Neural Language Modelling using Recurrent Neural Networks

Madhukshara Sarkar (B18735)

M.Sc. Big Data Analytics

Ramakrishna Mission Vivekananda Educational and Research Institute

Objective:

Formal languages, like programming languages can be fully specified. All the reserved words can be defined and the valid ways that they can be used can be precisely defined. We cannot do this with natural languages. Natural languages are not designed; they emerge, and therefore there is no formal specification. There may be formal rules for parts of the language, and heuristics, but natural language that does not confirm is often used. Natural languages involve vast number of terms that can be used in ways that introduce all kinds of ambiguities, yet can still be understood by other humans.

Neural language models are a fundamental part of many systems that attempt to solve natural language processing tasks such as machine translation and speech recognition. Currently, all state of the art language models are neural networks.

In this assignment we try to build a Neural Language Model using LSTM and GRU. The dataset used is "The Works of Charles Dickens" collected from Project Gutenberg. There are 32 text files in the dataset, out of which the following 6 have been used:

1. The Magic Fishbone A Holiday Romance from the Pen of Miss Alice Rainbird, Aged 7
2. To be Read at Dusk
3. Hunted Down [1860]
4. The Seven Poor Travellers
5. A Message from the Sea
6. Some Christmas Stories

Methodology:

In this study we implement an LSTM, a bidirectional LSTM and a GRU networks to build language models. But first we clean and prepare the data before fitting the model to it. The pre-processing steps are as follows:

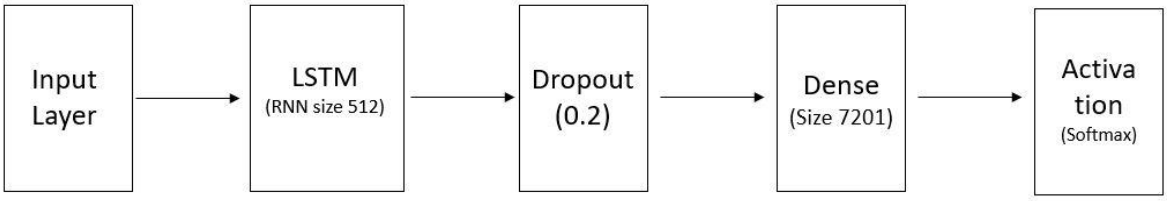
1. Data cleaning
    i) For cleaning the data we first remove the Gutenberg text from the data. This text basically contains the information about the text in context of the Gutenberg project.
    ii) Next we remove all special characters from the text and tokenize it, i.e. we split the entire text into separate words or tokens.
2. From all the tokens we now choose the most commonly used ones, and create a vocabulary. Note that our vocabulary contains only unique tokens. The size of our vocabulary is 7201.
3. Next we perform a word to index mapping, i.e. we assign a unique identifying index to each word in the vocabulary.
4. Then we generate sequences. Here we choose sequence length as 30. That is to say we create sequences of 30 words. The sequences are created in the following manner. Given a sequence of length 30, the next sequence is such that the first word of the previous sequence is dropped, and the next word in the text is added. Our data has 61,374 such sequences.
5. Finally we perform word embedding using one-hot-representation of each sequence and using the vocabulary.

Now we send the embedded data as input to our model. The model is trained such that, given an input sequence it produces the next sequence as output. Essentially, given a set of words, the model tries to predict the next word in the sequence.

We describe the three models used below.

