Assignment2 : Neural Language Models

Arpan Mangal
M.Tech, CSA (14353)
Indian Institute of Science
arpanmangal@iisc.ac.in

Abstract

In this Assignment, I experiment with Language Models on **Gutenberg corpus** corpus. There are two tasks which are performed.We divide each dataset into train, dev, and test.
Task1: I build token-level LSTM based model language model. I evaluate the results based on "perplexity" as metric.
Task2: I build character-level LSTM based model language model. I evaluate the results based on "perplexity" as metric.
Task3: Using our best model in both token-level and character-level, I generate a sentence of 10 tokens.
I compare our results from Assignment-1 to compare the results obtained from classical N-gram LM and Neural LM.

1 Pre-Processing

Before actually applying language models on corpus, I need to pre-process,to clean the data. For this task, I need to generate good sentences in the end. And I have to evaluate the model based on "Perplexity" as metric. These steps are taken:

1. Stop Word Removal and Stemming and Punctuation can't be performed because they are basic part of grammatically correct sentence.
2. Since LSTM based model uses the memory state to remember history, it doesn't matter if I covert data to lower case.
3. Append each sentence with start token in the beginning. Then I can pass whole text as string to the model.Model can identify the beginning of sentence with start token.

2 Splitting

In order to get unbiased estimate, it's important that all the three, **train**, **dev** and **test** data set has same proportion of each type of sentence. Therefore, I first shuffled all the sentences and then split as **80% train, 10% dev and 10% test** data. After splitting the sentences, I just merge all the sentences in the corresponding split, by inserting "start tag"(for token-level LSTM only) in between. It will be helpful as I am measuring perplexity as evaluation metric, which need text as one sentence.

3 Character Level LSTM

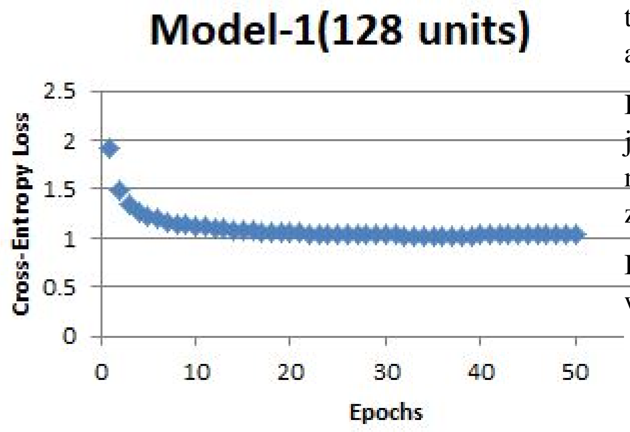
I have tried building three models for this. All the models were using two hidden layers of LSTM. They differ in the number of units in hidden layers. Both the hidden layer contains same number of units.Model 1,2 and 3 have 128,256 and 512 units in each hidden layer respectively.

I tried training the models on half on the gutenberg corpus. My training set contains 2 millions characters. I have used these parameters:

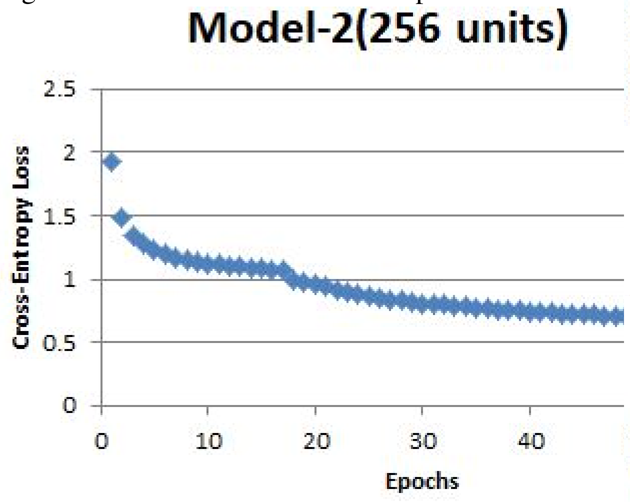
- Back-propagation through time : 100
- Stride : 3
- Batch size : 128
- Learning Rate: 0.001

1. **Training**
   I tried changing these parameters for small dataset to check. I didn't observe any progress. Therefore, for the final experiment I kept these parameters fixed as above.

