CSE 6240 Project #2
LSTMs for Language Detection
Due: 4/24/2017

LSTMs are useful for building character-sequential prediction models. In this project you will use it to build string scoring models which you will then combine into a single language detector. To begin, please refresh you background on LSTMs:

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Part 1 -- Building String Scoring Models [100 points]

- Download this dataset, https://github.com/GT-CSE6240/proj2/tree/master/data , which is used in this blog, http://cloudmark.github.io/Language-Detection-Implemenation/ , for a language detection experiment.
- Review the Keras documentation and example on LSTM for text generation at https://keras.io/getting-started/sequential-model-guide/ and https://github.com/fchollet/keras/blob/master/examples/lstm_text_generation.py . The example, in particular shows how to train an LSTM model. Note that we will not be using the sample generation part of the code (feel free to look at it, though).
- For eng.txt and frn.txt, split each file into 80/20 learning/holdout subsets. Lowercase all of the letters for simplicity.
- Train two LSTM models (size 128), one for eng.txt and one for frn.txt, on the learning datasets you created. Set nepochs=5. You can experiment with other settings if you like. An example line from your script might be: model.fit(X_train_eng, y_train_eng, batch_size=128, nb_epoch=5)
- Evaluate your model and plot an ROC
    - Generate a test dataset from the holdout files. From each of eng_holdout.txt and frn_holdout.txt files, randomly select 100 5-char substrings. You will end up with 200 test strings. These are the x_test for evaluation. The y_test are 1 for english and 0 for French.
    - For each test string, compute the log likelihood of that string for each model. This is the hard part of the assignment and may require a little bit of thought, but you use model.predict() and conceptually generate the predicted probability for each string. For example, if my test_string is “trump” you would use model.predict() to compute Pr(t|START), Pr(r|START,t), Pr(u|START,tr), … Pr(p|STARTtrum), take the log of the probabilities and add them up. So for each test string you will end up with two numbers (log(Pr(string|eng)), log(Pr(string|frn))).

    - Use standard sklearn to compute an ROC where y_hat is the ratio of the two loglikelihood ratios. For example, if “trump” -> (0.8, 0.4), your y_hat would be 0.8/0.4 = 2
    - Plot the ROC on semilogx and print the AUC-ROC on the plot.
- Questions
    - Is this model good?
    - What are at least three alternatives to language detection that you can think of or find on the internet? What are the pros and cons of each approach?
    - Briefly describe at least 5 ways that you can improve this model, and what you think the value and predicted result of each approach would be? Example: “could use GPUs for training -> faster to get datasets; no change in efficacy”

Part 2 -- Extra Credit [up to 40 points]

For extra credit you should explore improvements to the language detection experiment. You are free to choose, and the amount of extra credit will be subjectively assigned based on how complete, interesting and effective the approach is at improving the experiment. Note that you don’t have to necessarily improve your detector results; some baseline experiments that might be expected to perform poorly may still be valuable for educational purposes.

Some examples:

- Train models for all languages (download data from http://cloudmark.github.io/data/subset.zip). How do you score, then?
- Explore early stopping by monitoring the validation dataset fit.
- Explore variations in the hyperparameters of the model, including the size and number of LSTM layers.
- Compare to simpler methods like n-grams or hmms or others.

Deliverables

Your blog posts + supporting files and data + your code as python | ipython notebooks.

This project may be computationally expensive so don’t procrastinate!