Natural Language Processing, 2018: Report on Assignment-2: Language Modeling using RNN-LSTM

Sqn Ldr Abhimanyu Vatta
M.Tech. First Year, CSA, IISc
SR No: 15251
abhimanyuv@iisc.ac.in, v.manyu@gmail.com

1 Problem Statement

First Assignment dealt with developing a Language Model using classical N-Grams. The Language built used Tri-Gram along with K-smoothing technique. Second Assignment involves developing of Language model using Long Short Term Memory (LSTM) Neural Network, a variant of Recurrent Neural Network (RNN) technique. Guidance has been taken from various links and papers such as (Olah, 2015), (Goldberg, S, 2017) and (Tensorflow, 2018). Current assignment is divided into three tasks which are briefed below:

1.1 Task-1

The task comprise of building a Token(word) level LSTM-based language model using Gutenberg Corpus by training and testing the the model on same corpus. The dataset (9 files) has been divides as follows:

- Trainning : 55% (2 files each of Shakespeare and Chesterton, 1 file of Austen)
- Heldout : 10% (1 file of Austen)
- Testing : 35% (1 file each from Shakespeare, Chesterton Austen)

1.2 Task-2

Second task is related to the development of Character level LSTM-based language model using the same settings as mentioned in Task-1

1.3 Task-3

Using the Language Models, developed in Task 1 and 2, we need to develop a script with the name $generate\_sentence.sh$ which will return a sentence of 10 tokens.

2 Baseline Language Model

Baseline to compare will be the Language Model developed in Assignment-1. However, it may be noted that the attributes differ from previous assignment and hence, the dataset used in assignment-2 is also used to rebuilt Language Model of assignment-1 for comparison.

2.1 Assignment-1:

- Perplexity : 489.636
- Sample sentence : as macian had understood his man pretty well , here

2.2 Assignment-2:

- The following details are at end of training model for 100 Epochs. The results are not good as the training when submitting report is very less, however the model is bound to improve with more training as it is a continuous process.

2.2.1 Token Based Model

- Perplexity: 270.42 (50 Epochs), 221.04 (100 Epochs)
- Loss: 5.6 (50 Epochs), 5.4 (100 Epochs)
- Sample Sentence: enter the the unk of the is be the the (100 Epochs)

2.2.2 Character Based Model

- Perplexity: 395.44 (50 Epochs), 340.35 (100 Epochs)
- Loss: 5.95 (50 Epochs), 5.83 (100 Epochs)
- Sample Sentence: of hbhh q!bh the bhbh! qbhhq teh. hbba (100 Epochs)

3 Token based LSTM

Generation of token based LSTM Language model consists selection of text parsing techniques, reshaping of the tokens and training configuration.

3.1 Pre-processing

NLTK library for the Brown and Guttenberg corpus have not been used, instead pre-processing has been carried out using regular expressions. Pre-processing of the corpus involves following subtasks:

- Reading of files
- Identifying the period punctuations (.?!) as end of sentence
- Removing other special characters.
- Addition of end-of-sentence symbol $\langle eos \rangle$
- Tokenization of the sentences
- Removal of single characters
- Declaring words occurring once in corpus as UNK.

3.2 Model Generation

LSTM Model is developed using Google's Tensorflow. Model is configured with CPU Id only. Check-pointing has been done using the tensorflow state saver utility to save the check points after each fixed number of epochs for continued training. Various parameters used are as follows:

3.2.1 Sequence Length

Sequence length determines the number of words fed to a single cell and thereby determine the time-steps involved in an unrolled LSTM graph. The variable num_steps is used for this purpose and set to 10 which is 80% of the average size (13) of a sentence in training corpus.

3.2.2 Batch Size

Batch Size helps instantiate various graphs in parallel and is the core of Tensorflow as it helps in faster and efficient training. Variable batch_size is set to 30.

3.2.3 Iterations

The whole dataset is divided into Batch Length of the $DatasetLength \div SequenceLength$. Therefore the matrix obtained is of dimension $BatchSize \times BatchLength$, however, sequence length(num_steps) $\bar{1}0$ requires us to run an epoch for $BatchLength \div num_step$ iterations. Therefore, the whole dataset is fed to LSTM in every epoch.

3.2.4 Layers

It helps implement a layered structure stacked upon each other and the corresponding variable num_layers is set to 2, thereby making the LSTM model doubly layered.

3.2.5 Hidden Layer Size

The parameter helps define the number of cells in a single hidden layer. The variable hidden_size is set to 100.

