Assignment 4

A Preprint

Didem Yanıktepe
yanikteped@gmail.com

June 2, 2019

1 Introduction

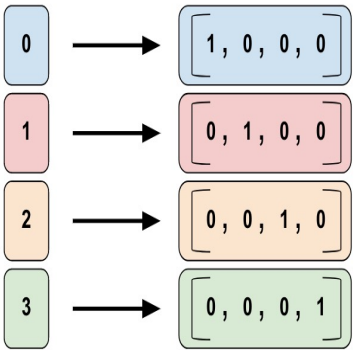
The main purpose of a neural network language model is the language models are trained so as to predict the upcoming word given its previous context, which are the words appearing before the word to be predicted. The word and its context is usually referred as an n**-gram**, which is a contiguous sequence of n tokens (words in textual data), consisting of one word to be predicted and $n-1$ words in the given context. For example, the word combination "The dog runs" is an n-gram with order of 3, and the language models aim at predicting the word "runs" given the context "the dog".

2 Dataset

There is 93264 poems in dataset. It is very big for me so firstly i used 100 of them. Training lasted 7-8 hours for 5000 poems. In dataset the line split with newline. I added $< s >$ and $< /s >$ all poems for better learning. I added poem's line's end *newline*.

2.1 One Hot Vector & Bigrams

One-hot encoding can be applied to the integer representation. This is where the integer encoded variable is removed and a new binary variable is added for each unique integer value. I read all words then i represented all unique word as a int value there 115561 one hot vector which is size of one hot vector and there are 3837056 bigrams.



(a) One Hot Vector

(b) Bigrams

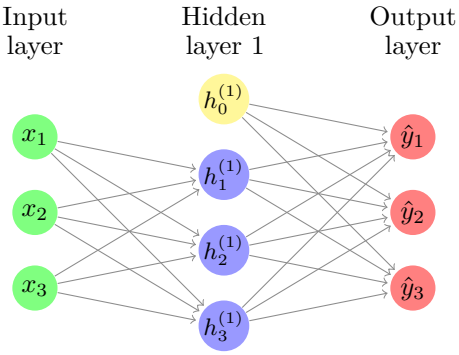
3 Task 1: Build Feed-Forward Neural Network Language Model

I built a feed forward neural network and trained them with bigrams. I used these function for forward

$$y(x_1, \ldots, x_n) = U * f(Wx + b) + d \tag{1}$$

- w_i,b,U,d are parameters
- f is the activation function is tanh

for backward dynet can calculate with its backward function.For trainer I used simleSGDtrainer which is in dynet library. And I tried different learning rate,different hidden layers its shown in Evoluation section.



In this neural network simulation x_i is one hot vector and $\hat{y}_i$ is output after feed forward.

3.1 Parameters in Dynet

```
    def __init__(self, input_dim, hidden_dim, output_dim, learning_rate):
        self.model = dy.Model()
        self.trainer = dy.SimpleSGDTrainer(self.model, learning_rate=learning_rate)
        self.W = self.model.add_parameters((hidden_dim, input_dim))
        self.b = self.model.add_parameters((hidden_dim,))
        self.U = self.model.add_parameters((output_dim, hidden_dim))
        self.d = self.model.add_parameters((output_dim,))
```

3.2 Training

```
    def training(self, input, output):
        for input_, gold in zip(input, output):
            dy.renew_cg()
            input_ = dy.inputVector(input_)
            h = dy.tanh(self.W * input_ + self.b)
            pred = self.U * h + self.d
            i = list(gold).index(1)
            loss = dy.pickneglogsoftmax(pred, i)
            loss.backward()
            self.trainer.update()
```

4 Task 2: Poem Generation

After training a model i write 2 generated method one of them cumulative sum for generation one of them using softmax. But when I used softmax after *START* token the line is always same same as *NEWLINE* token so I randomly choose 10 of them which has the maximum probability in softmax matrix.

4.1 Five Poem

4.1.1 First Poem

father soundless bones farewell time was
as breath her distracted farewell there
as breath her distracted farewell there
was bones farewell time was bones
her distracted farewell there are corpses

Perplexity = 201.18233836614183

4.1.2 Second Poem

there was upon at and left
i'd strength and left and left
us make our strength and left
yes upon at and left and
us make our strength and left

Perplexity = 158.42456043164276

4.1.3 Third Poem

and laugh
i'd for love them back to make us
i'd wonder what
strength ask wonder for sleep

Perplexity = 164.4571431221987

4.1.4 Fourth Poem

not smiles wonder what we'd them
we them wonder what we'd them
desire them wonder what we'd them
would is wonder what we'd them
we them wonder what we'd them

Perplexity = 194.076628451479

4.1.5 Fifth Poem

those words weep
desire them back to make uncontrolled
i'd wonder what what
we'd could baby we could keep could keep

Perplexity = 197.35126675426756

5 Task 3: Evaluation

Evaluation				
Traning Poem	Epoch	Learning Rate	Hidden Layer Node	Perplexity
10 poems	20	0.0001	1024	183.5972
	20	0.005	1024	132.417
	50	0.005	50	112.02
	50	0.005	100	108.87
100 poem	10	0.0001	50	105.67
500 poem	50	0.0025	50	102.98

For 10 poems when i increase hidden layer node size perplexity is increase i think it because the model does not learn it is memorize it. When I decrease epoch size the perplexity is better. I decrease the learning rate tthen I got more meaningfull poems for using argmax.