Model 1: LSTM

The network architecture is as follows

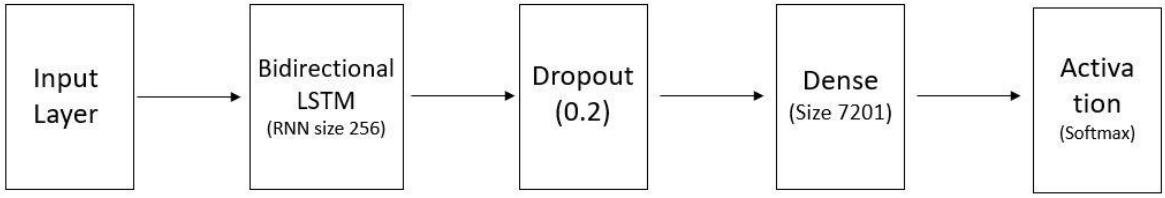


The model summary and number of parameters are as follows

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_2 (LSTM)                (None, 512)               15798272
_________________________________________________________________
dropout_2 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 7201)              3694113
_________________________________________________________________
activation_2 (Activation)    (None, 7201)              0
=================================================================
Total params: 19,492,385
Trainable params: 19,492,385
Non-trainable params: 0
_________________________________________________________________
None
```

Model 2: Bidirectional LSTM

The network architecture is as follows

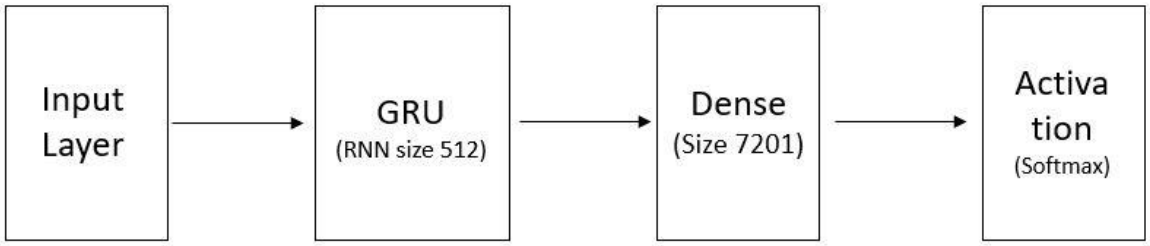


The model summary and number of parameters are as follows

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
bidirectional_1 (Bidirection (None, 512)               15273984
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 7201)              3694113
_________________________________________________________________
activation_1 (Activation)    (None, 7201)              0
=================================================================
Total params: 18,968,097
Trainable params: 18,968,097
Non-trainable params: 0
_________________________________________________________________
None
```

