Machine Learning Engineer Nanodegree

Capstone Proposal

Vandana Iyer

April 3rd, 2018

Proposal

Automated travel blogging with deep learning

Domain Background

The primary area of focus is Natural Language Processing [NLP] and Deep learning using Recurrent Neural Networks[RNNs]. During the course of nanodegree, I have worked on assignments related to supervised, unsupervised, reinforcement learning and Convolutional Neural Networks. I want to take this learning further by developing a deep learning model that uses RNNs and NLP to achieve automated travel blogging.

This project is based on the paper Character-level Recurrent Neural Networks in Practice: Comparing Training and Sampling Schemes

Problem Statement

Many travellers find very little time to blog on-the-go. For some travellers, blogging is one of the means to earn money to fund their future travel. Is it possible to automate travel blogging with artificial intelligence? Wouldn't it be great if your AI assistant helped you keep memories of your travels by automatically writing down stories about the places you visited? It turns out that this task could be tackled using deep learning because there is plenty of data available online.

In machine learning terms, this problem belongs to the sequence prediction category and is very different from regression or classification. It requires that we take the order of observations into account and use recurrent neural networks (RNNs) which have memory and can learn any temporal dependence between observations. A category of RNN called Long Short-Term Memory (LSTM) helps us achieve this.

The input to the LSTM model will be a text file (around 7MB) containing the blog content of around 1000 blogs from 2 top travel bloggers. It comes to about 6.5 million characters. The output will have 4 different paragraph suggestions of around 400 characters each, depending on the sentence given to the RNN. For example, if I type: "I am travelling to", the possible outputs could be:

```
"I am travelling to "


I am travelling to Europe (finding the companions while I was there) on me. You buy
reading a night's
I am travelling to Cornwall: put I follow you to do get the Olymbookality Starbucks
pretty packed alo
I am travelling to move on the god dorms time you wan me get this mysterio. Everyone
over sizes that
```

Or I could write something like "The hostel was " and the output would be:

```
"The hostel was "


The hostel was cool. So day in the world, move cards, and come back to why time while
I look at the m
The hostel was on the Anded tourist attractions through Europe. 3. The street to the
Musen Gudding is
The hostel was probably not even darked to Travel the Bath. I was celebrated so
often, a high day
```

We can see that results are not perfect but not too bad either. These sentences might not be in the original data set and thus the network will generate new ones on its own.

Datasets and Inputs

Most of the travel blogs that I plan to crawl have been written with Wordpress and they share almost the same HTML structure. I have created my own python crawler, which crawls top 2 travel blogs and does a basic level of filtering by removing javascript and style content that usually gets mixed with the text.

The below table gives an idea of the number of blogs crawled and the amount content extracted.

Sl no.	Travel blogs	Total blogs crawled	Size
1	Wandering Earl	40 pages, 10 blogs per page = 400	2.3 MB
2	Nomadic Matt	100 pages, 6 blogs per page = 600	4.5 MB

A total of around **1000 blogs** have been crawled amounting to about **6.5 million characters (7 MB of text).**

Solution Statement

It is possible to train a character-level Recurrent Neural Network (RNN) with the data collected. I will be using a keras-based implementation of Character-Aware Neural Language Models to train and predict the next character after a sequence of characters. (Reference - How to Develop a Character-Based Neural Language Model in Keras). A LSTM will be used to achieve this. I shall at a later stage use Word-Level Neural Language Model if the Character-based RNN does not give meaningful sentences. I will also try out both stateful and stateless LSTM and use the one that gives good results.

Benchmark Model

I have come up with a vanilla stateless LSTM that uses the same dataset which I will be using for my final model. Sharing the link to the ipython notebook below:
https://github.com/sapvan/udacity-mlnd-capstone-project/blob/master/blog_bot.ipynb

This has a loss of **1.5983** and perplexity score of **15683.8785.** This will be my benchmark model.

Evaluation Metrics

I looked into BLEU, METEOR and ROGUE. In my honest opinion, I think nothing can beat human evaluation for a generative RNN which gives out random characters at each point. I even posted in stackexchange and found that it is still an active area of research and human evaluation is the best metric so far. For the purpose of the project, since a metric other than human evaluation is mandatory, I found out perplexity is a better measure as compared to BLEU, METEOR or ROGUE. So have used the perplexity as an evaluation metric.

Project Design

First step will be data collection. I have collected about 7 MB data crawling around 1000 blogs of top 2 bloggers. It amounts to about 6.5 million characters. I have come up with my own crawler to achieve this. I have designed a vanilla LSTM which uses this data and comes up with 4 different suggestions of a paragraph containing 400 characters each. The ipython notebook for the same can be found here:

https://github.com/sapvan/udacity-mlnd-capstone-project/blob/master/blog_bot.ipynb

To begin with, the data from the text file is read and analysed. It contains a lot of unwanted characters. All non-ASCII characters are filtered out first. There are around 12 more unique ASCII characters which are not required. They are filtered out as well

The total number of unique characters turned out to be around 58. The characters cannot be modelled directly. It should first be converted to integers. This is achieved by mapping every character to an integer. A reverse mapping is also done, so that the integers can be converted back to characters.

The integers should be rescaled to a range of 0-to-1 to make the patterns easier to learn by the LSTM network that uses the sigmoid activation function by default. In order to do that, the input sequences are one-hot encoded by creating a 3-dimensional matrix representation of sentences as described below:

- The 1st dimension is the total of all sentences (nb sequences).
- The 2nd dimension is the length of each sentence, in our case 40
- The 3rd dimension is the length of total vocab/unique characters, in our case 59.

The output consists of a one-hot encoded 2-dimensional matrix.

- The 1st dimension remains the same as input.
- The 2nd dimension has the length of total vocab/unique characters.

The output pair tells the next char for every input pair.
Feeding this data to a vanilla LSTM model gives the following scores:

Loss: 1.5983
Perplexity: 15683.7448

Improvements:

- Crawl all travel blogs mentioned here
- Train the model on padded sentences rather than random sequences of characters.
- Increase the number of training epochs to 100 or many hundreds.
- Add dropout to the visible input layer and consider tuning the dropout percentage.
- Tune the batch size, try a batch size of 1 as a (very slow) baseline and larger sizes from there.
- Add more memory units to the layers and/or more layers.
- Experiment with scale factors (temperature) when interpreting the prediction probabilities.
- Change the LSTM layers to be "stateful" to maintain state across batches.
- Turn it into a webapp using Flask or web.py
- Launch it on AWS as an app that returns paragraphs of travel blogs based on the input text given to the API.

References

→ Character-level Recurrent Neural Networks in Practice: Comparing Training and Sampling Schemes
→ LSTM Text generation
→ Sequence prediction using LSTM
→ Travel Blogs with Deep Learning
→ Crawling Blogs
→ Character level deep learning
→ The Unreasonable Effectiveness of Recurrent Neural Networks
→ How to Develop a Character-Based Neural Language Model in Keras
→ Develop a Word-Level Neural Language Model and Use it to Generate Text
→ Character-Aware Neural Language Models. A Keras-based implementation
→ Metrics NLG Evaluation
→ Using different batch sizes
→ Stateful LSTM
→ Sequential model guide