Vidyabagish

MLND - Capstone Project Report

Kabya Basu

(A) Definition

Project Overview:

Deep learning (also known as deep structured learning or hierarchical learning) is the application of artificial neural networks (ANNs) to learning tasks that contain more than one hidden layer. Deep learning is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms.With Deep Learning, it is now possible for an algorithm to predict things, classify images (objects) with great accuracy, detect fraudulent transactions, generate image, sound and text. These are tasks that were previously not possible to achieve by an algorithm and now perform better than a human.

In this project we will focus on Text Generation. Text Generation is part of Natural Language Processing and can be used totranscribe speech to text, performmachine translation, generate handwritten text, image captioning, generate new blog posts or news headlines.

In order to generate text, we will look at a class of Neural Network where connections between units form a directed cycle, called Recurrent Neural Network (RNNs). RNNs use an internal memory to process sequences of elements and is able to learn from the syntactic structure of text. Our model will be able to generate text based on the text we train it with.

Problem Statement:

Ramchandra Vidyabagish, was an Indian lexicographer,writter and Sanskrit scholar. He is known for his Bangabhashabhidhan, the first monolingual Bengali dictionary, published in 1817.

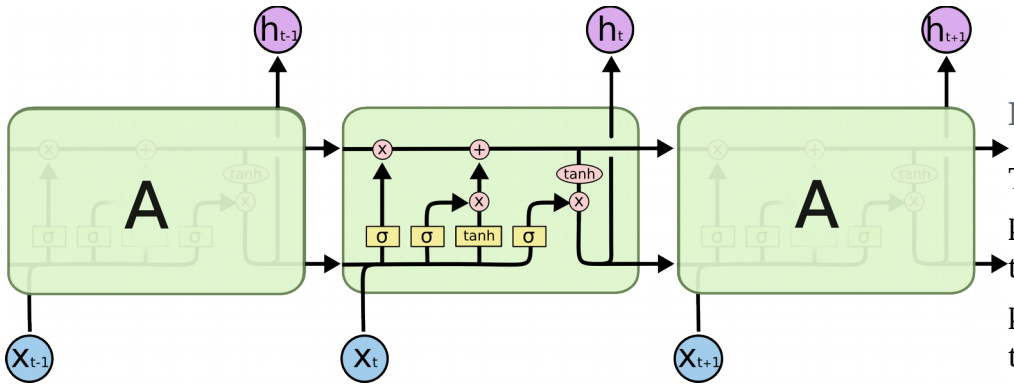
Unfortunately, Vidyabagish passed away 100 years ago and he will not be publishing new novels. But, wouldn’t it be great if we could generate some text inspired on Jyotish Sangrahasar and other novels he published?

To solve our problem, we can use text from novels written by Vidyabagish in combination with the incredible power of Deep Learning, in particular RNNs, to generate text. Our deep learning model

LSTM Cell

For our model to learn, we will use a special type of RNN called LSTMs (Long Short Term Memory), capable of learning long-term dependencies. LSTM can use its memory to generate complex, realistic sequences containing long-range structure, just like the sentences that we want to

generate. It will be able to remember information for a period of time, which will help at generating text of better quality.



Metrics:

The purpose of this project, is to create a model that will be able to generate text inspired in novels written by Vidyabagish.

The performance of our model will be measure by:

- Perplexity is a commonly used evaluation metric when generating text. Perplexity tells us how many words is the model considering as an output, having a perplexity of 3 means that the model could choose 3 words at an equally likely probability, we want our perplexity be as low as possible since lower choice corresponds to a higher likelihood of choosing the actually correct one.

  The typical measure reported in the papers is average per-word perplexity (often just called perplexity), which is equal to

$$e^{-\frac{1}{N}\sum_{i=1}^{N}\ln p_{\text{target}_i}} = e^{\text{loss}}$$

  Our goal is to achieving a **perplexity of less than 3.** Which is lower than the perplexity achieved by similar models used for text generation.

