Nathan Smith, David Devore

C463 Project Progress Report

2-24-18

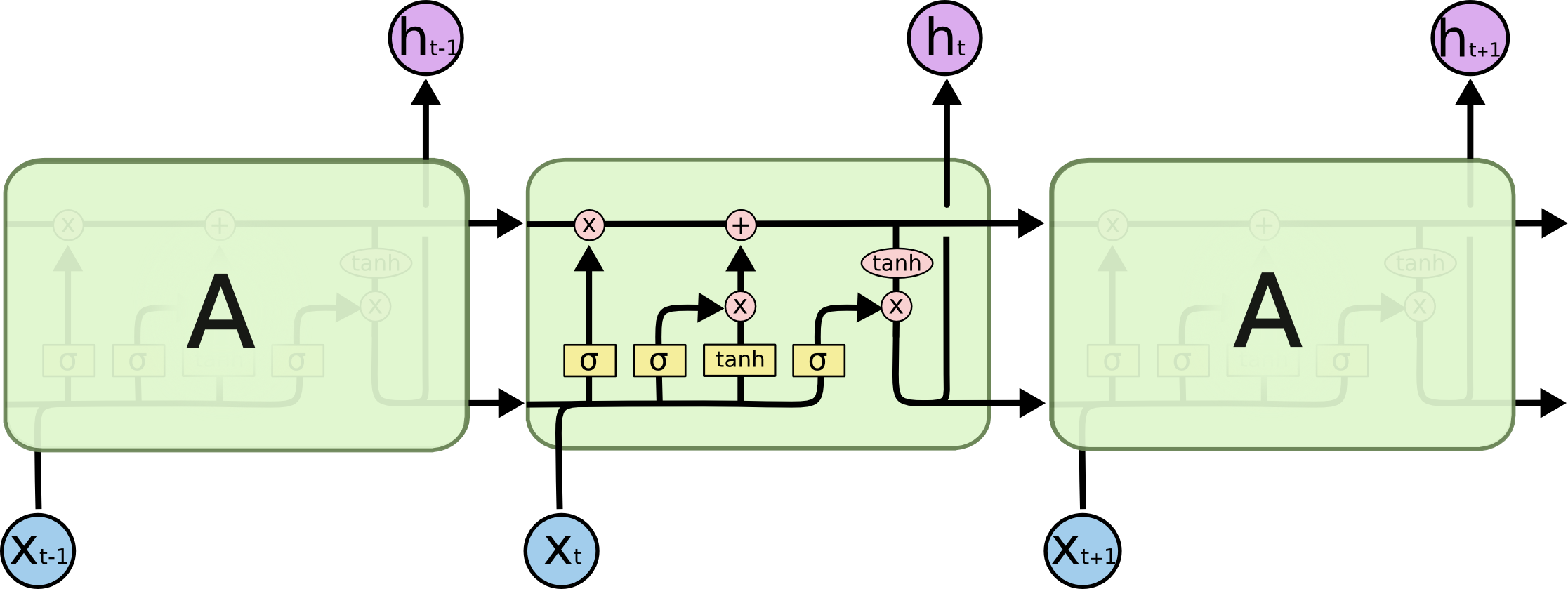
**Model**

We will be using a Recurrent Neural Network. We believe this will be a good choice because RNNs work well with sequences of inputs. They can look back to use previous information to interpret the current situation like human thinking (Colah). This applies to our problem because the task of choosing what word to say/write next, depends upon what you’ve already said.

RNNs are particularly good at language processing it turns out (Karpathy). This is thanks to the augmentation provided by LSTMs, another important piece of our model. LSTM stands for Long Short Term Memory, and it’s a type of recurrent neural network that allows us to create outputs using information from longer ago in our sequence. Traditional RNNs run into an issue of being able to connect previous information to the current situation as the time gap grows (Colah). LSTM cells fix this problem because their structure adds additional layers to each hidden layer. So, instead of having a hidden layer that performs some manipulation on the data, you have a layer that also does things like deciding what information should be kept and what should be forgotten.

**Algorithm**

We will analyze chunks of words of a given length and run them through a set number of hidden LSTM layers to produce an output. Internally, our cell state will keep track of things like the current subject of our writing and the word prompt given by the user. This will allow us to use the correct pronouns, verb conjugations, and relevant words more accurately.



***Figure 1. LSTM Cell***

Accordingly, our cell state will need to be at least a 3 dimensional vector keeping track of normalized numbers relating to who/what we’re writing about, the current tense of the sentence, and the sort of tone we want to achieve based upon the input.

Our first sigmoid gate will be responsible for choosing to forget previous poem subjects as new subjects arrive. For example, if the poem starts talking about a boy and then shifts to talking about a girl, we need to forget the boy as the current subject so that we don’t call the girl ‘he’. Similarly, this will watch for changes in verb tense so we know ‘when’ we are talking about.

The second step involves a second sigmoid gate and a hyperbolic tangent gate. This part is responsible for actually updating the cell state with the new information we are taking in. The sigmoid function picks the information that we’re letting in while the tanh function provide coefficients for scaling the magnitude with which we want to update the cell state.

Our final step is to produce this node’s output to the next node. This is done with a final sigmoid layer. This layer decides what we would like to pull from our cell state in order to create our output value. The job of this operation is to pull only the relevant information from our cell state. If we use the sentence structure model from the book, we may have decided we want to type a verb next. In this case, the gender of the subject of our writing is probably irrelevant. At the very least, it could be scaled down to be less important.

