Language Modelling Using Long Short-Term Memory(LSTM) Neural Networks

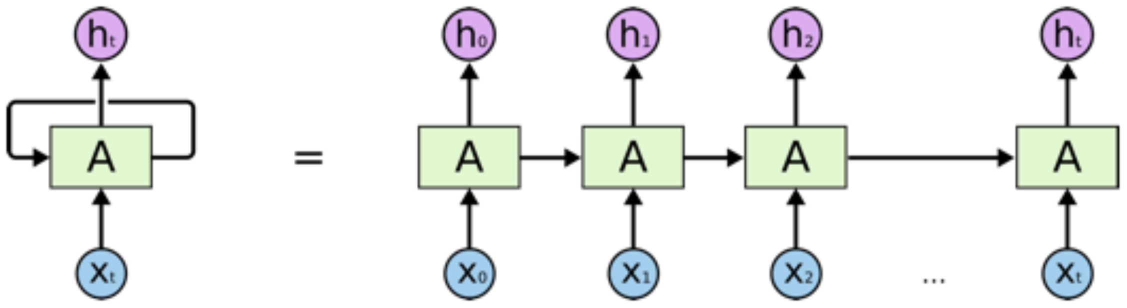
Goal

The goal of this project is to explore Long Short-Term Memory Neural Networks and see how it works for the task of language modelling. I have analyzed the effectiveness of LSTM on the Children's Book Test dataset published by Facebook.

Background

A Language model (LM) is a statistical model that assigns a probability to a sequence of words by generating a probability distribution. Language Models are used in different systems like for speech recognition, machine translation, information retrieval, word sense disambiguation etc. Speech recognizer profit from a probability assigned to the next word in a speech sequence to be predicted. Language Modelling is a challenging task because of the sheer amount of possible word sequences: the curse of dimensionality. The goal of language model systems is to ensure that the output of these systems not only convey meaningful words and the correct lexical choices but also that these words are in the right order, so that the output sound fluent in the language model.

Recurrent Neural Networks (RNN): Recurrent Neural Network is an adaptation of the standard feedforward neural network to allow it to model sequential data. They are a class of artificial neural networks which have loops in them so that information can persist. You can think of it as how your brain understands each word. When you read, your brain already has a previous understanding of the word, so you are not thinking about it from scratch. Similarly, even the recurrent neural networks persist information in them unlike the traditional neural networks.

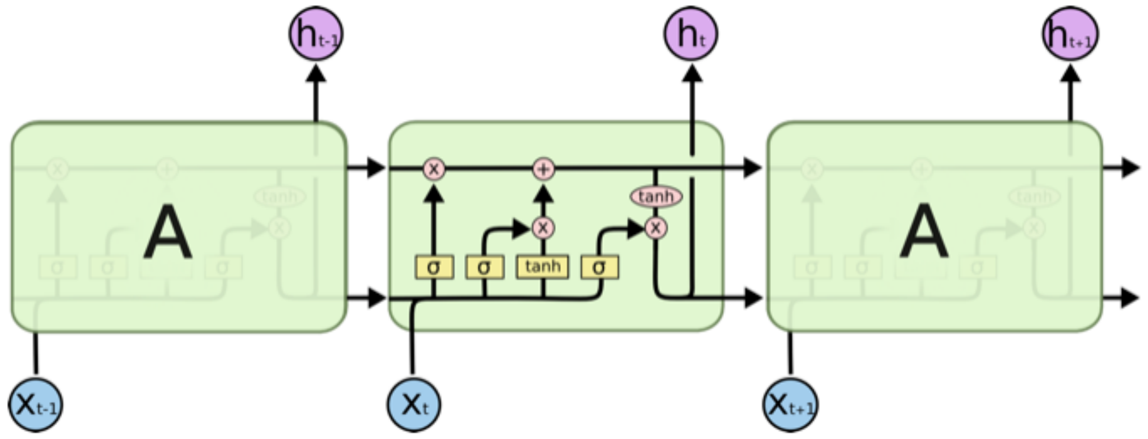


Above is an image of Recurrent Neural Network. On the left side, you can see the loop version and on the right side you have the unrolled version. You can think of the unrolled version as multiple copies of the same network. RNNs can be applied to a variety of problems like speech recognition, language modeling, machine translation, image captioning etc. RNNs have achieved substantial success at Language Modeling Mikolov et al [2]. In this project, I have used a special type of RNN known as the Long Short-Term Memory (LSTM) Neural Networks for Language modeling.