- Grammatically our model is able to:
  - Open, close quotations
  - Sentence length is similar to the median sentence length of the dataset. We will use the median as there is a large number of empty sentences (between paragraphs, separating chapters, after a title), which can skew our data. See histogram below.
  - Create paragraphs

(B) Analysis

Data Exploration and Exploratory Visualization:

Dataset

To train our model we will use the text from his novel Jyotish Sangrahasar and Bachaspati Mishrer Vivadachintamanih. All the novels are no longer protected under copyright and thanks to David Hare for providing the pdf file of these book.

Even though Vidyabagish native language was Bengali, the text used to train our model will be in English. This is to make it easier for the reader to understand the input and output of our model.

Our Dataset is small as it is composed of only 2 files - Jyotish Sangrahasar and Bachaspati Mishrer Vivadachintamanih with a total size of 3.4 MB. Bigger datasets work better when training an RNN but for our case that is very specific it will be enough. Some additional information of the contents of the files below:

Name	Size	Pages	Lines	Words	Unique Words
Jyotish_Sangrahasar.txt	2.3 MB	690	40,008	429,256	42,154
Bachaspati_Mishrer_Vivadachintamanih.txt	1.1 MB	303	17,572	189,037	

- Note: Values in the table above will change after preprocessing.

There is some manual preprocessing that we will need to do as the text retrieved from Gutenberg Project contains additional content that is not necessary to train the model, for example:

- Preface
- Translator’s Preface
- About the author
- Index
- Dedications
- Footnotes included in Exemplary Novels

Note: The files included in the dataset folder no longer contain the additional content mentioned above.

Exploring our Datasets

Lets extract text from our Datasets to get familiar with the data that we will be processing.We can see that our sentences are formed of 13 / 14 words. Paragraphs contain 5 or more sentences.

Jyotish_Sangrahasar.txt

himself up to reading books of chivalry with such ardour and avidity that he almost entirely neglected the pursuit of his field-sports, and even the management of his property; and to such a pitch did his eagerness and infatuation go that he sold many an acre of tillageland to buy books of chivalry to read, and brought home as many of them as he could get. But of all there were none he liked so well as those of the famous Feliciano de Silva's composition, for their lucidity of style and complicated conceits were as pearls in his sight, particularly when in his reading he came upon courtships and cartels, where he often found passages like "the reason of the unreason with which my reason is afflicted so weakens my reason that with reason I murmur at your beauty;" or again, "the high heavens, that of your divinity divinely fortify you

with the stars, render you deserving of the desert your greatness deserves." Over conceits of this sort the poor gentleman lost his wits, and used to lie a

Bachaspati_Mishrer_Vivadachintamanih.txt

t all the pains they took for that purpose proved vain, and the wishes they had felt on the subject gradually diminished, as the attempt appeared more and more hopeless. Thus, devoted to their studies, and varying these with such amusements as are permitted to their age, the young men passed a life as cheerful as it was honourable, rarely going out at night, but when they did so, it was always together and well armed.

One evening, however, when Don Juan was preparing to go out, Don Antonio expressed his desire to remain at home for a short time, to repeat certain orisons: but he requested Don Juan to go without him, and promised to follow him.

