Text Generation with LSTM

Avishek Banerjee, Arnab Banerjee , Amtoj Sandhu

September 10, 2019

Abstract

Prose and poetry has been a backbone of the English literature for centuries expressing feelings, emotions and experience. However with ages the old literature and style got converted into contemporary modern literature. With the expansion of machine learning and neural network can we generate back the same kind of poetry and text? This work focus on generating texts and prose based on the character level and word level models. We used bidirectional LSTM to train our model and we generated some sample examples. We also organized a social experiment where around 30 percent of the participants predicted our output as original human writings.

1 Text generation using character-based LSTM

1.1 Preprocessing of data

We have selected the novel by Leo Tolstoy titled War and Peace as the dataset for our model. Before training, we map the characters to a numerical representation and create two lookup tables: one mapping characters to numbers, and another for numbers to characters. [4] [1]

```
 {
  'D' :  16,
  'X' :  36,
  'R' :  30,
  'b' :  40,
  'x' :  62,
  'A' :  13,
  ':' :  10,
  '.' :   8,
}
```

The input to the model will be a sequence of characters, and we train the model to predict the following character at each time step. Example: Consider the sentence: John is playing golf. Then the input and output for our LSTM will be of the form:

```
INPUT:  John is playing gol
OUTPUT: f
```

We take the input set as the array of character sequences of length 40, encoded in their numerical representation and chunked in steps of 3 characters from our corpus text, and the output set as the next character of the 40 character sequence. In order to reduce training time, we have only converted the dataset into lower case characters which provides us with a vocabulary of 69 characters after considering the punctuation symbols, space, ASCII and new-line characters.

1.2 Model

We start with a sequential model from keras library and add two LSTM hidden layers with 160 units. Next we use a dense layer with 69 units representing each character in the vocabulary. The dense layer has softmax as the activation function for performing the multi-class classification to predict the next character in a sequence. The RNN is trained using the AdamOptimizer to optimize the weights in our model which combines the benefits of both AdaGrad and RMSProp and we use *sparse_categorical_crossentropy* as our loss function.

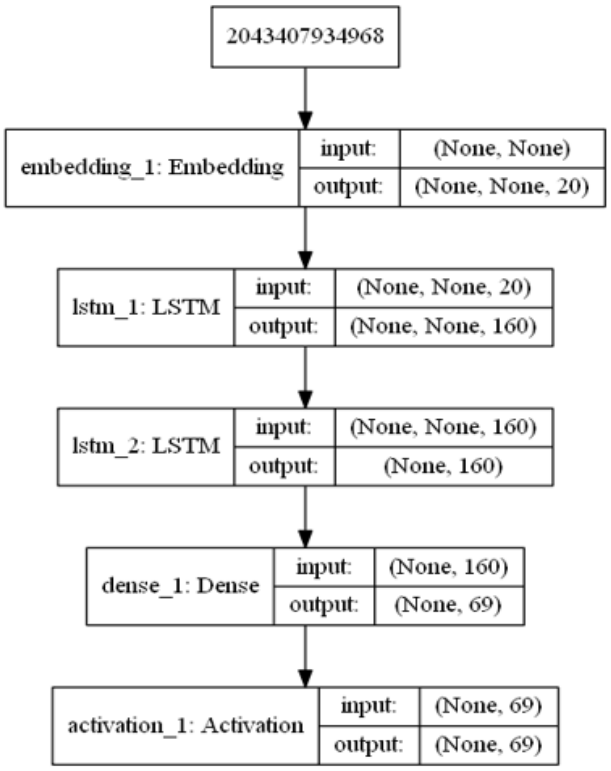

Figure 1: Model

1.3 Text Generation

We input a feed string into the model, initializing the RNN state and setting the number of characters to generate and then get the prediction distribution of the next character over the character vocabulary and use a multinomial distribution to calculate the index of the predicted character and append this character to input character sequence while dropping the first character of the input sequence to make the new input sequence for the next character prediction. We also use a scaling factor called temperature applied to the outputs of our dense layer before applying the softmax activation function. In a nutshell, it defines how conservative or creative the model's guesses are for the next character in a sequence. Lower values of temperature (e.g., 0.2) will generate "safe" guesses whereas values of temperature above 1.0 will start to generate "riskier" random guesses.

1.4 Tuning the model

1. **Temperature** Lower temperature makes the model more confident, but also more conservative in its samples of characters resulting in repetitive phrases. Conversely, higher temperatures will give more diversity but at cost of more mistakes (e.g. spelling mistakes). Taking a median value of temperature like 0.5 or 0.6 gives good results as manage to limit both shortcomings to a certain extent.

