Bereket Eshete Kebede
UID - U00827234

M.S. Student
University of Memphis, Fall 2022
COMP 7745 Machine Learning

Recurrent Neural Network (RNN) model to predict Google Stock Prices
Design and Evaluate RNNs models for Hotel Description Generation Tasks

Instructor: Prof. Zahangir Alom
Assignment #4 Report

Table of contents

Recurrent Neural Network (RNN) model to predict Google Stock Prices

1. Introduction

A recurrent neural network is a class of artificial neural networks where connections between nodes can create a cycle, allowing output from some nodes to affect subsequent input to the same nodes. The objective of this project is to develop a simple Recurrent Neural Network (RNN) model to predict Google stock prices. Stock price refers to the current price that a share of stock is trading for in the market. The price of a stock will go up and down in relation to a number of different factors, including changes within the economy as a whole, changes within industries, political events, war, and environmental changes.

2. Methodology & Model descriptions

For this report we will use supervised learning using RNN to predict google stock prices. RNN is ideal for this task because information persists throughout the network over time. As the name RNN suggests, the update function is essentially a recurrence relation that happens at every step of the sequential process, see figure 1.

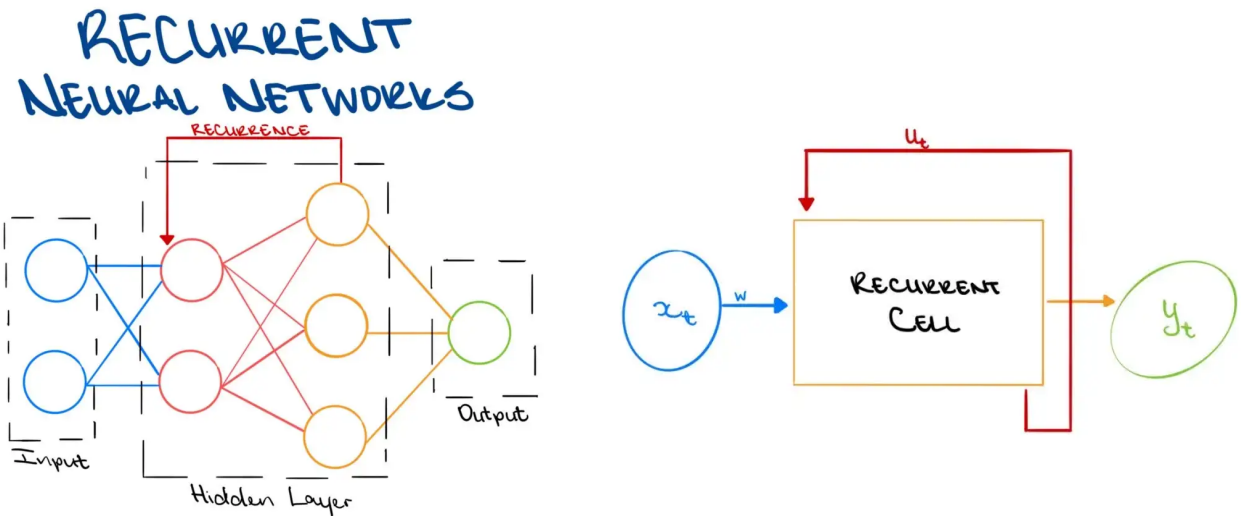


Figure 1. Recurrent Neural Network (RNN) architecture

3. Experiment and Results

Experiment

Step 1 – Importing required libraries.
Step 2 – Reading our training data.
Step 3 – Getting our training data in shape.
Step 4 – Creating the Stock Price Prediction model.
Step 5 – Training the Stock Price Prediction model.
Step 6 – Reading the test data.
Step 7 - Getting the Stock Price Predictions on test data.
Step 8 – Plotting the predictions and real data.

We use keras and python to implement the RNN prediction model on the NVIDIA RTX GPU, with training runtime less than a minute.

Epoch	200	**Loss**	Mean square error
Batch	32	**RNN model**	LSTM from Keras
Optimizer	Adam	**Normalization**	Min Max Scaler

Table 1. Experiment hyperparameters for our RNN model

(a) Database

Google_Stock_Price_Trainset.csv
Google_Stock_Price_Testset.csv

(b) Training and testing logs

Training and testing code be found it the following link:
https://github.com/bereketeshete/Machine-Learning-Projects/tree/main/Homework%204

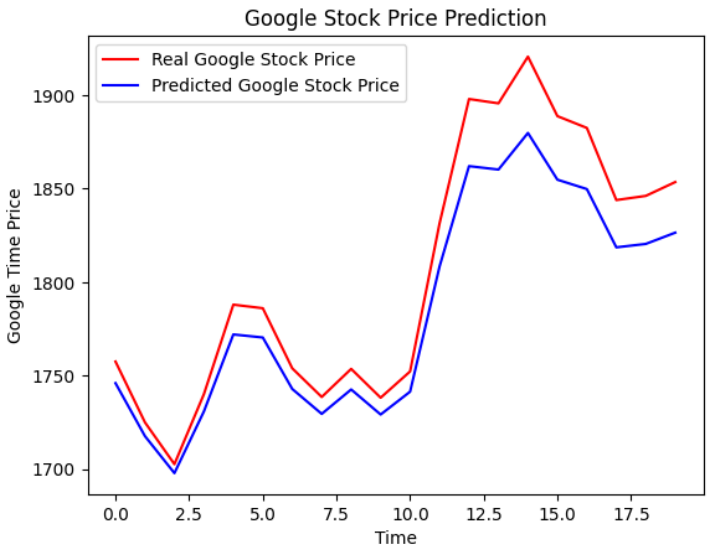


Figure 1. Real Google Stock price is in red and the predicted stock price is in blue

The root mean square error (rmse) between the predicted and the real google stock price is 17.03.

