Assignment 2 - LSTM based Language Model

Sourabh Balgi
Sr. No. - 14318
EE Department
M. Tech. (Systems Engineering)
sourabhbalgi@gmail.com

Abstract

This document contains the second assignment report with the details of the methodology, observations, results and plots for **Character level and Word level language models were implemented using the *LSTM (Long-Short Term Memory)***. The LSTM models were used to compare the results from the Assignment 1 over the same dataset with same Train, Dev and Test partitions. Important observation from this assignment is the high improvements over the N-gram model in terms of perplexity. Another observation, CUDA enabled PyTorch was able to provide almost 30 times faster training when compared to the CPU version.

1 LSTM language model

1.1 Data splitting and Pre-processing

The corpus was split in 80 % Train, 10 % Validation and 10 % Test based on the number of sentence in each files of Gutenberg corpus. The data split was maintained to be same as N-gram model modeled in Assignment 1 for accurate results comparison. After pre-processing and tokenization, A total of 47 unique characters were obtained as vocabulary for the Character level training data and 30459 tokens vocabulary size for the word level training data. For Character level model and Word level model , "_" is used as unknown token to handle Out-of-Vocabulary tokens during perplexity calculation in Validation and Test data. Only a small set of punctuations were kept along with all the alpha-numeric characters. This helped in reducing the vocabulary size in case of character level model. In case of word level model, a threshold count of 2 is used to replace the words as unknown tokens to handle Out-of-Vocabulary words during perplexity calculation and also reduce vocabulary size. So words with count less than 2 are replaced as unknown token "_" in the dataset.

1.2 Tokenization

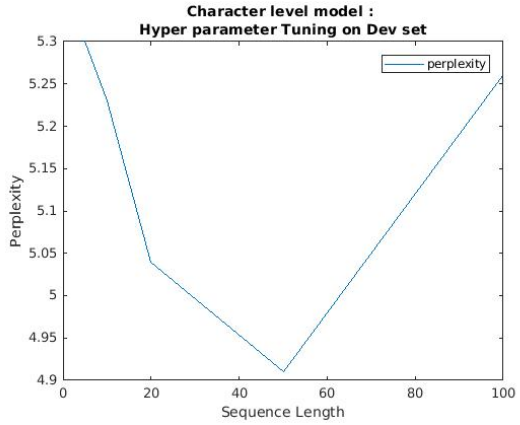
Two separate models were trained 2 observe the behavior of the different types of text processing and tokenization.

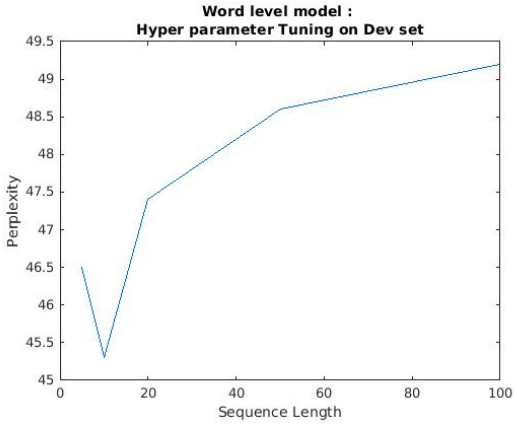
1.3 With replacement of uppercase to lowercase

In this, the complete corpus was converted to only lowercase and trained on character level. The vocabulary size in this method was 47. The Char level model generated from this data produced a test perplexity of 3.9. This type of tokenization helped in training the model faster because of the small vocabulary. Even though correct sentences were generated after training, The sentence was *not human-like* as everything is in lowercases. The Word level model also produced a perplexity of 59 in test data.
***Example** : make the mistaking the real of the as Mrs. Deart shall resolution the flag! As all that in the more*

1.4 With no replacement of uppercase to lowercase

In this, the corpus was not converted to lowercase and is being used as is. In this case, The vocabulary size was slightly more around 70. The Char level model generated from this data produced a test perplexity of 4. This type of tokenization took more time in training the model as the vocabulary is large. However sentences generated after training in this case was *more human-like* with both upper and lowercase letters and punctuations.
***Example** : first meeting , and yet no man will stretch out my hands , and as the waves of the sea , the whip tree shall needs be like water , and it is in*





the midst of them . 8: 3 And he will deliver tidings to the children of Israel , but it shall be the habitation of death : they shall know that I am the LORD . And the king went down against him to judge the king on the third day , and said , Lord , I am not with thee : thou shalt be

2 Model Implementation Methodology

All the models were implemented in PyTorch framework with cuda for GPU acceleration. The pre-processing of the text data was done separately for Character level and Word level model.