2. **Number of LSTM layers** Increasing number of LSTM hidden layers from 1 to 2 leads to a significant decrease in training loss from 1.32022 to 1.20648 when the model is trained for 20 epochs.

3. **Number of units in LSTM layer** The single hidden layer LSTM layer consisting of 128 units model trained for 20 epochs yielded a training loss of 1.39471 and increasing the number of units in the LSTM layer to 160 helped decrease the training loss to 1.32022.

The best performance when trained for 20 epochs was obtained by the model with two LSTM layers with 160 units each.

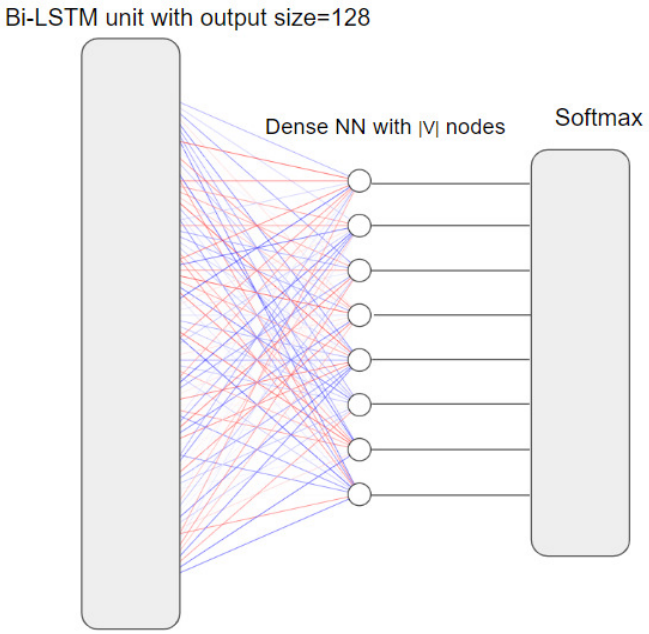

Figure 2: Word Based Bidirectional LSTM model

2 Word based Bidirectional LSTM

2.1 Idea of the algorithm

The bidirectional LSTM will be trained using a supervised manner. A sequence of words of fixed length from the corpus is fed to the model as input and the next word of that sequence is the target word for the model. The model learns the weights to maximize the probability of predicting the target word for the model. The set of all unique words in our training text forms the vocabulary from which the model selects the words. [5]

2.2 Bidirectional LSTM with One Hot Encoding

2.2.1 Input String representation

Firstly, a vocabulary of words is created using the words present in the text corpus. In order to avoid creating a large vocabulary, words are only inserted in the vocabulary if the frequency of those words in the text corpus exceeds a minimum frequency. This hyperparameter is called MinWordFreq In this work we experimented with 10 and 5 as the variable minimum frequency to generate the vocabulary.

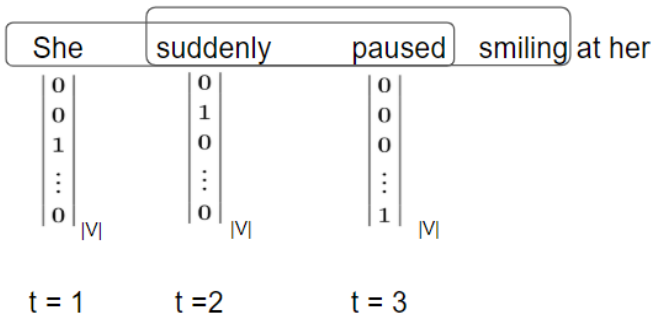

Figure 3: Input representation

The entire text corpus is then broken up into multiple word sequences of fixed length. As shown in Fig 3 "She suddenly paused" is one such sequence". The corresponding target word for this sequence would be the very next word after the sequence. As shown in Fig 3 it is the word "smiling". Now each word in the sequence is converted into a vectorized one hot encoding as shown in Fig 3. The whole sequence along with the one-hot encoded version of target word is fed into the model. The next sequence would get started by sliding one word to the right of the previous sequence. This continues until the whole text is fed to the model.

2.2.2 Model structure

As shown in Fig 2 the first layer in the model is a bidirectional LSTM. The dimension of the LSTM unit used is 128. That layer is followed by a dense Neural network layer. This last layer consists of as many nodes as the vocabulary size. SoftMax activation layer is used on top of this layer to output probability of the next word being the word represented by the node. To prevent overfitting a dropout variable is introduced whose value can be varied so that the model learns the weights without overfitting. By default, its value is set to 0.2. Our model used Adams optimizer to train the weights and the loss was calculated by categorical_crossentropy. The idea is that after many epochs this layer will learn the style of how the corpus is written, trying to adjust the weights of the network to predict the next word.

