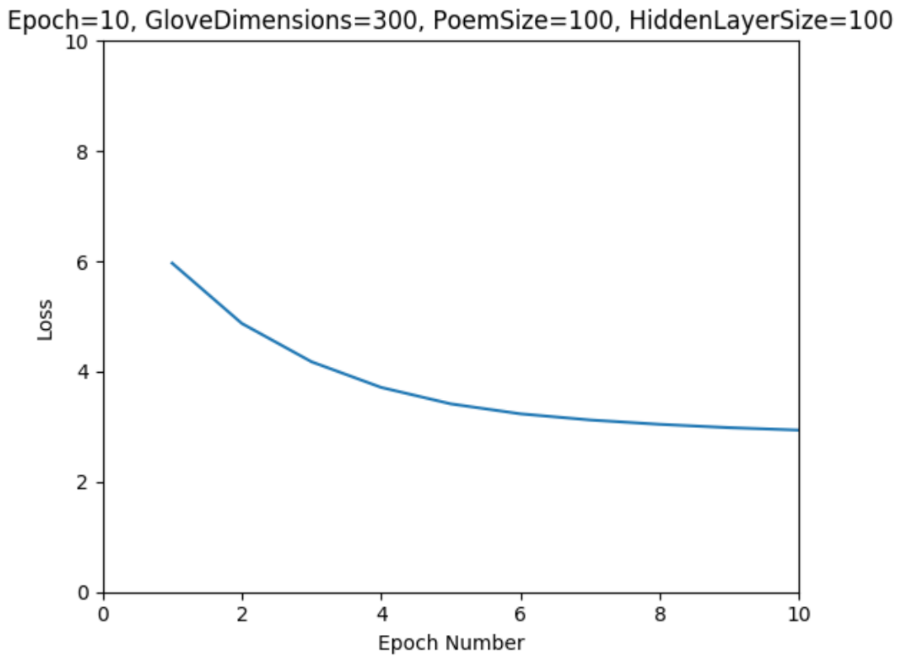
- Epoch 1. loss = 6.042409
- Epoch 2. loss = 5.141984
- Epoch 3. loss = 4.552683
- Epoch 4. loss = 4.151371
- Epoch 5. loss = 3.883204
- Epoch 6. loss = 3.708666
- Epoch 7. loss = 3.589989
- Epoch 8. loss = 3.499734

- Epoch 9. loss = 3.424644
- Epoch 10. loss = 3.368774

According to my results, increasing hidden layer dimension size decreases the loss since I only changed hidden layer dimension size between first and second model 10 to 100.

3) Third Model:

Poem Size: 100 poems
GloVe Dimension: 300
Hidden Layer Dimension: 100
Epoch number: 10



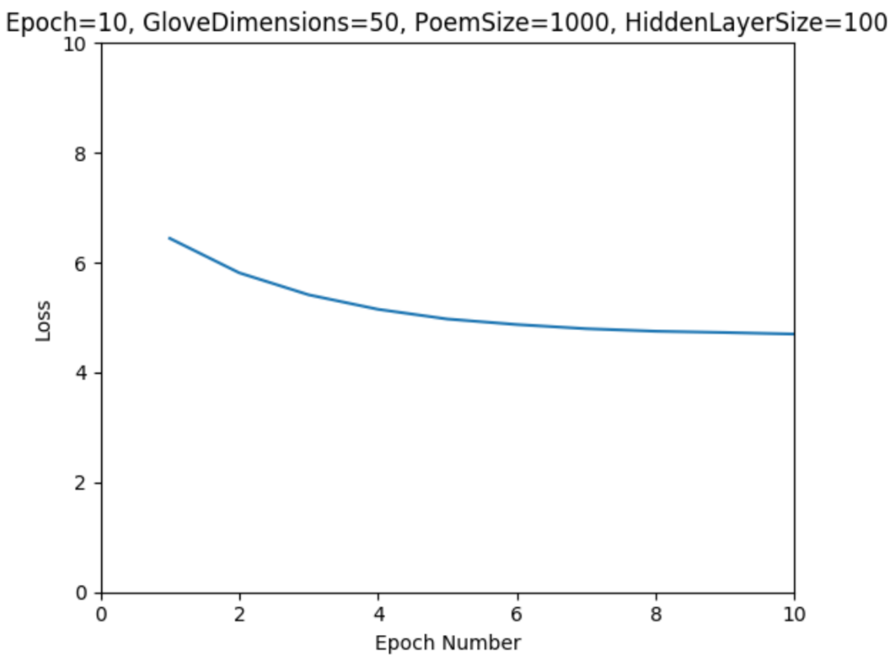
My loss results:

- Epoch 1. loss = 5.967529
- Epoch 2. loss = 4.874099
- Epoch 3. loss = 4.178724
- Epoch 4. loss = 3.713428
- Epoch 5. loss = 3.414762
- Epoch 6. loss = 3.234073
- Epoch 7. loss = 3.122787
- Epoch 8. loss = 3.043453
- Epoch 9. loss = 2.982990
- Epoch 10. loss = 2.938505

According to results, we can say that increasing dimension size of GloVe word embeddings, decreases the loss since I only changed GloVe word embeddings size 50 to 300 between second and third model.

4) Fourth Model:

Poem Size: 1000 poems
GloVe Dimension: 50
Hidden Layer Dimension: 100
Epoch number: 10



My loss results:

- Epoch 1. loss = 6.443192
- Epoch 2. loss = 5.815629
- Epoch 3. loss = 5.415357
- Epoch 4. loss = 5.150879
- Epoch 5. loss = 4.973963
- Epoch 6. loss = 4.873677
- Epoch 7. loss = 4.798096
- Epoch 8. loss = 4.751528
- Epoch 9. loss = 4.728447
- Epoch 10. loss = 4.700469

2.2 Task 2: Poem Generation and Task 3: Evaluation

In this part we were expected to generate poems using the model which we trained in the first task and calculate the perplexity of generated poems for evaluation.