<u>Model 3</u>: GRU

The network architecture is as follows



The model summary and number of parameters are as follows

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru_1 (GRU)                  (None, 512)               11848704
_________________________________________________________________
dense_1 (Dense)              (None, 7201)              3694113
_________________________________________________________________
activation_1 (Activation)    (None, 7201)              0
=================================================================
Total params: 15,542,817
Trainable params: 15,542,817
Non-trainable params: 0
_________________________________________________________________
None
```

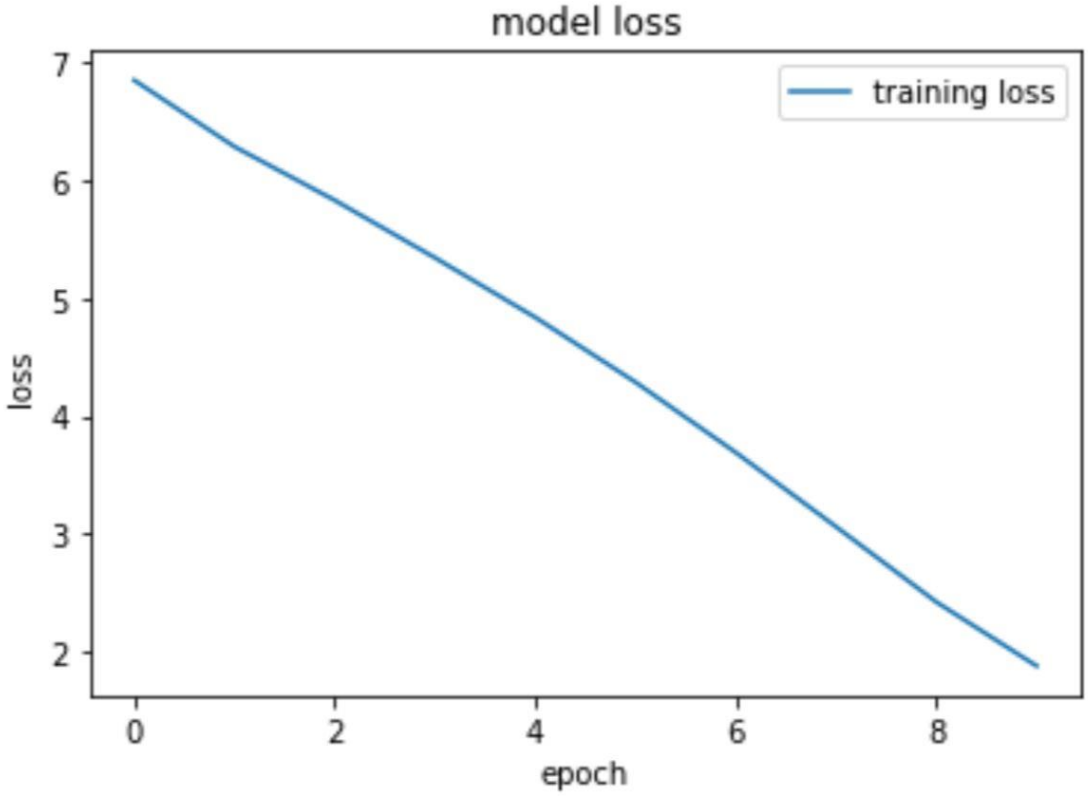
Besides these, the set of hyper-parameters used, and other model specifications are described below

Hyper-parameters	LSTM	Bidirectional LSTM	GRU
Batch Size	30	30	30
Number of Epochs	10	10	10
Dropout Rate	0.2	0.2	-
Learning Rate	0.001	0.001	0.001
Optimizer	Adam	Adam	Adam
Loss Criterion	Categorical Cross Entropy	Categorical Cross Entropy	Categorical Cross Entropy

Results and Discussion:

Model 1: LSTM

The loss vs epochs plot is as follows



Minimum loss after 10 epochs ≈ 1.8

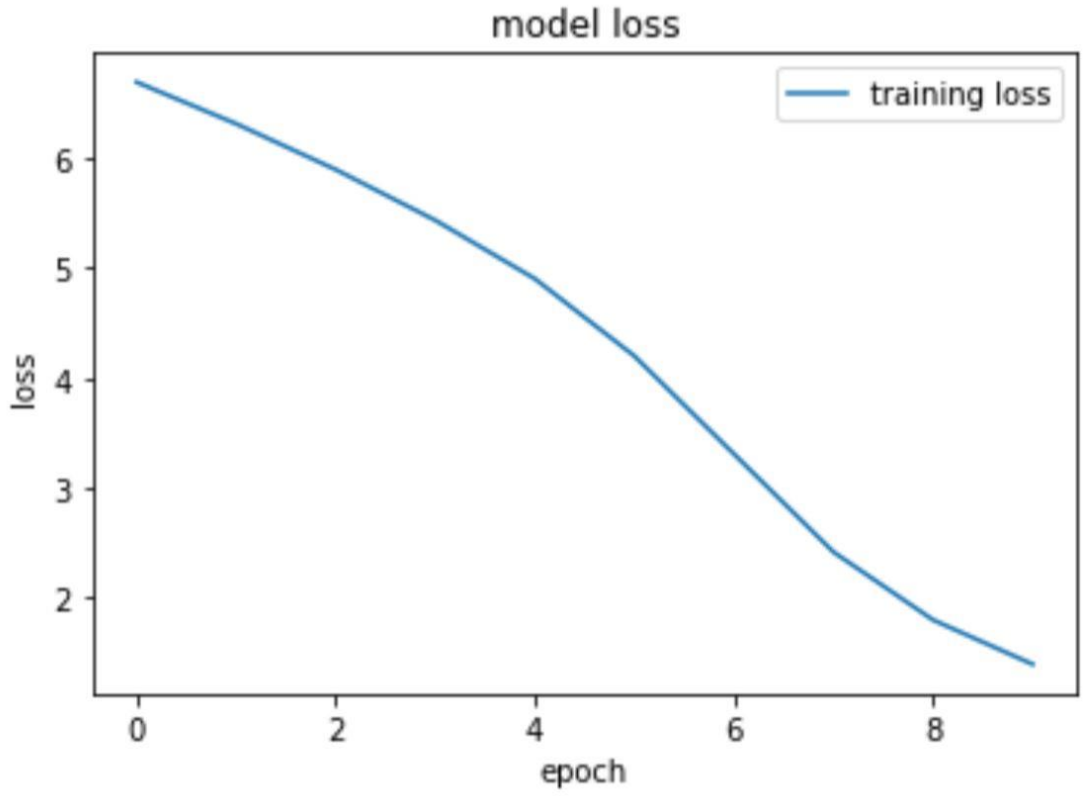
Generated text:

“reluctance to trouble men of business with inquiries for friends knowing the probabilities to be a thousand to one that the friends will never follow them up people are so fickle so selfish so inconsiderate don’t you in your time it had been seen by any one and in the sons of night and sat down on the room with a large window window whence i could see the other that the breakfast and within the table were all were”

Clearly, the generated text is not very semantically or grammatically sound. We cannot extract any meaning sentence or sequence from this generated text.

Model 2: Bidirectional LSTM

The loss vs epochs plot is as follows



Minimum loss after 10 epochs ≈ 1.3

Generated text:

“you down you have been tracked to death at a single individual’s charge i hear you have had the name of meltham on your lips sometimes ’ i saw in addition to those other changes a stoppage stoppage coloured his hand has support of lovely and i should begin to murdered you unpromising with lovely and i reckon climbed up again and probabilities you at a word of his life veins beatified crafty that question but fisherman may at the”

