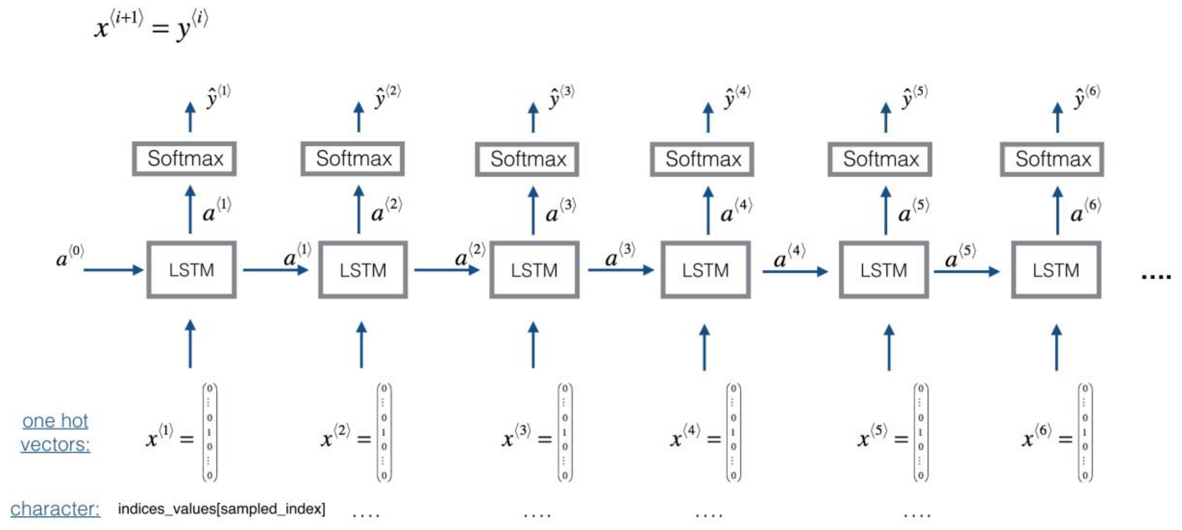
CSC 4700 Homework 2 (due Apr. 4th by end of day)

Your task in this homework is to build a character level LSTM network to generate (English) names.

Download training data from moodle. (You may further reduce the number of names by randomly sampling 5000 names to use as training data.) Also remove names with special characters and convert the rest to lower cases such that the training data contain only names with characters a-z.

Let the length of the longest name string in your training data be N. You should make all the strings have the same length N by padding the shorter ones with 'space' character.

Implement a LSTM network with the following structure. (The LSTM layer has 128 cells).