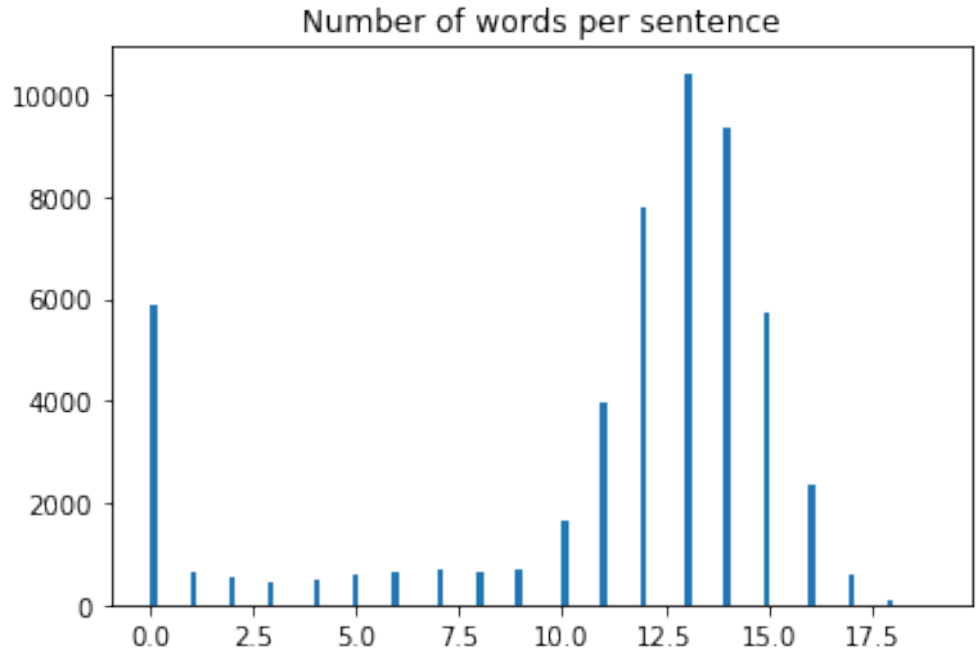
"Why should I go out to wait for you?" said Don Juan. "I will stay; if you do not go out at all to-night, it will be of very little consequence." "By no means shall you stay," returned Don Antonio: "go and take the air; I will be with you almost immediately, if you take the usual way."

"Well, do as you please," said Don Juan: "if you come you

Dataset Stats and Exploratory Visualization

As explained in the Metrics section, we can see that there is a large number of empty sentences in our dataset.

More stats:



- Unique words: 39229
- Number of chapters: 135
- Average number of sentences in each chapters: 392.7925925925926
- Number of lines: 53162
- Median number of words in each line: 13.0
- Average number of words in each line: 10.975941461946503

Algorithms and Techniques

Here we will be building a recurrent neural network (RNN) model.

Recurrent Neural Network

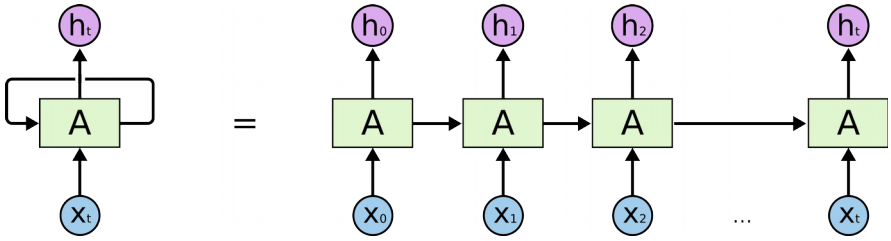
Humans don’t start their thinking from scratch every second. As you read this essay, you understand each word based on your understanding of previous words. You don’t throw everything away and start thinking from scratch again. Your thoughts have persistence.

Traditional neural networks can’t do this, and it seems like a major shortcoming. For example, imagine you want to classify what kind of event is happening at every point in a movie. It’s unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones.

Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

In the above diagram, a chunk of neural network, A, looks at some input xt and outputs a value ht. A loop allows information to be passed from one step of the network to the next.

These loops make recurrent neural networks seem kind of mysterious. However, if you think a bit more, it turns out that they aren’t all that different than a normal neural network. A recurrent neural network can be thought of as multiple copies of the same network, each passing a message to a successor. Consider what happens if we unroll the loop:



An unrolled recurrent neural network.

This chain-like nature reveals that recurrent neural networks are intimately related to sequences and lists. They’re the natural architecture of neural network to use for such data.