Long Short-Term Memory Neural Networks (LSTM)

With conventional “Back-Propagation Through Time” or “Real-Time Recurrent Learning” the gradients flowing back in time tend to either (1) blow up or (2) vanish. This is because the back propagated error exponentially depends on the sizes of the weights. Because of this RNNs have the problem of long-term dependencies. When the gap between the given relevant information and the next word is small the RNN can learn to use the past information. But if this gap is large, then RNNs cannot learn from the previous information. The LSTM do not have this problem and are capable of learning long-term dependencies.

LSTMs were introduced by Hochreiter and Schmidhuber in 1997 and were refined and popularized by many people. LSTMs were explicitly designed to handle the long-term dependencies problem. Remembering information for long periods of time is what they are designed to do.



Above is the diagram of an LSTM Neural Networks. RNNs also have a chain like structure like above but they have a simple structure like a tanh function. In an LSTM as you can see above the structure is more complex. Instead of having just one single neural layer, there are four, interacting in a very special way.

The top horizontal line is the cell state. It is kind of like a conveyer belt which runs through the entire chain and to which the information can be easily added. The information can also just flow along it unchanged. You can add/remove information by using gates. The gates are composed of a sigmoid neural network and a pointwise multiplication operation. If the value of gate is zero then no information is let through and if it is 1 then all information is sent through.

Forget Gate Layer – This is the first step. It takes the input from the previous layer and the in the case of language model the new word and outputs either 0 or 1. This decides what

information is to be kept. For example, in language modeling, the present subject might be a table and the cell state needs you remember that it is a thing so that the correct pronoun is used. When the subject changes, we want to forget this information.
Input gate – This gate(sigmoid) decides which values are to be updated.
G gate – This is a tanh layer that creates a vector of new candidate values, that could be added to the cell state.
Both of these gates combine to decide what new information is going to be stored in the cell state. After that you update the old cell state into the new cell state.
Output gate – A sigmoid layer decides what parts of the cell states are going to be outputted. This gate filters the cells state and decides the output. LSTMs can be summarized by the below equations:

$$\begin{bmatrix} I \\ F \\ O \\ G \end{bmatrix} = \begin{bmatrix} \text{sigma} \\ \text{sigma} \\ \text{tanh} \\ \text{sigma} \end{bmatrix} W' \begin{bmatrix} h_t^{t-1} \\ h_{t-1}^{t} \end{bmatrix}$$

LSTMs take vector from below in depth and before in time concatenate them and multipliet with weight matrices.

Methodology

Tools - I have used TensorFlow for this project. TensorFlow is an open source software library for machine intelligence. Computation is carried out using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between the nodes.

Dataset

For this project, I have used the children's stories dataset from Facebook called the Children's Book Test. This dataset contains a collection of children's books which were gathered from Project Gutenberg archives. The dataset is 25.7 MB, contains 5.2 million words out of which 68031 are distinct. I experimented with this dataset by creating different smaller datasets of it. This is because training a large file of 25.7 MB takes really long time on a machine without GPUs.

Dataset 1 – I prepared this dataset by removing all the words which have length of above 7 characters and also words which have hyphen in them. The overall count of distinct words was around 27000. The file passed as input was the original file but the words that weren't in the vocabulary would not be considered.

Dataset 2 – Again similar to the previous dataset this too had same constraints but before adding the constrains, I converted all the words to lowercase. This eliminated a lot words which were same but in different form like the and The. This dataset consisted of 22,000 distinct words and the input file remained the same.

Dataset 3 – I trimmed down the dataset from 25 MB to 4.4 MB. I reduced the dataset to contain only 10 books, converted all the characters into lowercase and added the same constrains as above. But, kept words above the length of 8 characters.

Model Configurations – There are various parameters that can be tuned so as to make the model perform better. These parameters make up the configuration of the model. I have used 3 different configurations for the models I have developed. More information about each configuration can be found as comments in the code.

