CS551H Natural Language Generation

Practical 6 – Character-level Neural NLG

In this practical, you will learn to build a character-level neural NLG model using the Keras deep learning library. The code samples in this practical have been tested on Google Colaboratory (Colab) – a cloud based Jupyter notebook environment that comes with a Keras installation. You are free to use your personal computers for this practical if you have Keras installed on them.

A neural language model predicts the probability of the next token (character/word) in a language sequence based on the tokens already seen in the sequence. A neural NLG model is nothing more than a neural language model run iteratively generating (predicting) text token by token. In this practical we consider character-level tokens to train character-level neural language models and use them to generate text.

Part 1. Getting Started with Colab and Keras

1. If you are new to Colab, signup for the service and go through the short video on the welcome page.
2. From MyAberdeen download the hello_world_of_keras.ipynb file and upload to Colab (File\Upload notebook). Although this program does not relate to NLG, it is a simple program that works on a popular dataset (MNIST https://en.wikipedia.org/wiki/MNIST_database) and therefore is good for learning the five step process to model development in Keras discussed in lectures. MNIST dataset has been one of the first major success stories for deep learning. A neural model is defined in terms of its layers and the units inside each of the layers. Understand how to architect neural models. Each layer is like a Lego brick and you join compatible bricks to build a coherent model. Try changing the number of layers in the model and the number of units inside these layers. Experiment with different activation functions as well. Use Keras documentation for valid activation functions. You may like to keep one of the several Keras cheat sheets, available online, to hand.

Part 2. The Five Step Process applied to Language Modelling

3. From MyAberdeen download char_lang_model.ipynb file and upload to Colab. This notebook has a program to develop a very simple character-level LSTM model. This program currently trains on a short rhyme and regenerates it. Run the program (run all the cells) to generate text.
4. The notebook has code cells that correspond to the five-step process. Spend some time understanding the code corresponding to each of the five steps. Your attention should be on the multiple code cells that correspond to the first step of defining your training data. As you would expect data preparation involves several steps in our case because language sequences cannot be directly fed to neural nets (NNs). There are mainly three steps to create the required input and output tensors: tokenization (in this case into characters) of cleaned text, encoding the tokens into integers and finally vectorization. Learn this three-step process well because this plumbing work is required in all your future neural NLG programs.
5. The next point of interest relates to learning the shapes of input and output tensors as these are standard for LSTM-based neural NLG. Add code to print the shapes of these tensors and understand their shapes.

6. Experiment with tuning the model – change the number of layers and the number of units in these layers, running the model each time to see the impact of your changes on the output.
7. From MyAberdeen download warpeace_input_short.txt file using the code from the second code cell in the notebook. Make changes to the Main code cell (last cell) to read the uploaded file. The text from this file generates a reasonable number of character sequences for training a good model. Modify the main code cell to change the seed_text passed to the generate_text function – you could simply peek into the character sequences and manually type them into the code.
8. Try changing the length of the sequence used for training the model – remember you will need to change the model to match with the sequence length and will need to change the length of the seed_text as well. Observe the impact of longer and shorter sequences on the quality of output.

Part 3. Using temperature-controlled sampling

9. From MyAberdeen download char_lm_sampled.ipynb. This is a more concise program for developing a character-level neural NLG model. This model trains and generates incrementally and therefore you can observe what the system is learning as the training progresses.
10. Compare the two models you have and make a note of all the differences you spot in model definition, configuration and prediction.
11. The new model uses a temperature-controlled sampling process for brining in an element of creativity into the generated text. Try changing the temperature values to find what works best in terms of balancing randomness (creativity) and learnt pattern.

Part 4: Create your own program for character-level neural NLG

12. Create your own program using the two programs you worked with in this practical to build a character-level neural NLG model to learn the five-step process better. Experiment with it to understand how these models work and more importantly how they perform. What do you see as the main merits and demerits of this technology as an alternative to rule based or statistical NLG?

References

http://karpathy.github.io/2015/05/21/rnn-effectiveness/