3.2.6 Dropouts

Dropout is used to drop the weights while values are transfered from one layer to another in stack model and is set to 0.5. It helps in drawing long term dependency compared to a single layer LSTM model.

3.2.7 Gradient Optimizer

Adagrad Gradient optimiser has been used after comparing with ADAMS and Basic Gradient Optimizer. The main reason was the momentum it provides for faster learning compared to Basic Gradient Optimizer and better learning curve when compared to ADAMS. Learning rate also has also been changed after every 10 epochs to avoid over/under fitting.

3.2.8 Batch Generator

Batch Generation is a key component in the training of the dataset. A sequence is fed after making calculations based on Sequence Length(num_steps) and Batch Size in every epoch. The function batch_generator does this task.

3.2.9 Epoch

The total number of above explained task to be carried out in a single cycle in one Epoch. More number of epochs will result in better training but may also lead to over-fitting. The model trained uses 1000 epochs for training.

4 Character based LSTM

Generation of token based LSTM Language model consists selection of text parsing techniques, reshaping of the tokens and training configuration.

4.1 Pre-processing

NLTK library for the Brown and Guttenberg corpus have not been used, instead pre-processing has been carried out using regular expressions. Pre-processing of the corpus involves following subtasks:

- Reading of files
- Identifying the period punctuations (.?!) as end of sentence
- Removing other special characters.
- Tokenization of the dataset into characters

4.2 Model Generation

Only the difference from Token base LSTM model are highlighted below.

4.2.1 Sequence Length (num_steps)

It is set to 20 which $5 \times averagesize(4) ofaword$ in training corpus. Therefore it learns dependency among last five words.

4.2.2 Hidden Layer Size

The hidden layer size is set to 36 which is the vocabulary size in this case.

4.2.3 Epoch

The model trained uses 2000 epochs for training which is much larger than Token based LSTM.

5 Observations

5.1 Token Based LSTM Model

The final model was trained with Hidden layer size = 100, Sequence length(num_steps) = 10, Batch_size = 30 and Layers (num_layers) = 2. Below given charts give a comparison on changing either of these parameters and corresponding changes in the values of Loss and Accuracy for 10 epochs. After extensive trials the final parameters were fixed as given above.

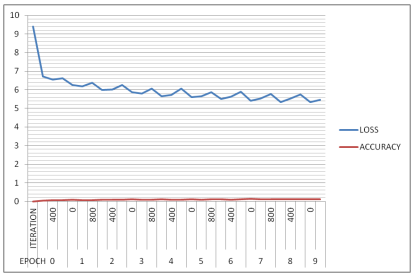

Figure 1: Hidden_size = 100, Sequence_legth = 5, Batch_size = 30

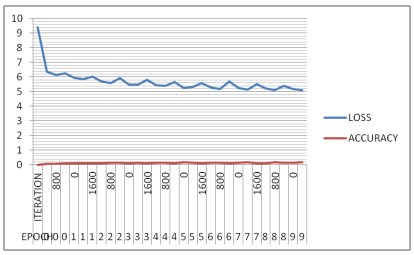

Figure 2: Hidden_size = 100, Sequence_legth = 10, Batch_size = 10

- **Change of Sequence Length to 5** (Figure1). Loss starts of high at a value of 9.4 with a sharp drop to 6.7 attaining a minimum of 5. High fluctuation in loss might result in missing the minima and hence the model is not good for learning. Accuracy starts at 0.3% and rises to maximum of 12% at the end of last epoch.
- **Change of Batch Size to 10** (Figure2). Loss starts of high at a value of 9.4 with a sharp drop to 6.3 attaining a minimum of 5.3. Although there is high drop in loss initially but fluctuation of values in subsequent iterations is less than previous model. Accuracy starts at 0% but rises to a maximum of 17.5%. Hence, the increase in sequence length results in better learning with less chance of missing minima.
- **Final Model** (Figure3). Loss starts at modest a value of 6 and slowly drops to a minimum of 5. Accuracy also starts at good 6.2% and reaches 15.7% at the end of last epoch. The model proves to be good at learning and performance.

