# CPSC 393 Assignment 5: Markov Chain and LSTM text generation

By Tarek El-Hajjaoui

## Introduction

One of the capabilities of machine learning are models that predict the future state of a system given information of the current and previous states. Markov Chains (MCs) and Long Short-Term Memory (LSTM) models are 2 models that accomplish this.

Markov Chains are methods of predicting future states of a system. For example, MCs can be used to predict what the weather in a particular area will be like. Markov Chains are able to make predictions of a system's state by using the knowledge of the system's present state, "*or several of the most recent states for a higher-order Markov process*" [I]. The transitions between states are conditioned, or dependent, on the previous state(s) before the transition occurs. The probability distribution of state transitions is typically represented in a matrix called the transition matrix. "*If the Markov chain has N possible states, the matrix will be an N x N matrix, such that entry (I, J) is the probability of transitioning from state I to state J*" [II]*; "each row represents its own probability distribution"* [II]. The entries of each row in the transition matrix add up to exactly 1, making it a stochastic matrix. The other component of a Markov Chain is the initial state vector of dimensions N x 1, where each entry is the probability of the system's state starting at its N respective state. To compute the probability of a system's next state (N + 1), the transition matrix is multiplied by the initial state vector.

An LSTM is a specialized type of RNN (Recurrent Neural Network) that was developed to overcome the shortcomings of RNNs learning long-term patterns and is particularly powerful in sequence problems. LSTMs maintain a cell state that holds memory of the previous series, which helps overcome the vanishing gradient problem that occurs in RNNs. The cell state is maintained and updated via gate mechanisms: Forget, Input, and Output gates. A unique feature of LSTM gates is their use of  tanh activation functions (for input and output gates); since tanh has a wider range than functions like sigmoid, more derivations can be taken from the function before it reaches 0. This allows weight values associated from early layers to be to be taken into account when calculating the gradient over time. Because of this, LSTMs tend to be more effective than simple RNNs with the downside of longer training time / consuming more computational resources. Like Markov Chains, LSTMs can be used to predict the future state of a system after it has been trained appropriately.

For this experiment, 2 models are created: 2nd order Markov Chain and an LSTM to generate rhymes based on nursery rhyme texts. The goal for this experiment is to examine the effectiveness and differences of each model in generating rhymes.

## Background

The models are being based/trained on text from a file of nursery rhymes in order to generate their own nursery rhymes. With the line breaks, the text is over 5000 lines long of which each line is a rhyme consisting of about 5 - 10 words.

The 2nd order Markov Chain model was developed from scratch with the help of an article (source provided). Although it does not explicitly use a transition matrix, it employs the MC concepts via Python dictionaries. The LSTM was developed with the TensorFlow and Keras libraries.

## Techniques

### Preprocessing

Extracting the corpus from the provided file of nursery rhymes, nuseryrhymes.txt, involved preprocessing methods such as removing characters i.e. line breaks, empty spaces, and punctuation. If these characters were to be included in the corpus, the models would not perform as well. Each word was then lower-cased so that capitalization would not make the models interpret words differently. Additionally above each set of rhymes is a header for the rhymes, but are not rhymes themselves. Since the goal of these models is to produce rhymes, these headers were removed from the corpus. After the corpus was cleaned for the LSTM Model, the words of the corpus were tokenized with the Keras Tokenizer API. Tokenizing is converting the sequences of words to sequences of integers representing the words. After the corpus of words is tokenized into a list of tokens, the LSTM Model creates the X and y datasets. The corpus list of tokens is iterated to generate sequences of text (X) and their respective proceeding sequences (y) where X is a pair of words (like 2nd order Markov Chain), and the y values are the true values or the word that comes after the pair. In order to calculate error and train the model, the y values must be one-hot encoded because it gives the network a ground truth to aim for from which the error can be calculated from to update the model [III].

