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

Submission Assignment #4

Instructor: Burcu CAN *Name:* Utku İPEK, *Netid:* 21627356

Abstract

In this assignment, it is expected to build a neural language model to generate poems. The assignment consists of three sub-tasks: building the neural network, generating new poems and evaluating them. Following sections exposes the details of each sub-tasks and the results can be found in last section.

1 Feed Forward Neural Network Language Model

Before the neural language model is built, first, a class is created for all the tasks which is named *PoemGenerator*. When creating an object of this class, the user determines the numbers like embedding dimension, number of units in the hidden layer, batch size, number of poems that will be used while building and training the neural network.
The input vector x of the neural network is a context word represented by a vocabulary size one-hot encoded vector. To get the embedding of a given context word, matrix C which holds the GloVe pre-trained embeddings has been created. So, the vector of the embedding layer, or another words the embedding of a given context word can be obtained multiplying C with x. After the embedding layer vector is obtained, to obtain the hidden layer vector, matrix U and vector b has been created. U holds the weights and b holds the biases. So, the vector of the hidden layer can be obtained (U * (embedding vector)) + b operation. In the hidden layer, as an activation function, the *tanh* has been used. To obtain the output layer, matrix H and vector d has been created for weights and biases. Using (H * tanh(hidden layer vector)) + d operation, it can be obtained. The output vector represents the target word of a given context word and it is also represented by a vocabulary sized one-hot encoded vector. In the output layer, as an activation function the *softmax* has been used. Finally, to calculate the loss function, true target words have been determined using the bigrams dictionary that is created while reading the data. As the loss function, binary-log-loss has been used.
After the neural network is built, a function has been implemented to perform the training process which is named *train*. As the training optimizer, Adam has been used. An important note here is that the reason dividing the data into mini-batches is to speed up the training process. Like the batch size, the number of epochs is also determined by the user.

2 Poem Generation

Before generating poems, first, neural networks with different parameter sizes have been trained. After the training process is done, five new poems have been generated for each trained neural network. To generate a poem, the input word given to the neural network should be the beginning of the line token created while reading the data. Until the neural network produces the end of the line token, word generating process continues. Using the function *generatePoems*, the number of poems and the number of lines that will be generated can be determined by the user. An important note here is that to avoid encountering the target words while generating the poems, a function named *getNextWord* has been implemented that does a random weighted choice from output layer.

3 Evaluation

To evaluate the poems, a function has been implemented which calculates the perplexity of a given poem. Since the perplexity uses the inverse probability of a given text, another function has been implemented to calculate the probability of a given poem. To obtain a probability value, unigram and bigram dictionaries of the data has been used.

4 Conclusion

Firstly, to get the results faster, 5000 of the 93000 poems have been used while training neural networks. Secondly, different parameter sizes have been chosen to train the networks. As the number of units in the hidden layer, 8, 32, and 64 are used. Increasing it has made the model more complex, so, has decreased the number of epochs needed to reduce the loss. As the batch size, 32 and 64 are used. Increasing the batch size has also increased the number of epochs needed to reduce the loss, but the time passed per epoch has decreased. As the embedding dimension, 50 and 100 are used. Increasing the embedding dimension has increased the training time but, it has also made the model more complex, and has increased the diversity of the words generated. An important note here is that as the loss reduces, the perplexity of the poems reduces but, the loss should not be reduced too much, because that causes generating almost identical poems over and over again and it can be interpreted as the model overfits the data.
The results can be seen below:

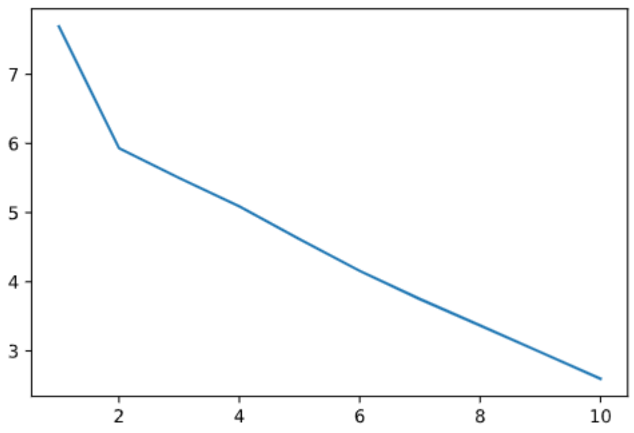

Figure 1: Decrease of loss per epoch (embedding dimension = 100, hidden units = 64, batch size = 64)

- Some example poems generated with embedding dimension = 50, hidden units = 8, batch size = 32:
  - and the books
    and me
    and a short
    in the around
    Perplexity = 102.98832926968633
  - in a without the pack
    of all for the this triggered been
    and the like to me
    in with the - lair
    Perplexity = 490.4194985312225
- Some example poems generated with embedding dimension = 50, hidden units = 32, batch size = 32:
  - with a as your wings
    is in to the knife
    and my out of there
    my for the that a of the to the clashed
    Perplexity = 564.0166995524545
  - in his heart
    and the amp listening of
    to the they as the further
    and me
    Perplexity = 261.8386416304184
- Some example poems generated with embedding dimension = 50, hidden units = 64, batch size = 64:

  - and the smile
    life
    is that by the poet
    and the make you
    Perplexity = 148.97678488634796
  - with a of wishes
    and the of
    of in the pots the month
    and my confidence seven-fold so tin horse
    Perplexity = 684.3158935367801

- Some example poems generated with embedding dimension = 100, hidden units = 32, batch size = 64:
  - with the every to to to your on the path
    and in came to his world
    with the pages
    can the toward past
    Perplexity = 629.912520597827
  - for the schooner in the for me
    and the pack
    in the rancour emerald justice
    for the last
    Perplexity = 271.1277775541776

- Some example poems generated with embedding dimension = 100, hidden units = 64, batch size = 64:
  - not to a neither is it that that as the this is the reconstruction
    and her
    with the becomes a in the pain
    sharp history
    Perplexity = 646.2409448105682
  - at the flute-music
    an' atoms of inherited
    children
    and
    Perplexity = 381.1453205330207