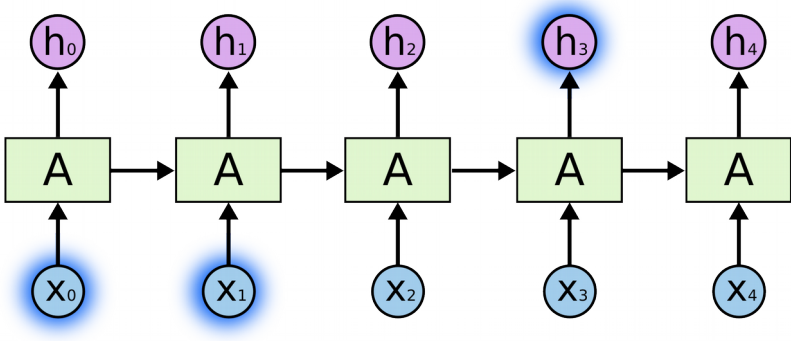
And they certainly are used! In the last few years, there have been incredible success applying RNNs to a variety of problems: speech recognition, language modeling, translation, image captioning… The list goes on. I’ll leave discussion of the amazing feats one can achieve with RNNs to Andrej Karpathy’s excellent blog post, The Unreasonable Effectiveness of Recurrent Neural Networks. But they really are pretty amazing.

Essential to these successes is the use of “LSTMs,” a very special kind of recurrent neural network which works, for many tasks, much much better than the standard version. Almost all exciting results based on recurrent neural networks are achieved with them. It’s these LSTMs that this essay will explore.

The Problem of Long-Term Dependencies

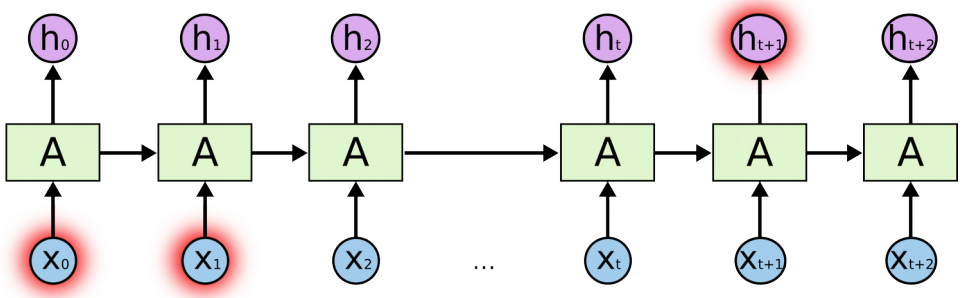
One of the appeals of RNNs is the idea that they might be able to connect previous information to the present task, such as using previous video frames might inform the understanding of the present frame. If RNNs could do this, they'd be extremely useful. But can they? It depends.

Sometimes, we only need to look at recent information to perform the present task. For example, consider a language model trying to predict the next word based on the previous ones. If we are trying to predict the last word in "the clouds are in the sky," we don't need any further context – it's pretty obvious the next word is going to be sky. In such cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.



But there are also cases where we need more context. Consider trying to predict the last word in the text "I grew up in France… I speak fluent French." Recent information suggests that the next word is probably the name of a language, but if we want to narrow down which language, we need the context of France, from further back. It's entirely possible for the gap between the relevant information and the point where it is needed to become very large.

Unfortunately, as that gap grows, RNNs become unable to learn to connect the information.



In theory, RNNs are absolutely capable of handling such "long-term dependencies." A human could carefully pick parameters for them to solve toy problems of this form. Sadly, in practice, RNNs don't seem to be able to learn them. The problem was explored in depth by Hochreiter (1991)

[German] and Bengio, et al. (1994), who found some pretty fundamental reasons why it might be difficult.

Thankfully, LSTMs don’t have this problem!