2.1 Character Level LSTM

In Character level LSTM, Each character is considered to be separate token for modelling and represented as 128 dimensional word embedding. Two LSTM layers were created with hidden size of 128. The data was divided into 50 batches and a sequence length of 50 was considered as input for the LSTM model. The output of the LSTM model was obtained through a fully commented layer with softmax activation to produce the probabilities for each of the possible outputs from the vocabulary. Cross Entropy loss was used as the cost function for optimization and gradient descent.

2.2 Word Level LSTM

In Word level LSTM, Each character is considered to be separate token for modelling and represented as 300 dimensional word embedding. Two LSTM layers were created with hidden size of 300. The data was divided into 50 batches and a sequence length of 10 was considered as input for the LSTM model. The output of the LSTM model was obtained through a fully commented layer with softmax activation to produce the probabilities for each of the possible outputs from the vocabulary. Cross Entropy loss was used as the cost function for optimization and gradient descent.

2.3 Observations

Both the models were trained for 40 epochs using GPU. The total training time was seen to be around 3-4 hours with GPU for 40 epocs and 6 hours without GPU for just 1 epoch. so the model generation was terminated and only GPU enabled training was used there after. The hyper parameter sequence length was tuned using grid search method. In case of character level model, sequence length of 50 yielded the best perplexity over the validation set among the other values 5, 10 ,20 ,50 and 100 for the sequence length. In case of word level model, sequence length of 10 produced the best perplexity over the validation set among the values 5, 10, 20, 50 and 100 for the sequence length. The model can be further fine tuned for all the hyper parameter such as *batch size, number of hidden LSTM layers, size of the word embeddings, sequence length* .

3 Model Evaluation Metric

Perplexity: The perplexity (PP) of a language model on a test set is the inverse probability of the test set, normalized by the number of words.
For a test set $W = w_1 w_2 ... w_N$:

$$\text{PP(W)} = \text{P}(\text{w}_1 w_2 ... w_N)^{-\frac{1}{N}} = \sqrt[N]{\frac{1}{P(w_1 w_2 ... w_N)}}$$

The generated model is tuned for the discounting parameter over the development set and then tested on the test set. The perplexity defines the measure of likely predicting the test set. A low perplexity would imply a very good model, as most information is captured in the model

In case of LSTM model, The perplexity is calculated as
PP(W) = $e^{crossentropyloss}$

4 Results

Corpus : Gutenberg from NLTK
Number of epochs : 40

- *Character Level LSTM model* :
  - Validation perplexity (lowercase only) : 3.9
  - Test perplexity (lowercase only) : 4.0
  - Validation perplexity (both) : 4.1
  - Test perplexity (both) : 4.5
- *Word Level LSTM model* :
  - Validation perplexity (lowercase only) : 45.2
  - Test perplexity (lowercase only) : 59.6
  - Validation perplexity (both) : 76.4
  - Test perplexity (both) : 85.3
- *N-Gram model from Assignment 1* :
  - Validation perplexity (lowercase only) : 126.4
  - Test perplexity (lowercase only) : 159.3

5 Model Selection

Both the models were used in the sentence generation to compare the nature of sentence generated. In word level model, the sentence generated rarely switched the context while generating the sentence. For example, if it was initialized to words from Bible , It continued generating sentences in the context of Bible. However in the case of the character level model, It was seen to switch between a variety of contexts while generating the sentence. One possible reason for this is the sequence length used to train the model. Since the sentence generated were fairly good with both the models, The character level LSTM model was selected for final sentence generation as it produced the lowest perplexity.

6 Sentence Generation

1. *Character Level LSTM model*
   Example : make the mistaking the real of the as Mrs. Deart shall resolution the flag! As all that in the more
2. *Word Level LSTM model*
   Example : first meeting , and yet no man will stretch out my hands , and as the waves of the sea , the whip tree shall needs be like water , and it is in the midst of them . 8: 3 And he will deliver tidings to the children of Israel , but it shall be the habitation of death : they shall know that I am the LORD . And the king went down against him to judge the king on the third day , and said , Lord , I am not with thee : thou shalt be