**Concrete Application**

Similar projects have been completed with these methods. In order to help with our project, we’ve been studying an LSTM that analyzes one of Aesop’s fables and tries to output the rest of the story given a sequence of words. This exact version chunks the words into groups of three when running them through 512 hidden layers (Atienza). These words are assigned an integer that the LSTM can understand by finding the frequency of occurrence ranking them. The output of the LSTM is a vector with the same length as the number of unique words. This vector contains a probability at each index that pairs with a given word’s assigned integer value. The word with the highest probability is chosen as the next word to write.

The next step for us is to continue learning how this code works. I’ve been heavily referencing the tensorflow documentation as well as the articles linked below. Once we understand tensorflow’s setup for the RNN, we can dive in deeper to manipulate the hidden layers more. This will give us a more precise agent that can obtain a higher cohesion between sentences by remembering its current subject, topic, etc.

**Initial Results**

*Baseline*

We need to improve the list of words it works with and change them to more commonly used words, so more of the output can make sense. We also need to make improvements on making the words relate to each other. As of right now they have no way to relate to each other except if by chance the words randomly chosen actually do relate

The decision making skills of this particular agent are very limited. The following are the results of the baseline algorithm. It uses the sentence structure chart from our textbook in order to choose its words(Russell). It searches a dictionary and tries to match a word’s part of speech to what it’s writing now. For example, if it is writing a noun, it searches for a noun. It may optionally search for an adverb when constructing a verb phrase. Another problem with the baseline is it is not able to choose the tenses of words, so this makes the sentences that are generated less coherent.

* air stewardess embroil
* othersome bump someone off
* all cottage
* alforja close reach arsenious aldermanly foolhardily affordably alice blue ourselves
* theirs apotheosize whose adornation
* another bottle something up bushed their alphabetical organization infelicitously
* yah mitout bare albendazole exact herself
* suchlike air fuelling fiddle-faddle aphetically agency bove it behind aforeness like wot adaptive optics yewall acromegalic whatever abaxile sufficient alternative lifestyle
* othersome but akrasia deliquesce
* altar stair excogitate alley crop except yuh ablution
* yonder afterburner bream black and white whatso magniloquently balconied
* treble garrotte branded
* neither acronymania deed abstract expressionist animato alkalescent
* accused formulate acid-suppressing whichever aguardiente

*Oracle*

Our oracle obviously performs much better. However, it is much less interesting. It takes in a user input for a topic to write about. It then searches pre-existing poems for a match in classification. Once it finds one, it writes the poem out to the user. An example run looks like this:

Enter a topic: History

['Black,', 'like the memory-wound,', 'the eyes dig toward you', 'in the by heart-teeth light-', 'bitten crownland,', 'that remains our bed:', 'through this shaft you have to come—', 'you come.', 'In seed-', 'sense', 'the sea stars you out, innermost, forever.', 'The namegiving has an end,', 'over you I cast my lot.']

*Neural Network*

Currently, this agent is set-up to study a single story from Aesop’s fables. It then learns what word is likely to come next given a sequence of words before it. From there, it mimics knowing what should be written in a given context. It can perform well enough to rewrite the entire story solely by predicting what word will come next. But, this all depends on how long the neural network is trained.

After the first one thousand iterations, it’s not very accurate. Here are the numbers at that point:

**Iter= 1000, Average Loss= 4.389004, Average Accuracy= 6.60%**

**['bell', 'the', 'cat'] - [?] vs [.]**

It is telling us what token it predicted vs what it should have predicted. Here, it chose a question mark instead of a period. So, while it may not be very accurate, it’s guessing things that make sense. This is already a good improvement over the baseline.

At 10,000 iterations, the network is doing much better with just above 70 percent accuracy:

**Iter= 10000, Average Loss= 0.974052, Average Accuracy= 72.90%**

**['the', 'mice', 'had'] - [a] vs [a]**

We can also see that it’s made a correct prediction this time. It takes the neural net about 50,000 iterations to get above 90 percent accuracy. At that point, it can pretty much write the book on its own. Here are some examples from the net at 10,000 iterations:

**3 words: the mice had**

**the mice had a proposal the make venture all she approach . which mouse a general the that to this said that but at very well and then spoke . then the old mouse means**

It starts off alright and then loses its way. The biggest problem (considering how this relates to our goal) is that when the agent chooses incorrectly, it is often something close to the right thing that still makes no sense. This could be a huge issue for us since we aren’t hoping to rewrite a given text, but to create a new body of writing altogether. This is where our manipulation of the hidden layer could help out. We’d also change the internal cell state so that we don’t only care about what words came before the words given, but also about the overarching subject.

The plan is to combine what you just saw with the RNN with our grammar rules from the baseline, and some sort of enforcement on rhyme scheme. This allows us to have some control over the decision making and parameterize the possible mistakes of the agent. So, our end goal is to have an agent that learns how to write from other poems, produces new poems by looking at what it’s written about so far, and use decision making heuristics to enhance the writing’s cohesion.

**Sources**

Colah (August 27, 2015). *Understanding LSTM Networks*. Retrieved from <https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

Andrej Karpathy (May 21, 2015). *The Unreasonable Effectiveness of Recurrent Neural Networks*. Retrieved from <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>

Rowel Atienza (March, 17, 2017). *LSTM by Example using Tensorflow*. Retrieved from <https://towardsdatascience.com/lstm-by-example-using-tensorflow-feb0c1968537>

Russell, Stuart J., and Peter Norvig. *Artificial intelligence: A modern approach*. Pearson Education, Inc., 2010.