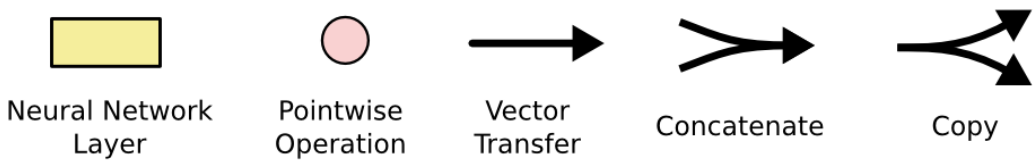
LSTM Networks

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997), and were refined and popularized by many people in following work.1 They work tremendously well on a large variety of problems, and are now widely used.

LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTMs also have this chain like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



In the above diagram, each line carries an entire vector, from the output of one node to the inputs of others. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denote its content being copied and the copies going to different locations.

Building of Vidyabagish Neural Network

A GPU is suggested to train the Vidyabagish Neural Network as text generation takes a long time to train. The building blocks of the Vidyabagish Neural Network are includ inVidyabagish_nn.py.

Functions included in Vidyabagish_nn:

- get_inputs: Creates the TF Placeholders for the Neural Network
- get_init_cell: Creates our RNN cell and initialises it.
- get_embed: Applies embedding to our input data.
- build_rnn: Creates a RNN using a RNN cell
- build_nn: Apply embedding to input data using your get_embed function. Builds RNN
- using cell and the build_rnn function. Finally, it applies a fully connected layer with a

- linear activation.
- get_batches: Creates a generator that returns batches of data used during training

Building our Neural Network

The RNN cells used by our model are LSTM -

https://www.tensorflow.org/tutorials/recurrent#lstm , which will process a word at a time and will calculate the probability of the next word when predicting.

To give the model more power, we can add multiple layers of LSTMs to process the data. In our Neural Network we use 2/3 RNN layers of size 256. The output of the first layer will become the input of the second and so on. To achieve this, we use a MultiRNNCell.

In order to prevent overfitting, we use dropout between the LSTM layers.

To calculate our models loss, we use tf.contrib.seq2seq.sequence_loss, which is the

weighted average cross-entropy (log-perplexity) per symbol.

We want to minimize the average negative log probability of the target words:

$$loss = -\frac{1}{N}\sum_{i=1}^{N} \ln p_{\text{target}_i}$$

The typical measure reported in the papers is average per-word perplexity (often just called perplexity), which is equal to (equation below) and we will monitor its value throughout the training process.

$$e^{-\frac{1}{N}\sum_{i=1}^{N} \ln p_{\text{target}_i}} = e^{\text{loss}}$$

To optimise our model we use AdamOptimizer.

Benchmark

How does our model performs when compared to our initial expectations (metrics/benchmark)?

- Our benchmark was to create a model that achieves a perplexity of less than 3. Our model is achieving a perplexity of less than 1. Which means, it has a high likelihood of choosing the correct word.
- The paragraph structure generated by our model is similar to the structure from our dataset.
  (a) The sentences in the paragraph have betwen 10 and 15 words.
  (b) The paragraphs are formed of 5 or more lines.

We discussed this again in the result section with more detail.

(c)Methodology

Data Preprocessing

We need to prepare our data for our RNN, lets do some additional preprocessing:

- Lookup table: We need to create word embeddings to facilitate the training of our model.
- Tokenize punctuation: This is to simplify training for our neural network. Making it easy for it to distinguish between write and write!

Tokenize Punctuation

We'll be splitting the script into a word array using spaces as delimiters. However, punctuations like periods and exclamation marks make it hard for the neural network to distinguish between the word "mad" and "mad!".

Implement the function token_lookup to return a dict that will be used to tokenize symbols like "!" into "||Exclamation_Mark||". Create a dictionary for the following symbols where the symbol is the key and value is the token:

- Period (.)
- Comma (,)
- Quotation Mark (")
- Semicolon (;)
- Exclamation mark (!)
- Question mark (?)
- Left Parentheses (( )
- Right Parentheses ( ))
- Dash (–)
- Return (\n)

This dictionary will be used to token the symbols and add the delimiter (space) around it. This separates the symbols as it's own word, making it easier for the neural network to predict on the next word.