### Markov Chain Model

The transitions of words are represented in Python dictionaries as key-value pairs where the keys are the distinct pair of words in the corpus and the value for a given key is a list of words that appear after that key. A pair of words is used as the key because this is a 2nd order Markov Chain model. The chain was built by iterating through the corpus and adding pairs of words to the dictionary if they are not already keys in the dictionary. The values for a given key is a list of words which appear after its respective key. This dictionary architecture is able to represent the probabilities of the transitions in a simplistic manner via repeating words. The dictionary values or lists of transitions contain repeating words, which are randomly chosen. The more often that a word is repeated, the more likely it is to be chosen; these likelihoods could be thought of as the probabilities that certain words will be chosen next given a pair of words.

LSTM Model

The LSTM Model uses the Sequential model from Keras as its base and has 3 deep layers. The 3 layers are:

1. Embedding layer
    a. This layer converts each word token into a fixed length vector of defined size. The resultant vector is a dense one with real values instead of just 0's and 1's. The fixed length of word vectors helps represent words in a better way along with reduced dimensions. In this way, "*the embedding layer works like a lookup table. The words are the keys in this table, while the dense word vectors are the values*"[IV].
2. LSTM layer
    a. The LSTM layer is applied with 50 units, which were arbitrarily chosen.
3. Dense layer
    a. The output layer is a Dense layer composed of the corpus vocabulary size of units; each unit represents the probability that word is the prediction. The softmax activation function was applied because it is used for multi-class classification.
4. Epochs
    a. Two different LSTM models were created with the only difference being the number of epochs. The first one has 500 epochs and the second has 50 epochs.

## Results

### Markov Chain Model

The rhymes generated by the Markov Chain are diverse compared to one another, and although the sentences are not grammatically correct or very sensible, they're still impressive given the simplicity of the Markov Chain model architecture. In some of the sentences, rhymes are detectable such as:

**Line 11:** *again with a rowley powley gammon and spinach heigho says anthony rowley when they heard they began to sing was not.*

In the example above, the sentence has rhymes with the word pairs - (rowley powley) and (anthony rowley). Other sentences however did not have rhymes such as:

**Line 1**: *tonight but the rope began to fight the fryingpan behind the door while the old woman shall i sing with the.*

### LSTM Model

The first LSTM model achieved an accuracy of about 70% after training for 400 epochs, and the second LSTM Model attained only a 21% accuracy after training for 50 epochs. Similar to the Markov Chain, the rhymes generated by both the LSTM models are not very sensible and not grammatically correct, however, the results appear to be better than the Markov Chains in terms of generating rhyming sequences in the lines. For example:

**LSTM Model 1 ~ Line 4**: *jack nag who killed cock robin said my little dears a whisper reaches itching ears the same will come in and*

In the example above, the sentence has word rhymes with word pairs (little dears) and (itching ears). The second LSTM model did not produce as good rhymes which is expected because it achieved a much lower accuracy score from training a lot less iterations.

## Discussion

The Markov Chain model was able to produce a couple of rhymes despite its simple architecture. These results demonstrate how Markov Chain methods, despite being relatively simple, are still powerful. Although the MC results are decent, the LSTM model, specifically the one training for 400 epochs, appears to have performed better. Although the LSTM model performed better than the MC, the difference in their results is not very significant despite the LSTM's complexity. The LSTM model still has room for improvement however, such as training the LSTM strictly on the sequences of rhymes or using pre-training embeddings. It should also be noted that the data provided, rhymes.txt, does not only contain rhymes, there are a lot of instances in which the rhymes of the lines are not apparent or altogether not present. Additionally, these models were trained as 2nd order or 2 step-back models, meaning they predicted a word based on current and previous words only. Both of these models would most likely have performed better in terms of creating rhymes if they were trained with a higher order.

## Conclusion

Markov Chain methods are effective methods in predicting the future state of a system. LSTMs are more complex models that can learn latent patterns of sequences through their unique RNN architecture with cell states, and the use of backpropagation and other neural network practices. Although Markov Chains are effective, LSTMs have far more potential and applications in predicting future outcomes of sequences.

## Sources

- John Wittenauer, Markov Chains from Scratch[I]
- Devin Soni, Introduction to Markov Chains[II]
- Jason Brownlee, How to develop Word-Based Neural Language Models in Python with Keras[III]