COMPARISON BETWEEN MARKOV CHAIN MODEL AND LSTM RECURRENT NEURAL NETWORK

Quan Nguyen, Nam Anh Nguyen

1. Abstract

In this paper, we are using two algorithms to predict a discrete sequence over the English alphabet. We implemented the natural language processing model using Markov Chain Model (MCM) and Long Short-Term Memory Recurrent Neural Network (LSTM RNN) in order to guess the missing letter in a given string of alphabet characters.

In the Markov Chain Model, the process is based on the prior probability of the train data associated with the location of the letter. The model uses the previous and the later letter to predict the missing letter, while the LSTM RNN uses the prior data to generate letters based on a probabilistic distribution. The Long Short-Term Memory Recurrent Neural Network will take an input of the letter and generate a string of characters based on the given input.

2. Introduction and Background of Markov Chain Model and Long Short-Term Memory Recurrent Neural Network

The learning sequences in data and their connection keeps on being an essential assignment and a test in design acknowledgment and AI. Applications including consecutive information might require pre-style new occasions, age of new groupings, or decision-making like arrangement of successions or sub-groupings.

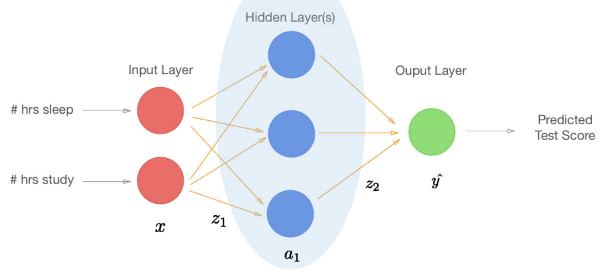
There are reasonably broad and offer different ways to examine and foresee arrangements over limited letters. Maybe the most normally utilized methods depend on the model created by Andrey Andreyevich Markov, the Chain Markov Models (CMMs). Well-given adaptable designs that can display complex wellsprings of successive information. Be that as it may, managing CMMs commonly requires impressive comprehension of and knowledge into the issue space to limit conceivable model designs. Likewise, because of their adaptability, the fruitful preparation of CMMs ordinarily requires huge preparation tests. A simple model and calculation of the Chain Markov Models are shown below.

$$\begin{tikzcd} x_1 \arrow[r] & x_2 \arrow[r] & \cdots \arrow[r] & x_{N-1} \arrow[r] & x_N \end{tikzcd}$$

$$p(x_n \mid \mathbf{A}) = \sum_{x_{n-1}} p(x_{n-1} \mid \mathbf{A}) p(x_n \mid x_{n-1} \mid \mathbf{A})$$

Unlike simpler machine learning models such as Chain Markov Models (CMMs) Recurrent Neural Networks (which was based on David Rumelhart's work in 1986), a deep learning model performs better compared to conventional approaches on practically each and every measurement. Ongoing work in profound AI has prompted more remarkable artificial neural network plans, including Recurrent Neural Networks (RNN) that can cycle input arrangements of self-assertive length. In this paper, we emphasize using an RNN known as a Long-Short-Term-Memory (LSTM), which was created by Seep Hochrieiter and Jurgen Schmihuber in 1977. LSTM networks have improved memory capacity, making the chance of involving them for learning and producing music lyrics, children's books, poems, sample texts, or even language.

From Tesla that can gain proficiency with a driving style to Nvidia Corp, learning a tremendously muddled game, the projects are equipped for figuring out how to take care of issues in a manner our cerebrums can typically do. First perceived in the 1800s, deep learning models are all the more remarkable because they can naturally find portrayals required for location or arrangement has given the crude information they are handled. In a neural network model, we stack various hidden layers in the neural network and perform the calculations below.



$$\begin{aligned} z_1 &= xW_1 + b_1 \\ a_1 &= \tanh(z_1) \\ z_2 &= a_1 W_2 + b_2 \\ \hat{y} &= softmax(z_2) \end{aligned}$$

3. Chain of Markov Model

Our Chain of Markov model uses the training data as a list of words and creates a 26x26 matrix containing the possible alphabetical order in the series. The matrix consists of the possibility that each character will appear after the letter; in this case, the row is the prior letter, and the column is the possibility that a specific letter will appear.

$$P = \begin{array}{c} \\ 1 \\ 2 \\ \vdots \\ k \end{array} \begin{array}{c} \begin{array}{cccc} 1 & 2 & \cdots & k \end{array} \\ \begin{pmatrix} p_1 & p_2 & \cdots & p_k \\ p_1 & p_2 & \cdots & p_k \\ \vdots & \vdots & & \vdots \\ p_1 & p_2 & \cdots & p_k \end{pmatrix} \end{array}$$

We only use the forward algorithm in this case because each letter in the series only depends on their adjacent letter. Hence when looking at a specific location, the probability of a specific location k would be:

$$P(x_k) = P(x_k \mid x_{k-1}) * P(x_{k+1} \mid x_k)$$

And the probability of 2 consecutive letter would be:

$$P(x_k, x_{k+!}) = P(x_k \mid x_{k-1}) * P(x_{k+1} \mid x_k) * P(x_{k+2} \mid x_{k+1})$$

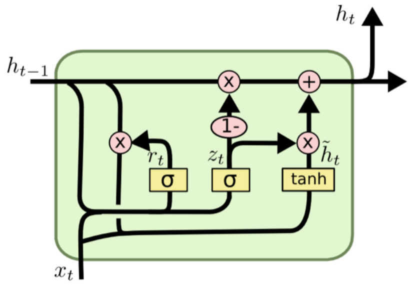
The model will only address the problem of 2 consecutive words as the run time for this approach will exponentially increase as more loops are required. The formula will only apply to letters have both adjacent letters, whereas:

$$P(x_k) = P(x_k \mid x_{k-1})$$

This formula will apply to the start and end of the string.

4. Long Short-Term Memory Recurrent Neural Network

The model is expected to be given training data, where it will first filter the data, which will result in a string of only letters in English with no spaces or punctuations. We will then perform a softmax function to get a guess from the multinomial distribution based on the prior probability for the output and return its value.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

In the above diagram, h’s are the hidden states, *x*t is the current input, W is the weight matrix, and b is the bias and z’s are the output.

We have also used an already implemented model for this Long-Short-Term-Memory recurrent neural network.

5. Discussion

The testing data seems to have a very high miscalculation rate since the prior data given in each scenario is lacking. In the LSTMRNN model, only 1 word let the multinomial distribution to have at most 26 local maximum, hence it is unlikely that the model would perform perfectly.

In the Markov model, we only use the formula above, hence the probability is constant throughout the testing. So the likelihood of each situation is fixed, the probability is not a distribution as the LSTMRNN model but a fixed letter. This problem occurs to both the Chain of Markov and LSTMRNN model. A solution to this problem is to increase the input given data, as in the Chain of Markov model, the input is a single letter, so by increasing the number of letters, the prior probability is more distributed and hence more likely to be correct.

6. Reference

- Oberlin College, Colin Dawson, Hidden Markov Model, 2021
- Recurrent neural network - Wikipediahttps://en.wikipedia.org › wiki › Recurrent_neural_network
- On Prediction Using Variable Order Markov Models - Ron Begleiter, Ran El-Yaniv, Golan Yona, 2004