Implementation

As eariler we have aleadry discussed in "algorithm section" that we will use RNN to creat our Vidyabagish model, here we will only discuss how we will train the model.

Training our Neural Network

To train our model, we first need to create a word embedding of our input. The word IDs will be embedded into a dense representation before feeding to the LSTM. This allows the model to efficiently represent the knowledge about particular words.

The embedding matrix will be initialized randomly and the model will learn to differentiate the meaning of words just by looking at the data.

```
embedding = tf.Variable(tf.random_uniform((vocab_size, embed_dim)), name="embedding")
    embed = tf.nn.embedding_lookup(embedding, input_data)
```

Hyperparameters

The following parameters are used to train the Neural Network:

- batch_size: The number of training examples in one pass.
- num_epochs: One pass of all the training examples.
- rnn_layer_size: Number of RNN layers
- rnn_size: Size of the RNNs.
- embed_dim: Size of the embedding.
- seq_length: Number of words included in every sequence, e.g. sequence of five
- words.
- learning_rate: How fast/slow the Neural Network will train.
- dropout: Simple way to prevents an RNN from overfitting - link.
- show_every_n_batches: Number of batches the neural network should print
- progress.
- save_every_n_epochs: Number of epochs the neural network should save progress.

Parameters used in our best performing model:

```
# Batch Size
batch_size = 512
# Number of Epochs
num_epochs = 700
# RNN Layers
rnn_layer_size = 2
# RNN Size
rnn_size = 256
# Embedding Dimension Size
# Using 300 as it is commonly used in Google's news word vectors and the GloVe vectors
embed_dim = 300
# Sequence Length
seq_length = 10
# Learning Rate
learning_rate = 0.001
```

Also please note if you are using Tensorflow 1.0.0 then bulid your LSTM in Vidyabagish_nn.py in the following way.

```
    lstm = tf.contrib.rnn.BasicLSTMCell(rnn_size, state_is_tuple=True)

    # Add dropout to the cell outputs to prevent overfitting
    drop = tf.contrib.rnn.DropoutWrapper(lstm, output_keep_prob=keep_prob)

    # Stack up multiple LSTM layers, for deep learning
    cell = tf.contrib.rnn.MultiRNNCell([lstm] * rnn_layers)
```

However is you are using Tensorflow 1.1.0 MultiRNNCell function need to be defined in the following way or else you will get error while running Vidyabagish_nn.py.

[lstm] * rnn_layers does not work in Tensorflow 1.1.0

```
def lstm():
    lst = tf.contrib.rnn.LSTMCell(rnn_size, reuse=tf.get_variable_scope().reuse)
    return tf.contrib.rnn.DropoutWrapper(lst, output_keep_prob=keep_prob)

cell = tf.contrib.rnn.MultiRNNCell([lstm() for _ in range(rnn_layers)], state_is_tuple = True)
```

As I had both the verssion I faced this issue during running it in differnet verssion this issue is registered in Tensforflow Github

Refinement

We can see that a learning rate of 0.01 is too large to train our Neural Network. When wetrained it with 0.01, we were never able to achieve a train loss < 3. Another indicator of this is that the learning plateaus in both runs (0001, 0003); in 0001 it plateaus at around epoch 100 and in 0003 at around epoch 180.

The training loss improved when we use a learning rate of 0.001. The lower learning rate improves our Neural network performance and we are getting closer to a perplexity of 1. Remember that the lower our perplexity, the better our model is at predicting the next work. As the training is not plateauing, we are also able to train it longer. This is why we increase the epochs of run 0002 to 500.