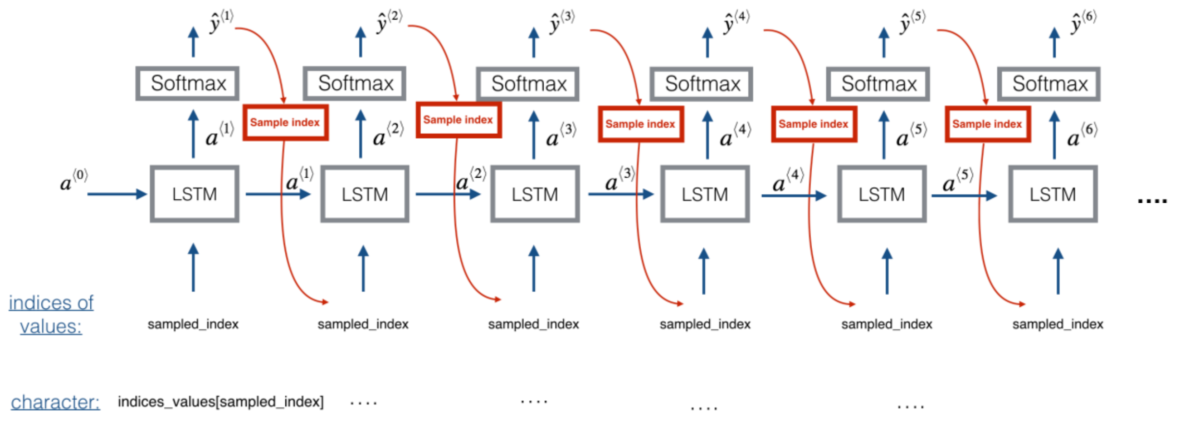

$x^{\langle i+1\rangle} = y^{\langle i\rangle}$
$\hat{y}^{\langle 1\rangle}$ $\hat{y}^{\langle 2\rangle}$ $\hat{y}^{\langle 3\rangle}$ $\hat{y}^{\langle 4\rangle}$ $\hat{y}^{\langle 5\rangle}$ $\hat{y}^{\langle 6\rangle}$
Softmax
$a^{\langle 0\rangle}$ $a^{\langle 1\rangle}$ $a^{\langle 2\rangle}$ $a^{\langle 3\rangle}$ $a^{\langle 4\rangle}$ $a^{\langle 5\rangle}$ $a^{\langle 6\rangle}$
LSTM
....
one hot vectors:
$x^{\langle 1\rangle}$ = $x^{\langle 2\rangle}$ = $x^{\langle 3\rangle}$ = $x^{\langle 4\rangle}$ = $x^{\langle 5\rangle}$ = $x^{\langle 6\rangle}$ =
character: indices_values[sampled_index] ....

Note that the same name (string of characters) is used both as input and output in training but the input is the string right-shifted by one position. After shifting, add an artificial character (marker for the beginning of the name) to the beginning (the left most) position. For the output string, you should add a different artificial character to the end, which serves as a marker for the end of the string. If you have padded 'space' to the string, this end-of-name character should be added to the actual end position of the name, not the position at the end of the 'space' characters. The input and the output strings are thus of the same length.

You should translate the characters in the strings into one-hot vectors and use the one-hot vectors as the inputs to the LSTM. Because of the two artificial characters, you have 28 characters in total and the length of the one-hot vectors should be 28. (As an example, for the input, $x^{<1>}$ in the above figure should be the one-hot vector corresponding to the first artificial character.) You can use an all zero vector for the padding 'space' character.

Train the model with cross-entropy loss from the softmax outputs at all (**valid**) steps. That is, if you padded a string, the cross-entropy from the padding positions should not be included in the final loss. To achieve this, keras has a masking layer. If you add the masking layer at the beginning of sequential, the masked positions won't have any effect in the downstream layers (including the loss).

After training the model, you can sample (generate names) using the model. Implement the following process to generate a name using the trained model. You should implement a function "generate_name(N)". The function uses your trained model and each call to the function should generate a name (string).



(You may ignore y in the above diagram.) The next character should be sampled following the probability (the output of the softmax layer), not using the one with the largest probability. (You may use numpy.random.multinomial to draw samples from a multinomial distribution.) Due to the sampling, consecutive calls to the function should generate different names. If your sample is the beginning-of-name character, ignore it and resample. If your sample is the end-of-name character, stop the process and return the name. Even if you don't see the end-of-name character in the process, once N characters have been generated, you should stop the process and return the name.

Homework Submission:

Upload a Python file (not .ipynb file) named **RNNModel.py** in moodle. You need have the following in the file:

1. A function **create_model(max_len)** which returns a keras model (untrained) that implements the above LSTM model. "max_len" is the max length of the input string.
2. A function **generate_name(N)** which returns a string generated by your (trained) model. In this function, your code should not train the model from scratch. Rather, the code should load a trained model from a model file and generate a name string using the model. (See example code on HW1 pdf for save and load trained model.) Same as HW1, You should not upload the .h5 file of the trained model to moodle. Instead, you should share it in your google drive and put the share link in a comment line at the beginning of the RNNModel.py. (*Before submission, make sure people other than yourself can download the model file using the link. Your code should work when the .py file and the model file are in the same directory.*)

We will test your model by running some code similar to the following:

```
from RNNModel import create_model, generate_name

um = create_model(mlen)
um.summary()    # print out summary of the model
um.fit(right_shifted_train_data, training_data, epochs=5, batch_size=128)
```

```
# note train data here are 3d arrays with shape [batch, mlen, 28].

# print 5 generated names (We expect there are different names among the 5.)
for i in range(5):
      print(generate_name(mlen))
```