Loss Function **–** A loss/ cost function is a function that maps an event to a numerical value. This numerical value represents the cost associated with the operations. I have used the legacy_seq2seq.sequence_loss_by_example loss function that TensorFlow provides. This calculates the weighted cross-entropy loss for a sequence of logits (wx + b) per example.

Optimizers – In TensorFlow optimizers provides methods to compute gradients for a loss and apply gradients to variables. I have used 2 different optimizers for this project –
AdamOptimizer – This optimizer is based on the Adam algorithm.
GradientDescentOptimizer - This optimizer is based on the gradient descent algorithm.

Metric – Language models are evaluated based on average perplexity across all words in a text. It is a transformation of cross-entropy: $PP = 2^{H(P_{LM})}$. This means that less probable words get higher perplexity values. When you compare perplexity values the ones with lower perplexity are better modelling the given language. Other than this, the quality of language model can be assessed directly in the application it is being used. In this project generating children stories.

Results

I got the following results, after experimenting with the different datasets that I have created.

Dataset	**Configuration**	**Epochs**	**Train Perplexity**	**Validation Perplexity**	**Test Perplexity**	**Training Time**
Dataset 1	**Small**	**13**	**74.165**	**88.735**	**103.004**	**26 hrs**
Dataset 2	**Small**	**6**	**71.404**	**87.562**	**99.450**	**12.38 hrs**
Dataset 3	**Small**	**6**	**74.998**	**168.931**	**168.436**	**1.61 hrs**
Dataset 3	**Medium**	**10**	**54.982**	**117.173**	**120.760**	**9.2 hrs**

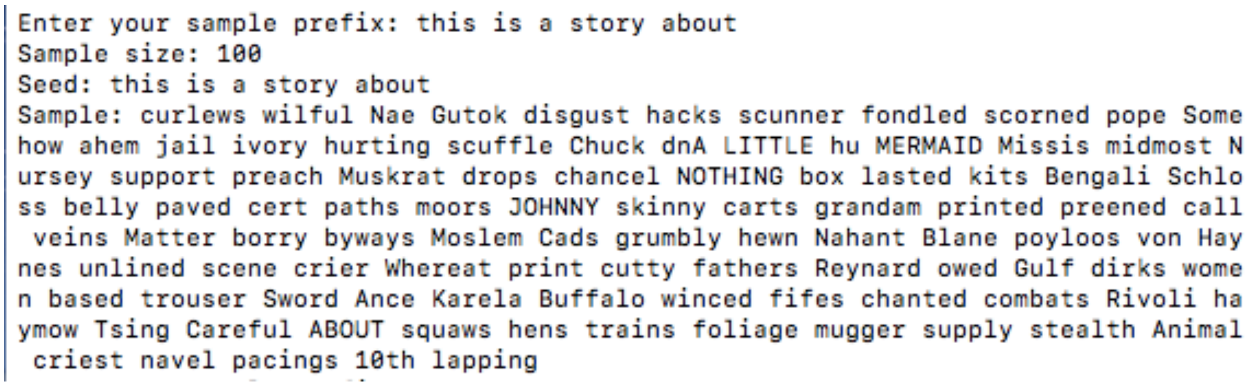
As you can see from the table, training over a long period of time gives better perplexity. In my first model, I observed that after about 7 epochs the perplexity didn't decrease and remained the same till epoch 13. Results on the second data were different than expected, I reduced the distinct words and the epoch size but the perplexity on the validation set and test set reduced. For the third and fourth datasets, after reducing the dataset considerably from 25 MB to about

4 MB the training time reduced drastically but the perplexity of validation and test set were high. Below I have posted some samples that were outputted by the different models. The stories don't seem to make any sense as a whole but few words put together do makes sense. So, I tried to predict the next word in a sentence and have posted the samples below. According to The Goldilocks Principle [3] LSTMs are excellent predictors of prepositions (on, at) and verbs (run, eat). I have tried to put this to test and see how my models work.
For example –
Seed - The boy came
Output – plaything excited slily

Similarly, I tried many sentences but they weren't very readable or didn't make any sense. I believe this is because of reducing the large number of words from the dataset and not training the model for longer periods of time with higher layers of LSTM cells and time steps. Even though this project did not produce great results. It gives an insight into the capabilities of LSTMs and RNNs in real world applications. This project also demonstrates how tools like TensorFlow can be used to generate recurrent neural network models.