(D)Results

Model Evaluation and Validation

The table below captures the results of training the Vidyabagish Neural Network with different hyperparameters:

Run ID	Batch Size	Epochs	RNN Layers	RNN Size	Embed Dim	Seq Length	LR	Dropout	Train Loss
0001	512	300	2	256	300	5	0.01	0.6	3.438
0002	512	500	2	256	300	5	0.001	0.6	1.488
0003	512	300	2	256	300	10	0.01	0.6	3.015
0004	512	500	2	256	300	10	0.001	0.6	1.112
0005	512	500	3	256	300	10	0.001	0.6	1.178
0006	512	500	3	256	300	20	0.001	0.6	1.317
0007	512	700	2	256	300	10	0.001	0.6	0.998

For a detailed view of the training loss, checkout the training logs included with the project.
Our models 0004 and 0007 are the best performing when trained with our dataset.

Sequence Length
Our basic RNN was trained with a sequence length of 5. The sequence length, is the number of words to be included in every sequence.

We can see an improvement when increasing the sequence length to 10. This means that our RNN will use a longer sequence to train our Neural Network. Which ends up improving significantly the quality of text generated by our network.

With a sequence length of 5, our text didn't made much sense, the sentences are short and the paragraphs are not well structured.

When using a model trainer with sequence length of 10, we can notice that the text makes much more sense, and the quality of the sentences and paragraphs improves significantly.

Justification

- In order to generate better quality of text, we need a large text corpus and in this project we are limited by the amount of text we can use as an input. Saying that, the model is generating text that looks similar to the original text.
- The same algorithm can be used to generate text for different subjects. For example,the same algorithm can be trained to generate text for Barack Obama. There is a lotmore text available for Obama (books, speeches)
- A 3 layer RNN takes a long time to train, it might be possible to achieve better results with it but it will need to be trained longer.

Statsticaly our best result in the the first 6 runs was run 0004. If we train our network longer with the same parameters, we achieve a train loss of less than 1.

Few Question about the final Model :

Q -Is the model robust enough for the problem?

Yes, the 0004 and 0007 model we can say robust, as we can clealry see that they are both able to open and close quotations.The text makes more sense. Paragraphs are well formed.Sentence length is similar and close to the average sentence length.However there are lots of scope of improvement.

Q- Can results found from the model be trusted?

We can trust the result, however to generate a novel exatly like Vidyabagish there is long way to go. Hence I wouldn't recomand this model as production ready.

Q-Is the final model reasonable and aligning with solution expectations?

Yes , our benchmark for final model was that achieves a perplexity of less than 3. Our model is achieving a perplexity of less than 1. Which means, it has a high likelihood of choosing the correct word.

The paragraph structure generated by our model is similar to the structure from our dataset.The sentences in the paragraph have betwen 10 and 15 words.The paragraphs are formed of 5 or more lines.

(D)Conclusion

Free-Form Visualization

Based on the results, we can see that RNNs are very effective when understanding sequence of text and can be used to generate text.
Our best performing RNN consists of 2 RNN layers of 256 in size. We added some dropout to prevent overfitting and used Adam to optimise our model.
Generating text is a hard problem to solve in Machine Learning. It takes a long time to train a simple model and the larger the input the longer it will take to train the model properly. In our training, the models took close to an hour to train.
Even though our train loss for the last run is less than 1, we can see in the samples below that the text generated by run 0004 and run 0007 models are similar in several ways, as you

would expect when both models have a perplexity that is close to 1:

- They are both able to open and close quotations
- The text makes more sense when compared with run 0001, which is expected as the
- sequence length (10) used to train both models is longer than run 0001 (5)
- Paragraphs are well formed.
- Sentence length is similar and close to the average sentence length

Here are some visualization example of the diffennt model generated texts.