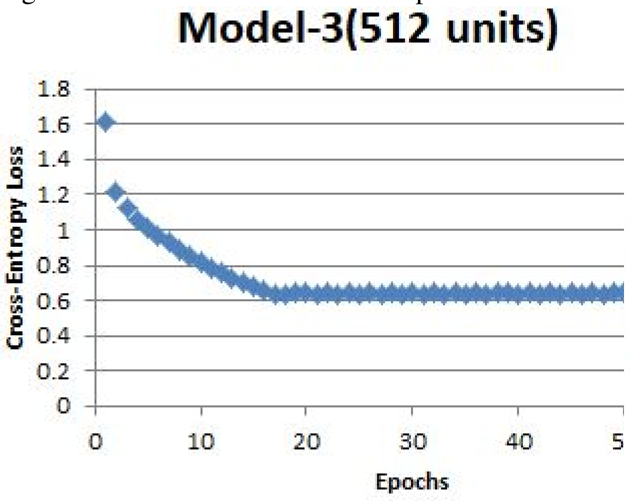
   I trained my models for 50 epochs.In each epoch, I calculated cross-entropy loss.Following are the graphs for that. To validate and test, I have used weights corresponding to the minimum loss in the corresponding model.



It gave minimum loss of 1.0204 at epoch 39.



It gave minimum loss of 0.7015 at epoch 50.

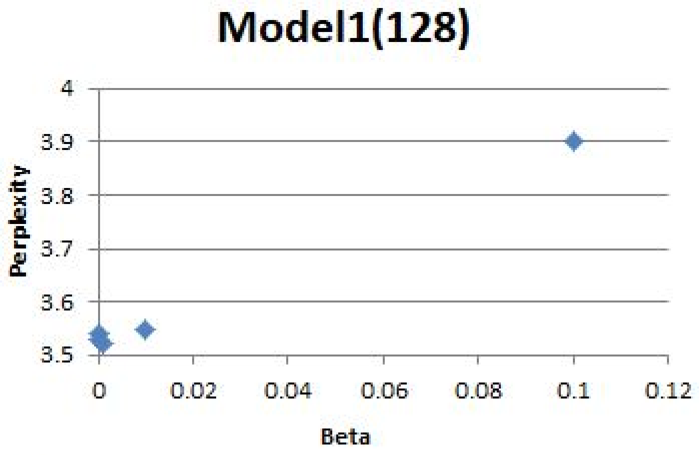


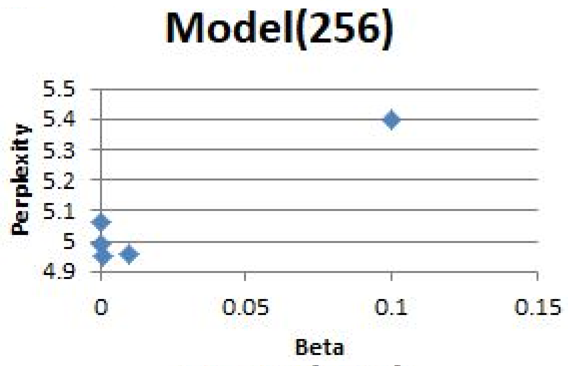
It gave minimum loss of 0.6383 at epoch 17.

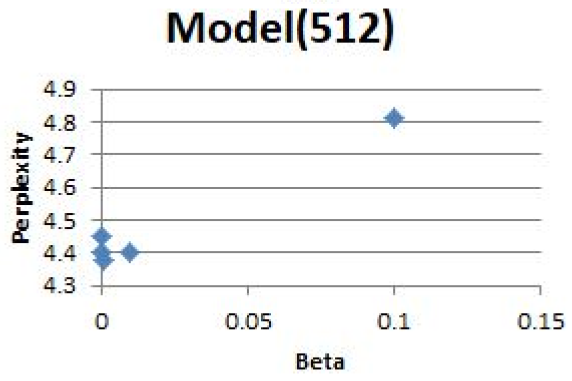
2. **Validation**
   As output, model used to predict softmax probabilities of each character. There were 56 unique characters in the part of corpus used for experiment. To calculate perplexity, I need to smooth the probability for each character in order to avoid zero probabilities. To smooth the probability, each character probability was tweaked a bit by:

   It ensures that probability sums to one. Its just remove very small probability mass from non-zero probabilities and distribute among zero probabilities.