5.2 Character Based LSTM Model

The final model was trained with Hidden layer size = 36 (same as vocabulary size), Sequence

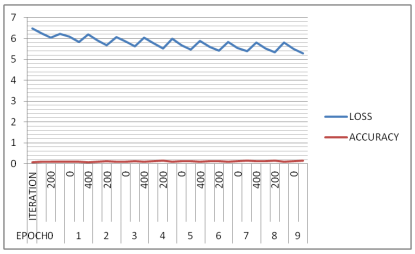

Figure 3: Hidden_size = 100, Sequence_legth = 10, Batch_size = 30

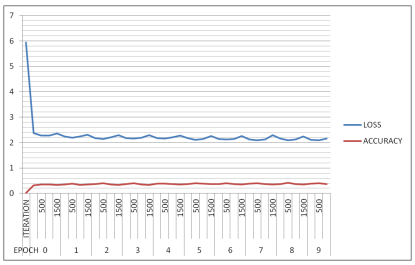

Figure 4: Hidden_size = 36, Sequence_legth = 10, Batch_size = 30

length(num_steps) = 20, Batch_size = 30 and Layers (num_layers) = 2. Below given charts give a comparison on changing either of these parameters and corresponding changes in the values of Loss and Accuracy for 10 epochs.After extensive trials the final parameters were fixed as given above.

- **Change of Sequence Length to 10** (Figure4). Loss sharply drops from a high initial value of 6 to 2.3 up to a minimum of 2.08. Accuracy sharply rises from 16% to 32% to a maximum of 40%. The model overall fluctuates over the values and is likely to miss the minima.

- **Change of Batch Size to 10** (Figure5). Loss starts with a low value of 3.5 and slowly drops to a minimum of 2.15. Accuracy also starts low compared to previous model at 15% and rises to a maximum of 36%. Therefore, reduction in batch_size brings about reduction in learning rate.

- **Final Model** (Figure6). Loss starts with a value of 3.6 which is higher than both the above models. Loss very slowly drops to a minimum of 2.1 which helps in better learning without missing minima. Accuracy starts very low compared to previous models at 4.2% but due to good learning it rises sharply to 23.7% and reaches a maximum of 40%. The model learns and performs better.

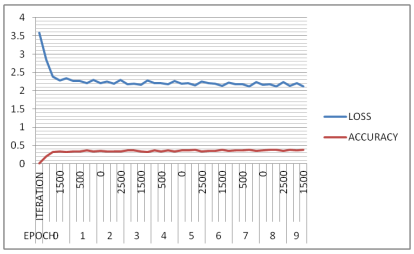

Figure 5: Hidden_size = 36, Sequence_legth = 20, Batch_size = 10

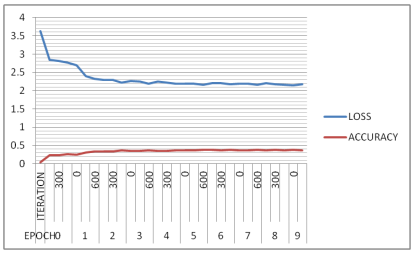

Figure 6: Hidden_size = 36, Sequence_legth = 20, Batch_size = 30

6 Usage

Solution contains two python files *LSTM_word.py* for building Token Based LSTM model and *LSTM_char.py* for building character based LSTM model along with six .sh files, one folder containing the dataset used (Guttenberg) and two folders for maintaining logs of each model (used for check-pointing).

The files are located at *e1-246-30@clserv:/̃NLU/Assign2/*. It may be noted that before running any of the files tensor needs to be activated with command *source /̃tensorflow/bin/activate* after which the command prompt will change to *(tensorflow) e1-246-30@clserv:/̃NLU/Assign2$*

The usage of this Language Model interface is:

LSTM_word.py [-t] [-t T]
LSTM_char.py [-t] [-t T]

- -t: Set 1 for Training, 2 for getting Perplexity & 3 for Generating Sentence.
  (Default value set to 3).

6.1 Running various Tasks

There are a total of six .sh files to carry out following tasks.

6.1.1 Token Based LSTM Model

- Training: sh train_LSTM_word.sh
- Perplexity: sh perplex_LSTM_word.sh
- Sentence: sh gen_sentence_LSTM_word.sh

6.1.2 Character Based LSTM Model

- Training: sh train_LSTM_char.sh
- Perplexity: sh perplex_LSTM_char.sh
- Sentence: sh gen_sentence_LSTM_char.sh

7 Conclusion

The model trains well if run for more number of epochs, however the perplexity gradually falls up to a certain limit after which it starts to increase due to over-fitting. In contrast the accuracy keeps increasing may be due to the fact that model will now generate predictable sentences.

References

Goldberg, S. 2017. *Neural Network Methods for Natural Language Processing*. Morgan Claypool Publishers.

Chris Olah. 2015. Understanding lstm networks. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

Google Tensorflow. 2018. Tensorflow: An open-source machine learning framework for everyone. https://www.tensorflow.org/.