Run 0001 Model

Quixote," and my profession abideth about her Majesty then, the cloth was frantic, that it is so hast been, that she has not spread of showing and tear resting with him, for the persuasion, leaving him by force or get up by full great achievements of what book, Leaves the pastime?"
" I would lay myself upon him with a burnished hand to the waist. The thieves laughed at the other little intelligence, and where the redress, in the service of some dwarf, and we were in the line of mutual Sancho, this, I have not such matters of us with I not heard by which is the best world, Non when the bano seated myself and Sancho Panza asked the same of his story, they were excellent wife at once, and here the green intentions of the Judge so cautiously gave her. The blush, and heard them came to over a payment with shepherd, laying round it he ordered her, and

Run 0004 Model

Quixote," and am the idea thou wilt give more good quickly to see now thou hast won the good thing, as I have told them not."
" Senor," said Sancho," I mean to know from this perilous journey in the ugly which has been bound; for it is that? What are it in me; but the knight-errant should come free?"

” At any rate, Dulcinea,” replied the actor held in nonsense of your wife, master the devil who is so generous that I can go to the house of Luscinda, Preciosa, and more will by everything the wrong the beauty itself ought to wash its course.
” What could mean be” The greatest who is long in his own behalf so long as I have chosen to Uchali, who shall not go to defend your ass will run wrong, for the mercy I was now in a coach that, after having sold their riches and language that Preciosa asked

Run 0007 Model

Quixote or cost him his squire, unless indeed his wife might follow him or with great respect. Their master would be, the blind parents describes, or look free, and all the rest with your life shall be imagined; the gipsy had rule come to his heart. The extreme day I have heard of her good; in the moment of this my house and soul to find the paper; though, as it were, we believe it is because they who are dead, and bind me to undergo the exertion thou great desire, for indeed he said to him cannot read it aloud.
Don Quixote bade Sancho he settled three days with open his heart in fixing his affections should comply with Preciosa, and to keep secret whatever will be no wish on, as it was queens as his master had heard him, he strove to sing him, maintaining the sun favouring befall), which was not of it, both of them, served in her courtesy for the

From the above example it is very clear the the model 0004 and 0007 is better then the 0001

Reflection

NLP is an interesting field of machine learning that can be applied to many exciting problems. The problem we have applied it to generation of new text from the old one.

While Deep Learning is capable of solving many problems, the part that I find most challenging is the specialized compute resource needed to get good results. GPUs are becoming cheaper and I believe that every device would soon be able to train deep learning models, or at least retrain their models based on new information. Also as Sundar Pichai mentioned in his last I/O speach that Google is going to give TPU for free to resercher no doubt that this filed is going to be very exciting in the near future. https://www.youtube.com/watch?v=vWLcyFtni6U

In this project we acttually trained our model on the text that we obtained from Ramchandra Vidyabagish’s work. The text was went through various preprocessing then we feed it to a RNN. The trained models from RNN was able to meaningfull sentences which are completly new.We observe that the sentece length,Learning rate of the optimizer and number of RNN layers greatly effect the model.

Improvement

There are two ways which we could improve the results of our model:

- Larger text corpus. We can search for more Vidyabagish novels available in English or we can train our model to learn Bengali, as it would be more probable to find the novels in Bengali than English.
- Train our 3 layer, 20 sequence length RNN further. As our model is bigger, we could train it longer and potentially could have better results.

References

- NLP Tokenization -https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html
- Vector Representations of Words - https://www.tensorflow.org/tutorials/word2vec#motivation_why_learn_word_embeddings
- Recurrent Neural Networks - https://www.tensorflow.org/tutorials/recurrent
- Alex Graves - Generating Sequences With Recurrent Neural Networks https://arxiv.org/pdf/1308.0850.pdf
- Christopher Olah - Understanding LSTM Networks http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- Prasad Kawthekar, Raunaq Rewari, Suvrat Bhooshan - Evaluating GenerativeModels for Text Generation - https://web.stanford.edu/class/cs224n/reports/2737434.pdf
- Ilya Sutskever, James Martens, Geoffrey Hinton - Generating Text with Recurrent Neural Networks - http://www.cs.utoronto.ca/~ilya/pubs/2011/LANG-RNN.pdf