2.3 Bidirectional LSTM with embedding layer

It is similar to the previous model described however it has an extra embedding layer as shown in Fig 4. Each of the words in the vocabulary will be mapped to a positive index. For example the will be mapped to 1, is will be mapped to 2 and similarly all other words will be mapped to an index. The input representation of the string war is destruction will be an array $[3, 2, 5]$. This array will be fed to the model as input.[5]

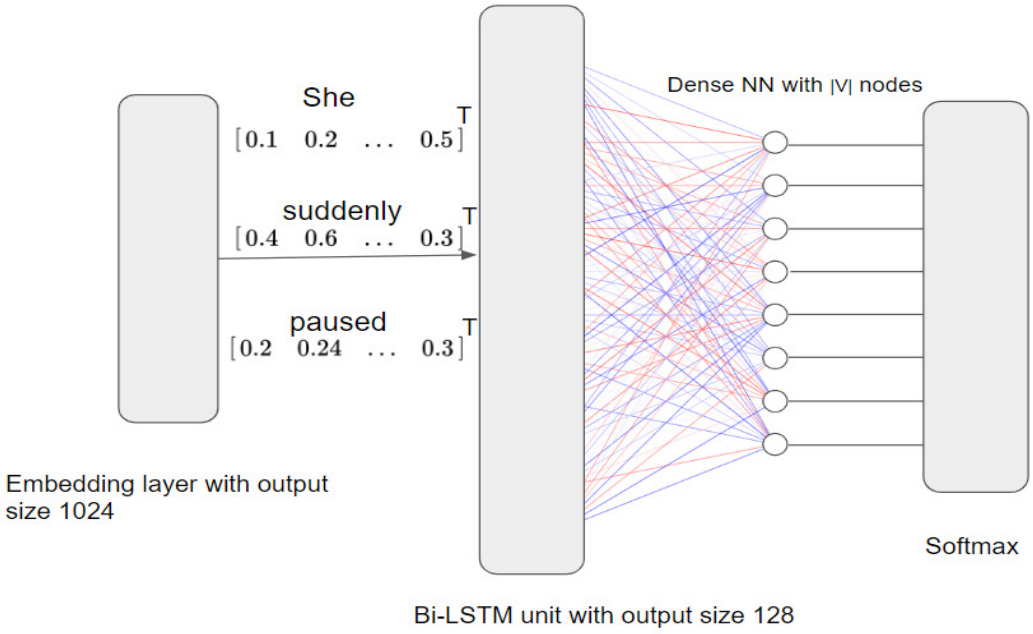

Figure 4: Word Based Bidirectional LSTM model with embedding layer

The embedding layer maps positive integers (indexes) of the words in the vocabulary into dense vectors of fixed size. This layer is required when sparse vector representations of words using one-hot encoding is to be avoided. This step is particularly useful with large vocabulary size. In our case, the dense vectors have a fixed size of 1024. This layer can only be used as the first layer in a model.

2.4 Text Generation

Once the training is done, a new sentence of the same fixed length as used during training is fed into the model as shown in Fig 5 Upon prediction of a new word, the first word of the sentence is removed and the newly predicted

word is appended to the sentence which is again fed into the model for prediction of the next word. This process goes on for the number of words that are desired to be generated. As discussed in the character-based LSTM we use the temperature hyperparameter here to diversify the possibilities of words generated by the model.

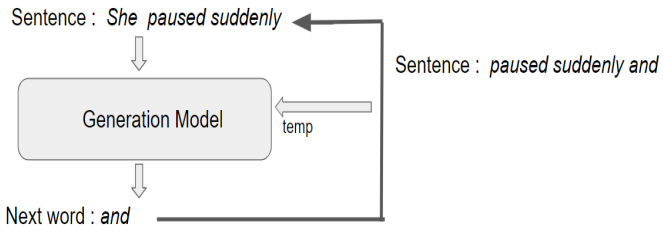

Figure 5: Text Generation

3 Training and Testing

For training our model, we downloaded the text version of the book War and Peace by Leo Tolstoy [3]. The book consists of 0.56 million words and 3 million characters.

