Vidyabagish

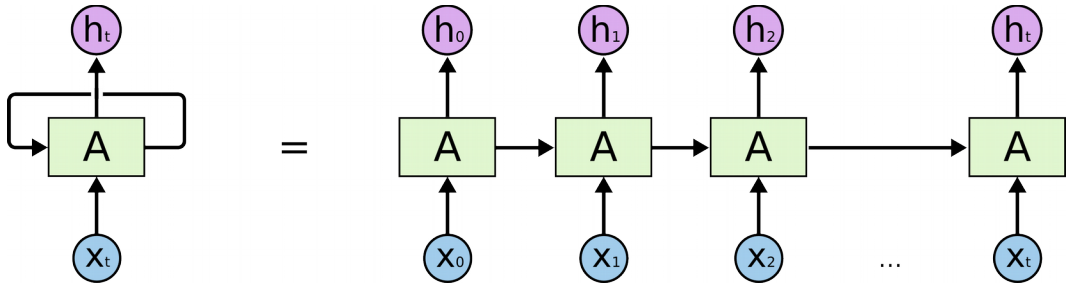
MLND - Capstone Project Proposal

Kabya Basu

Domain Background

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.

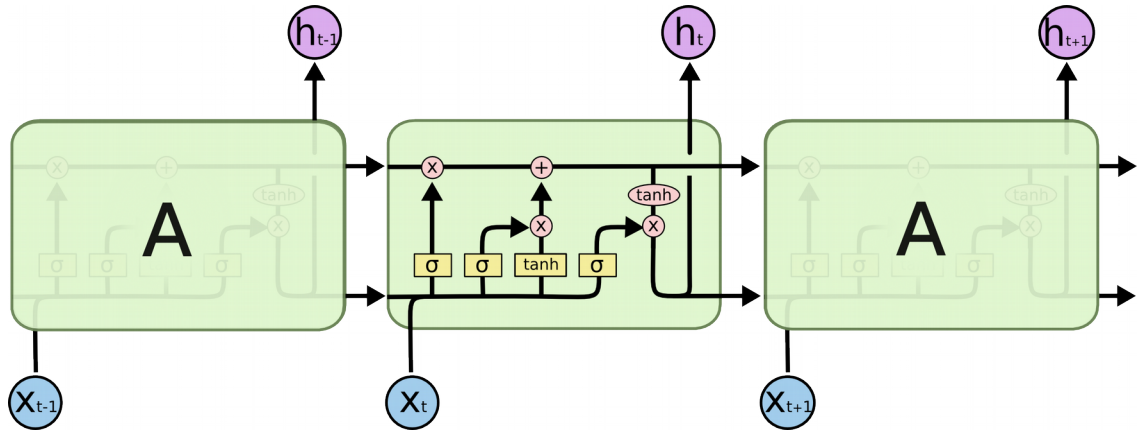
Problem Statement

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

will be trained on existing Vidyabagish works and will output new text, based on the internal representation of the text it was trained on, in the Neural Network.



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.

Datasets and Inputs

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.

Solution Statement

RNNs arevery effective when understanding sequence of elements and have been used in the past to generate text. I will use a Recurrent Neural Network to generate text inspired on the works of Vidyabagish.I will test generating text with different RNN architectures and tune / train it to generate readable text.
One thing to take in consideration is that to generate good quality text, a large corpus of text is needed. There is a limitation on the amount of Vidyabagish text available but it will be enough to generate text that is readable.

Benchmark Model

We will use 2 models to generate text:

1. Our entry model will be a basic RNN with no tuning. I will generate text with it and use its output to compare results.
2. Tuned RNN, different hyperparameters (batch size, RNN size, epocs, batch size, dimension, sequence length, learning rate) will be used to find the optimal RNN.

Evaluation Metrics

We will evaluate our model by reporting the loss and comparing examples of text generated by our RNN.

1.Loss: We will calculate the weighted cross-entropy loss for a sequence of logits for training. The goal is to achienve a training loss less than 1.0

2.Examples: Different training checkpoints will be saved and we will generate samples against them to see how the model evolves over time.The generate text will be qualitatively evaluated to see if from a human’s perspective the generate text makes sense.

Project Design

Steps required to complete the project:

1. Vidyabagish text available by David Hare in pdf format.Where we used “pdftotext” linux command to tranform it into a text file.

2. Data Preprocessing: As discussed in the Datasets and Input section, we will first remove all unwanted text and then proceed to create the word embeddings / tokenise the contents

3. Once our data is ready, we will proceed to train our RNNs as discussed in the Benchmark Model section.

a. Train basic RNN and save the model for future reference. This will help us generating text in the future.

b. Tune RNN and test different parameters to optimise our text generation. As withour basic RNN, we will save different models and they will be used to generate text and compare the results.

References

1. NLP Tokenization -https://nlp.stanford.edu/IR-book/html/htmledition/tokenization-1.html

2 .Vector Representations of Words -
https://www.tensorflow.org/tutorials/word2vec#motivation_why_learn_word_embeddings

3.Recurrent Neural Networks -https://www.tensorflow.org/tutorials/recurrent

4.Alex Graves - Generating Sequences With Recurrent Neural Networks
https://arxiv.org/pdf/1308.0850.pdf

5.Christopher Olah - Understanding LSTM Networks
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

6.Prasad Kawthekar, Raunaq Rewari, Suvrat Bhooshan - Evaluating Generative Models for Text Generation - https://web.stanford.edu/class/cs224n/reports/2737434.pdf