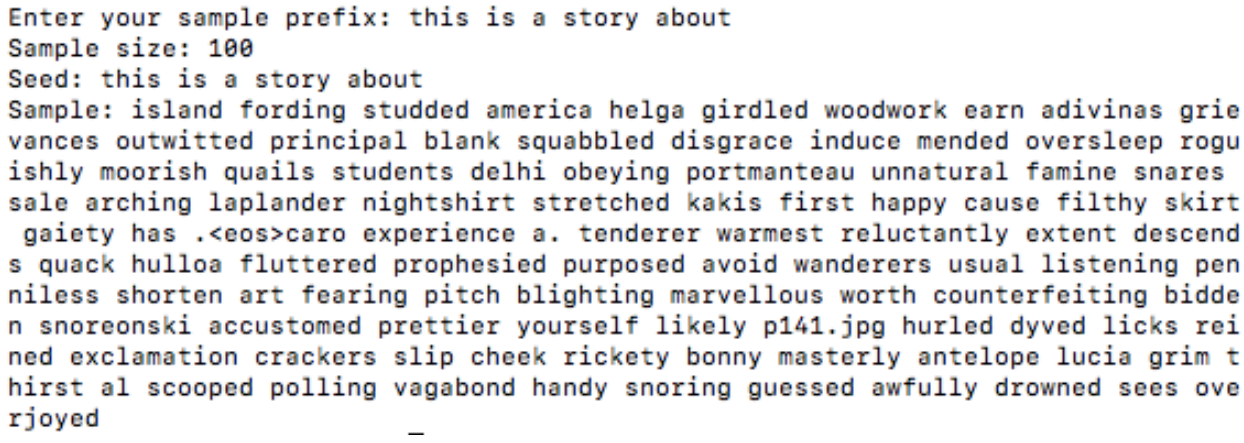
Samples

For result 1 -



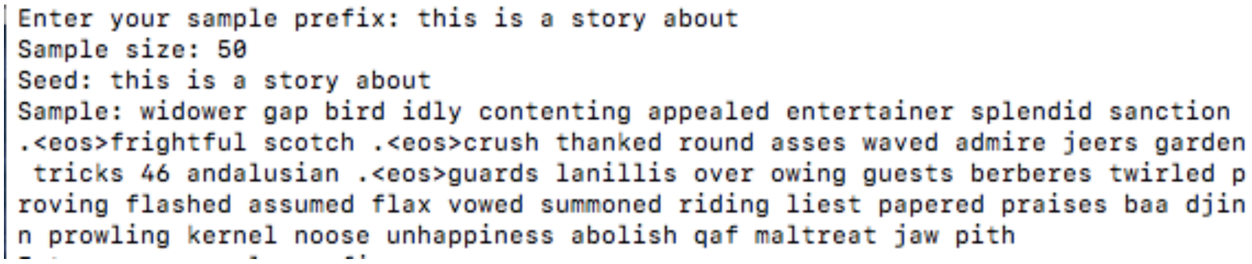
```
Enter your sample prefix: this is a story about
Sample size: 100
Seed: this is a story about
Sample: curlews wilful Nae Gutok disgust hacks scunner fondled scorned pope Some
how ahem jail ivory hurting scuffle Chuck dnA LITTLE hu MERMAID Missis midmost N
ursey support preach Muskrat drops chancel NOTHING box lasted kits Bengali Schlo
ss belly paved cert paths moors JOHNNY skinny carts grandam printed preened call
 veins Matter borry byways Moslem Cads grumbly hewn Nahant Blane poyloos von Hay
nes unlined scene crier Whereat print cutty fathers Reynard owed Gulf dirks wome
n based trouser Sword Ance Karela Buffalo winced fifes chanted combats Rivoli ha
ymow Tsing Careful ABOUT squaws hens trains foliage mugger supply stealth Animal
 criest navel pacings 10th lapping
```