We divide the entire text corpus into training set and a validation (or testing) set in an 80-20 ratio. So, the model is trained on the training set and we measure the test accuracy on the test set. We try out different values of Batch size, MinWordFreq in order to find the configuration with the highest validation accuracy

4 Results

4.1 Character Based LSTM

After training the model for 15 epochs and using the feed string: it is true that it may be quite possible and using the range of temperature values we generate the following text:

temperature=0.02

it is true that it may be quite possible **the saint-germeries of the cardinal and the saint-germeries of the cardinal and the saint-germeries of the cardinal and the saint-germeries of the stone of the cardinal and the saint-germeries**

temperature= 0.5

it is true that it may be quite possible **could to the cross with the challer as the devil of the man constraed his heady and the chanter, and the things with a gold. the man to the sark of the archdeacon. the man was all passed him architecture of the bell with the raim that with a spectacs of the wither, and the art of the ray of monsieur**

temperature=1.0

it is true that it may be quite possible, **having happined but dease it had fells out, while gringoire crossible, sepitule. this window afned the expont of a very sover volumbs besturned to up a fly foot close her bewerged, which he good law or is same on good. fiving jehan on the tdies; a final bell of which misalul**

4.2 Word Based LSTM

4.2.1 One hot encoding

Word freq = 10 Batch size = 32

It is true that it may be quite possible **that the emperor would not be able to find a man in the world to be a bad man in this way he thought of the words of the old man in the world he is**

Word freq = 5 Batch size = 64
It is true that it may be quite possible **that he is impossible to do so well the emperor was the first thing and the general who had been sent to the emperor was not captured.**

4.2.2 Embedding Layer

Word freq = 10 Batch size = 32
it is true that it may be quite possible **that the man is in love of looking at his mother and all that she had grown and she was now and still feeling that she was to love her she really saw her mother and that he were looking at her**
Word freq = 5 Batch size = 64
it is true that it may be quite possible **that he is not in it he is too long in a position my business he went up to the floor while prince andrew fell into the shed and the little princess who came down to the farther side of the wood he saw that it was a long time a few time in the**

The following table shows the test accuracy of the two different word based LSTM models with different configurations of MinWordFreq and batchsize.

ConfigurationNo	MinWordFreq	BatchSize	Test Accuracy- One Hot encoding	Test Accuracy- Word embedding
1	10	32	0.2096	0.2114
2	5	32	0.2092	0.2118
3	10	64	0.2153	0.2125
4	5	64	0.2151	0.2130

Table 1: Test Accuracy for word based LSTM models

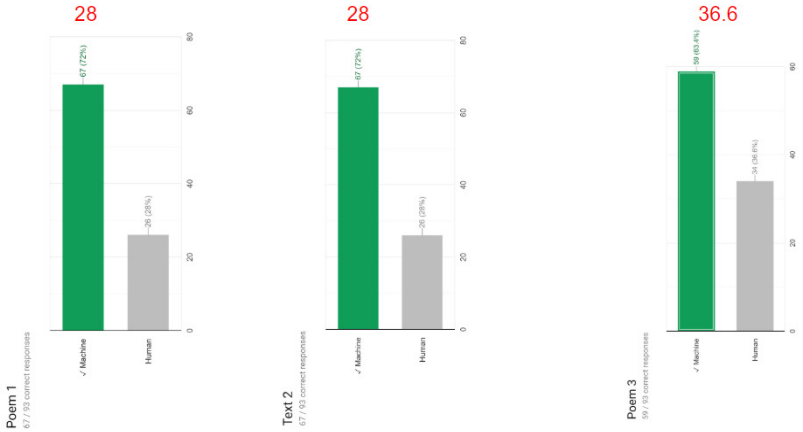

Figure 6: Social Experiment Results

4.3 Observations & Conclusions

1. The performance scores of character based LSTM model is well below the word based Bi-directional LSTM model because of spelling mistakes and incorrect wordings. The performance scores were generated from the Grammarly application based on the grammar and correctness of the English words.

2. From Table 1 we can observe that the test accuracy for higher batch size and higher MinWordFreq value is slightly higher. The reason can be attributed to lower vocabulary size due to higher MinWordFreq cutoff. Also, One-hot encoding slightly outperforms the embedding model with higher batch size.

3. The test accuracy saturates for both the word based LSTM models and subsequently overfits. Since the vocabulary is large, predicting the correct word from the vocabulary during testing is challenging.

4. Adding the embedding layer does not lead to a significant rise in test accuracy. Maybe a different neural architecture coupled with embedding layer will give a higher test accuracy.

4.4 Social Experiment

We conducted a social experiment having 6 questions, 3 generated by human and 3 by our model. As in Fig 6 we found around 30 percent of 100 people mistook our model as the human generated ones thus providing a realistic touch to our model. The quiz is still open and one can participate. [2]

References

[1] Beginners guide to text generation using lstms. (n.d.)., https://www.kaggle.com/shivamb/beginners-guide-to-text-generation-using-lstms.

[2] https://bit.ly/2gj79bw. In *Social Experiment form.*

[3] http://www.gutenberg.org. In *War Peace.*

[4] Karpathy, a. the unreasonable effectiveness of recurrent neural networks, http://karpathy.github.io/2015/05/21/rnn-effectiveness/.

[5] Lstm lyrics, https://github.com/enriqueav/lstm$_lyrics$.