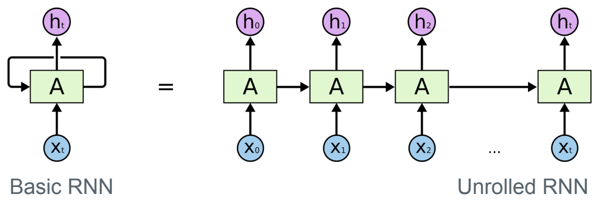
Cervantes 2.0

MLND - Capstone Project

Domain Background

Deep Learning is a new area of Machine Learning, which has attracted a lot of attention lately due to the amazing results produced by Deep Learning models. 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 to transcribe speech to text, perform machine translation, generate handwritten text, image captioning, generate new blog posts or news headlines.



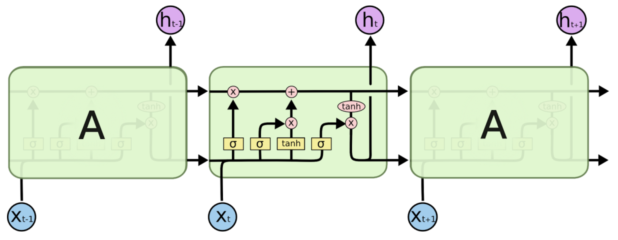
In order to generate text, we will look at a class of Neural Network where connections between units form a directed cycle, called Recurrent Neural Network (RNNs). RNNs use an internal memory to process sequences of elements and is able to learn from the syntactic structure of text. Our model will be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

Problem Statement

Miguel de Cervantes Saavedra, was a Spanish writer who is regarded as the greater writer in Spanish language. Famous for his novel, Don Quixote, considered one of the best fiction novels ever written.

Unfortunately, Cervantes passed away 500 years ago and he will not be publishing new novels any time soon…. But, wouldn't it be great if we could generate some text inspired on Don Quixote and other novels he published?

To solve our problem, we can use text from novels written by Cervantes in combination with the incredible power of Deep Learning, in particular RNNs, to generate text. Our deep learning model will be trained on existing Cervantes 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 most famous novel (Don Quixote) and other less known like Lady Cornelia, The Deceitful Marriage, The Little Gipsy Girl, etc. Also, we will not include any Plays, e.g. Numancia, to train our model as it's writing style differs from the novels and we want the generated text to follow the structure of a novel. All the novels are no longer protected under copyright and thanks to the Gutenberg Project, we are able to access all the text of these books.

Even though Miguel de Cervantes native language was Spanish, 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 - Don Quixote and Exemplary Novels 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:

File		**Totals**			
Name	**Size**	**Pages**	**Lines**	**Words**	**Unique Words**
DonQuixote.txt	2.3 MB	690	40,008	429,256	42154
ExemplaryNovels.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 dataset contains additional text that is not necessary to train our model, for example:

- Preface
- Translator's Preface
- About the author
- Index
- Dedications

After the manual preprocessing, the following preprocessing is needed to have our data ready for our RNN:

- 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 *mad* and *mad!*

Solution Statement

RNNs are very 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 Cervantes.

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 Cervantes 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.

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

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

Steps required to complete the project:

1. Download Cervantes text available online: The content of the novels is available for free at Project Gutenberg
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 with our basic RNN, we will save different models and they will be used to generate text and compare the results.
4. Once the project is complete, the results will be published in my blog (http://ajmaradiaga.com) and code will be available on Github

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 Generative Models for Text Generation - https://web.stanford.edu/class/cs224n/reports/2737434.pdf