For result 2 -



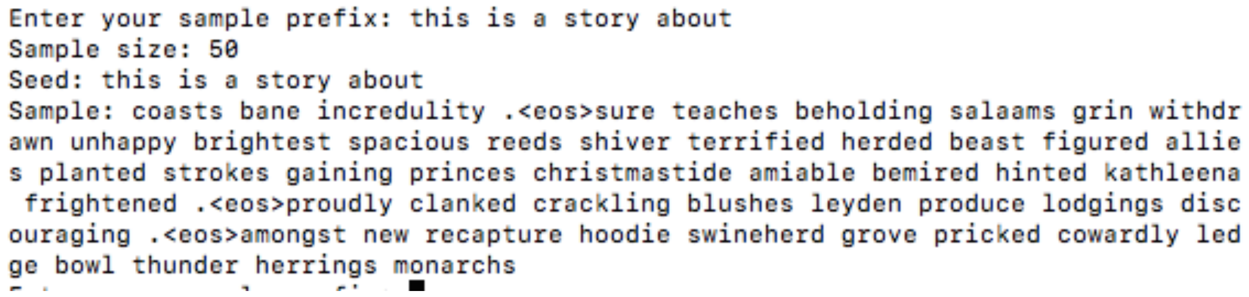
```
Enter your sample prefix: this is a story about
Sample size: 100
Seed: this is a story about
Sample: island fording studded america helga girdled woodwork earn adivinas grie
vances outwitted principal blank squabbled disgrace induce mended oversleep rogu
ishly moorish quails students delhi obeying portmanteau unnatural famine snares
sale arching laplander nightshirt stretched kakis first happy cause filthy skirt
 gaiety has .<eos>caro experience a. tenderer warmest reluctantly extent descend
s quack hulloa fluttered prophesied purposed avoid wanderers usual listening pen
niless shorten art fearing pitch blighting marvellous worth counterfeiting bidde
n snoreonski accustomed prettier yourself likely p141.jpg hurled dyved licks rei
ned exclamation crackers slip cheek rickety bonny masterly antelope lucia grim t
hirst al scooped polling vagabond handy snoring guessed awfully drowned sees ove
rjoyed
```

For result 3 –



```
Enter your sample prefix: this is a story about
Sample size: 50
Seed: this is a story about
Sample: widower gap bird idly contenting appealed entertainer splendid sanction
.<eos>frightful scotch .<eos>crush thanked round asses waved admire jeers garden
 tricks 46 andalusian .<eos>guards lanillis over owing guests berberes twirled p
roving flashed assumed flax vowed summoned riding liest papered praises baa djin
n prowling kernel noose unhappiness abolish qaf maltreat jaw pith
```

For result 4 –



```
Enter your sample prefix: this is a story about
Sample size: 50
Seed: this is a story about
Sample: coasts bane incredulity .<eos>sure teaches beholding salaams grin withdr
awn unhappy brightest spacious reeds shiver terrified herded beast figured allie
s planted strokes gaining princes christmastide amiable bemired hinted kathleena
 frightened .<eos>proudly clanked crackling blushes leyden produce lodgings disc
ouraging .<eos>amongst new recapture hoodie swineherd grove pricked cowardly led
ge bowl thunder herrings monarchs
```

Conclusion

1) LSTMs are better at the task of language modelling than RNNs.
2) Having a very large dataset of words and text for training language models takes long time and requires high computation effort.
3) Training models on smaller dataset and with fewer words do produce results but are not very meaningful.
4) There is no exact relation between how long the model is trained and the results. It is possible to achieve better results by training the model with right parameters for a lesser time as shown by result 2.

Reference

[1] A survey on language modeling using neural network. Nikolaos Pappas, Thomas Meyer.
[2] Recurrent neural network based language model. Mikolov, Tomas, Karafiat, Martin, Burget, Lukas, Cernocky, Jan and Khundanpur Sanjeev.
[3] The Goldilocks Principle: Reading Children's Books With Explicit Memory Representation. Felix Hill, Antoine Bordes, Sumit Chopra and Jason Weston
[4] Long Short-Term Memory. Sepp Hochreiter and Jurgen Schmidhuber