The generated text is not semantically, or grammatically very correct. However we can extract some snippets of meaningful text from it, such as

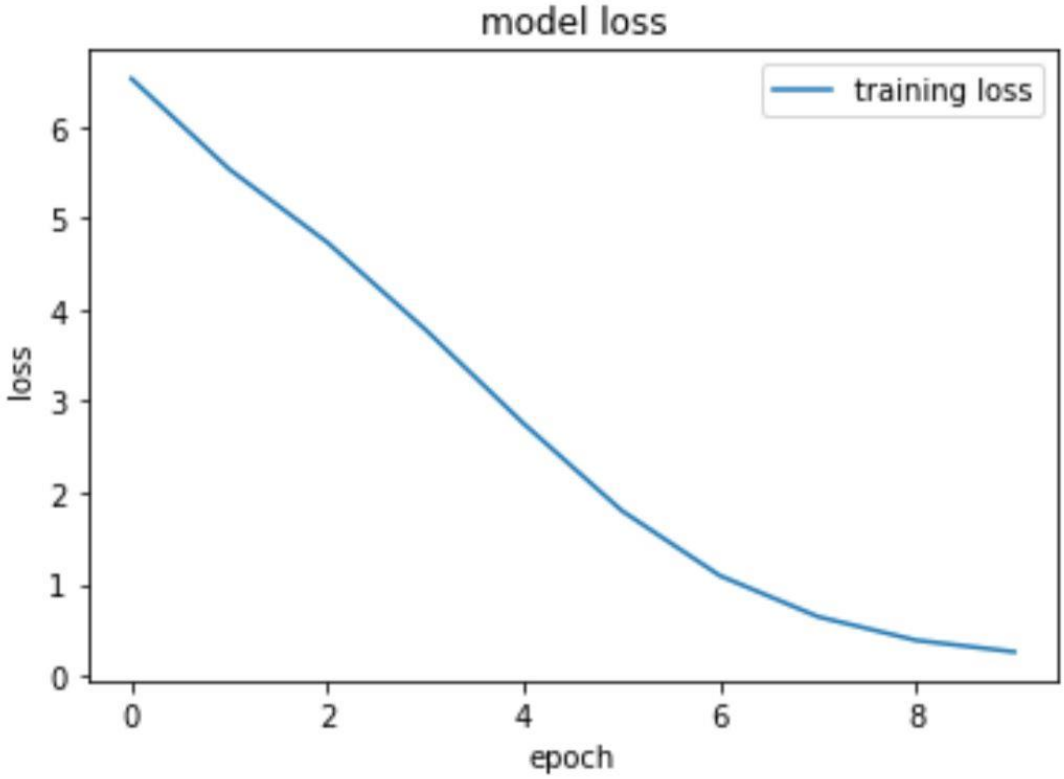
“you have been tracked to death at a single individual’s charge”

“i hear you have had the name of meltham on your lips”

“sometimes ’ i saw in addition to those other changes a stoppage”

<u>Model 3</u>: GRU

The loss vs epochs plot is as follows



Minimum loss after 10 epochs ≈ 0.3

Generated text:

"' said the german 'i should probably know a great deal more ' it was a good answer i thought and it made me curious so i moved my position to that corner of my inn boys and they were brought together at once a most exact and punctual manner i never saw my finer turkey finer beef or greater prodigality of sauce and gravy and my travellers did wonderful justice to everything set before them it made my heart"

Here, the text generated, though has some grammatical errors, but we can extract 3 separate meaningful texts from this entire text. They are:

"said the german 'i should probably know a great deal more ' it was a good answer i thought and it made me curious so i moved my position to that corner of my inn"

"they were brought together at once a most exact and punctual manner"

"i never saw my finer turkey finer beef or greater prodigality of sauce and gravy and my travellers did wonderful justice to everything set before them it made my heart"

So we can conclude that the GRU model is capable of generating a meaningful text. The model can be made much better by fine tuning the hyper-parameters, adding multiple layers to the network, adding a bidirectional layer to better capture the semantic relations between words and phrases, etc.

Conclusion:

The objective of the study was to develop a language model using some RNN architecture. We used 3 different networks: LSTM, Bidirectional LSTM and GRU.

Based on our models, from the loss vs epoch plots, and more importantly from the generated texts we can say that the GRU network acts as the best language model. However, the model can be subjected to further improvements.

Limitations:

Some limitations of our study are as follows

1. We could not use more number of files. The dataset had 32 text files. Had we trained our model on more amount of text, our model would have been able to generate much better results.
2. We could not increase the number of epochs. Obviously, more number of epochs would make the model better.
3. We could not use deeper networks with multiple layers and greater RNN sizes.