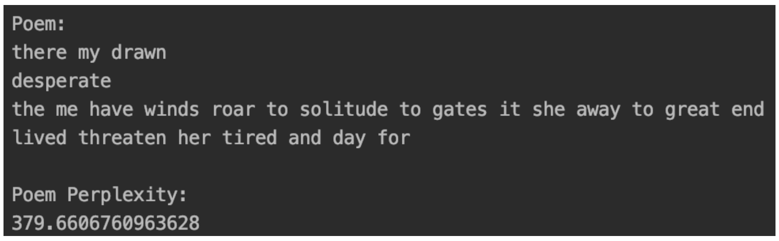
Firstly program asks for number of poems which will be generated and line number of poem from the user.

For generating a poem program starts with start token initially which is a **"bos"** in the GloVe. Then, program predicts one word at each time using the previous word and feeding this word as input to the neural network in the next time step.

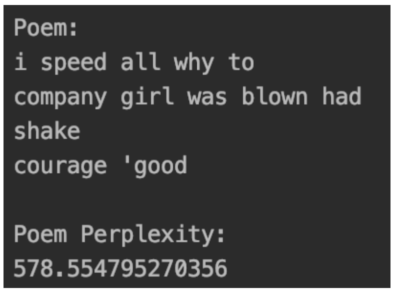
For picking next word I used random choice function of numpy library. Since by the help of softmax function, I normalized weights between zero and one. So random choice function selects next word according predictions.

I will show my generated poems with different models:

First Model:

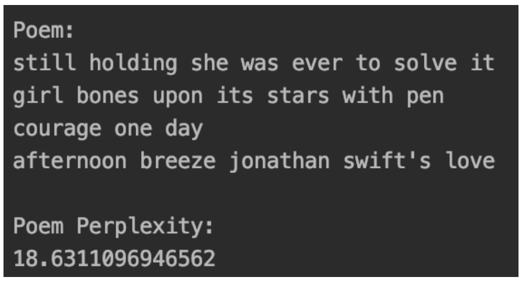


```
Poem:
there my drawn
desperate
the me have winds roar to solitude to gates it she away to great end
lived threaten her tired and day for

Poem Perplexity:
379.6606760963628
```

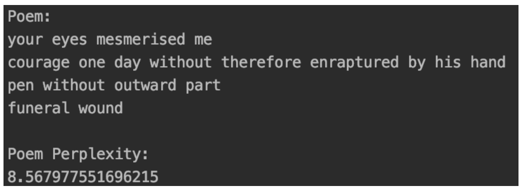


```
Poem:
i speed all why to
company girl was blown had
shake
courage 'good

Poem Perplexity:
578.554795270356
```

Second Model:

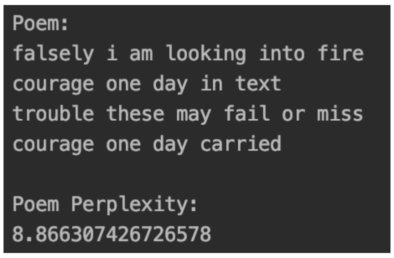


```
Poem:
still holding she was ever to solve it
girl bones upon its stars with pen
courage one day
afternoon breeze jonathan swift's love

Poem Perplexity:
18.6311096946562
```

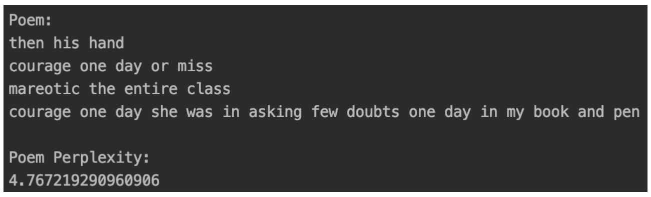


```
Poem:
your eyes mesmerised me
courage one day without therefore enraptured by his hand
pen without outward part
funeral wound

Poem Perplexity:
8.567977551696215
```

Third Model:

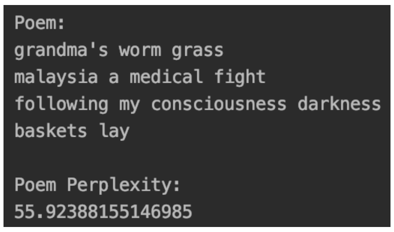


```
Poem:
falsely i am looking into fire
courage one day in text
trouble these may fail or miss
courage one day carried

Poem Perplexity:
8.866307426726578
```

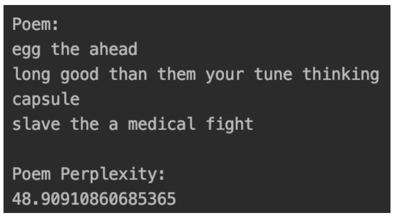


```
Poem:
then his hand
courage one day or miss
mareotic the entire class
courage one day she was in asking few doubts one day in my book and pen

Poem Perplexity:
4.767219290960906
```

Fourth Model:



```
Poem:
grandma's worm grass
malaysia a medical fight
following my consciousness darkness
baskets lay

Poem Perplexity:
55.92388155146985
```



```
Poem:
egg the ahead
long good than them your tune thinking
capsule
slave the a medical fight

Poem Perplexity:
48.90910860685365
```

3 Summary

According to my experiments and their results I can say that these situations would improve results:

- Increasing data size for training
- Adding more hidden layer
- Increasing dimension size of hidden layers
- Increasing epoch number
- Increasing GloVe word embedding dimension size