   It involves use of beta as a hyper parameter, which is tuned on validation set.







All of them gave minimum perplexity at beta = 0.001, which is taken for evaluation on test set.

3. **Testing** These are the values of perplexity obtained on test set using different models.

- **Model 1 : 3.54**
- Model 2 : 4.97
- Model 3 : 4.39

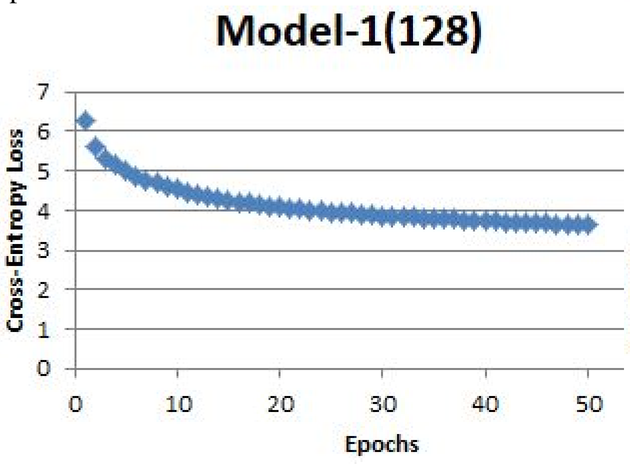
4 Token-Level LSTM

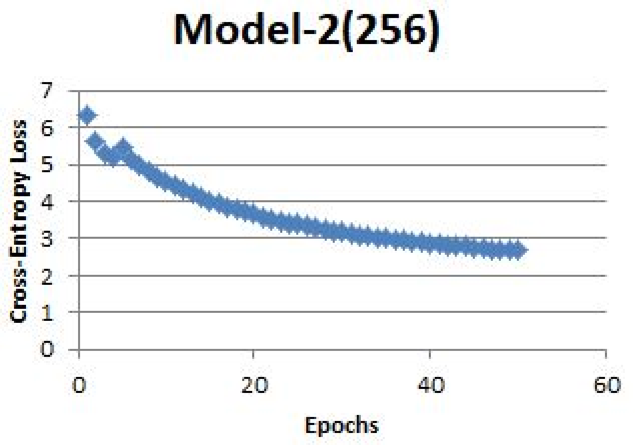
Similar to character-Level LSTM, I have tried to fit three models based on number of units in the hidden layers. In alll the three models I have two hidden layers with same number of units. Model 1,2 and 3 has 128,256 and 512 units in each hidden layer respectively.

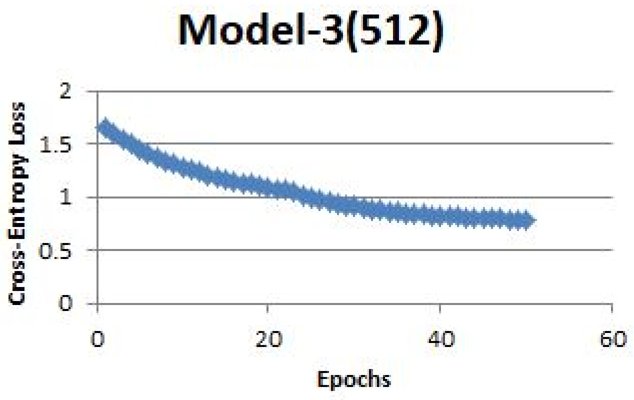
For this model, I have trained my model on **whole gutenberg corpus**. Since vocabulary contains approximately 40K unique tokens, it was infeasible to feed one-hot vector as input to LSTM model. Therefore, I have used pre-trained Glove word embedding as my input. I have tried with differnt dimensions of Glove embedding. Embedding with 300 dimension gives the best result(test on sample data). Therefore, it is used for main experiment. I have used these parameters:

- Back-propagation through time : 20
- Stride : 3
- Batch size : 128
- Optimizer : Adam
- Learning Rate: 0.001

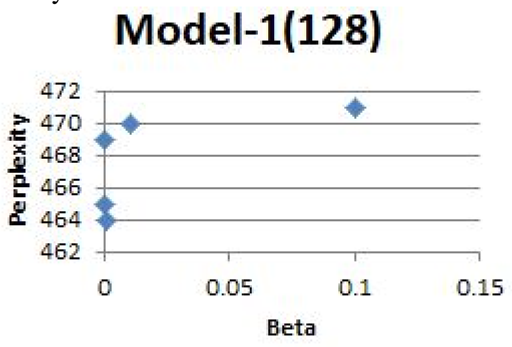
1. **Training**
   I trained all the models for 50 epochs.In each epoch, cross-entropy loss is calculated. Following graphs shows cross-entropy loss v/s epoch for all the models.

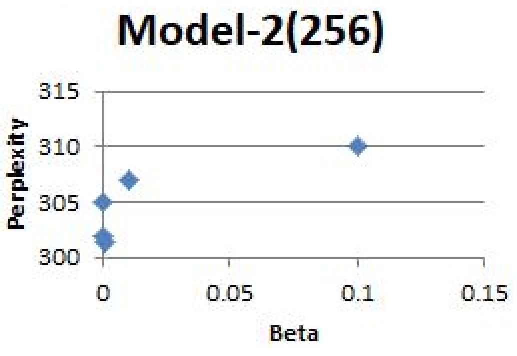


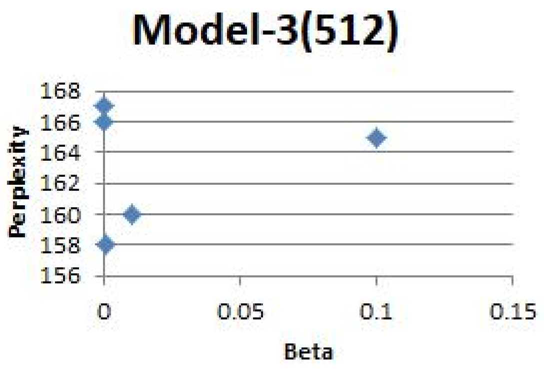




2. **Validation**
   Same as Character-Level LSTM, all these models are validated to tune the value of hyper parameter ”Beta”. Beta corresponding to which we get less is chosen to calculate perplexity on Test set.







All of them gave minimum perplexity at beta = 0.001, which is taken for evaluation on test set.

3. **Testing** These are the values of perplexity obtained on test set using different models.

   - **Model 1 : 459**
   - Model 2 : 304
   - Model 3 : 197

5 Sentence Generation

For generating the sentence, I am using best model, i.e the one who gave least perplexity on test set. Following are few examples of sentence generated:

Character-Level LSTM

- indeed you must and shall come what was to be
- the children been there the affair might have been determined
- should think it possible for me to have such feelings

Token-Level LSTM

- To send these three measures of post amongst you killed
- In the air with his twenty feet there which one
- pastorall and hittites though corrupt i see the prospect and

6 Comparison with Classical LM

In comparison with Classical LM implemented in Assignment-1, I observed that neural network based LM are better provide you trained them enough.Due to time constraints, I was not able to train them much. I just trained each model for 50 epochs. But with that as well NN based LM were generating good sentences and their perplexity is close to classical LMs. NN based model in my case were unable to beat Kneser-Ney Model implemented in Assignment-1. May be if we train them more, they are able to beat it as well.

7 Inference

Classical LM gives acceptable results in short time. NN based LM can beat Classical LM, but they need more time to train.

8 Source Code

https://github.com/mangalarpan/LSTM_based_LM