(c) Discussion and comparison

From figure 1 in the result section, we can see that our model is not that perfect but still it is capable of catching the spikes. Neural networks are a crucial component of AI and ML because they enable models to automatically learn from data and combine a version of human learning with excellent computing capacity, However, using a non-sequential structure for a sequential job would lead to subpar model performance and prevent neural networks from reaching their full potential. RNNs are artificial learning systems that continuously update themselves in order to anticipate outcomes with the highest degree of accuracy. This specific task of predicting stocks using RNN illustrates their powerful application where previous information is important.

4. Conclusion

In this report we demonstrated that we can use Recurrent Neural Network (RNN) to predict google stock prices and obtained an rmse of 17.03. This model is a powerful tool, as investing in a stock market is a source of generating substantial wealth and there is huge demand for predicting the stock prices to make the right move to maximize marginal profit.

5. References

[1] https://en.wikipedia.org/wiki/Recurrent_neural_network
[2] https://corporatefinanceinstitute.com/resources/wealth-management/stock-price/
[3] https://machinelearningprojects.net/stock-price-prediction/
[4] https://towardsdatascience.com/introducing-recurrent-neural-networks-f359653d7020

Design and Evaluate RNNs models for Hotel Description Generation Tasks

1. Introduction

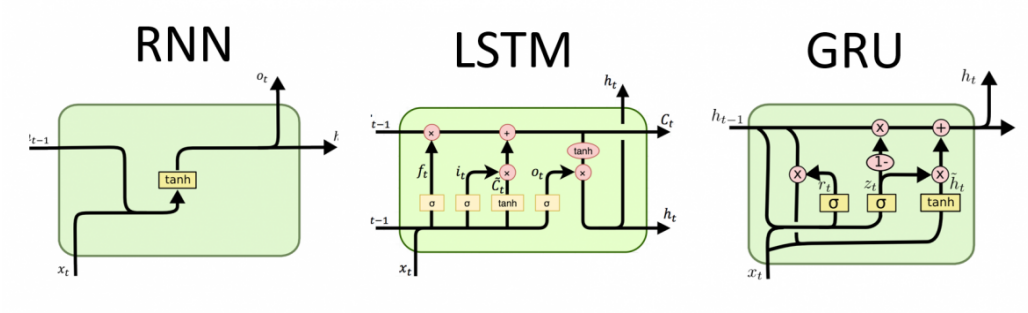
Text generation is a category of Language modeling problem. The main challenge for many natural language processing applications, including speech to text, conversational systems, and text summarization, is language modeling. A linguistic model that has been trained learns a word's likelihood of appearance based on the words that came before it in the text [5]. Our objective is given some input text, the goal of this project is to produce new hotel descriptions. We don't anticipate the results to be precise; as long as the text is coherent as projected, we'll be content [4].

2. Methodology & Model descriptions

RNN (Recurrent Neural Networks) are designed to work with sequential data. Sequential data(can be time-series) can be in the form of text, audio, video etc. RNN uses the previous information in the sequence to produce the current output. RNN's face short-term memory problems that are caused due to vanishing gradient problems.

GRU (Gated Recurrent Units), the workflow of GRU is the same as RNN but the difference is in the operations inside the GRU unit. Inside the GRU it has two gates, a reset gate and an update gate.

LSTM (Long Short-Term Memory) are pretty much similar to GRU's, they are also intended to solve the vanishing gradient problem. Additional to GRU here there are 2 more gates: a forget gate and an)output gate.



SimpleRNN

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 505, 10)           34290

 simple_rnn (SimpleRNN)      (None, 100)               11100

 dropout_1 (Dropout)         (None, 100)               0

 dense_1 (Dense)             (None, 3429)              346329

=================================================================
Total params: 391,719
Trainable params: 391,719
Non-trainable params: 0
_________________________________________________________________
```

LSTM

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 505, 10)           34290

 lstm (LSTM)                 (None, 100)               44400

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 3429)              346329

=================================================================
Total params: 425,019
Trainable params: 425,019
Non-trainable params: 0
```

GRU

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 505, 10)           34290

 gru (GRU)                   (None, 100)               33600

 dropout (Dropout)           (None, 100)               0

 dense (Dense)               (None, 3429)              346329

=================================================================
Total params: 414,219
Trainable params: 414,219
Non-trainable params: 0
```

No of Total Parameters in Order: LSTM > GRU > SimpleRNN

2.1 Text Pre-processing

Tokenization: The process of extracting tokens (terms or words) from a corpus is known as tokenization. To obtain the tokens and their index in the corpus, utilize the tokenization model integrated into the Keras library for Python.

We use Keras' Tokenizer to vectorize text descriptions,

- We remove all punctuation.
- We turn the texts into space-separated sequences of words in lowercase.
- These sequences are then split into lists of tokens.
- We set char_level=False , so every word will be treated as a token other than character.
- The lists of tokens will then be indexed or / and vectorized.
- We convert the corpus into a sequence of tokens.

3. Experiment and Results

We used the python programming language with keras library for this purpose. RNN prediction model was trained on NVIDIA RTX GPU, with training runtime about approximately 10 for each 3 types of RNN models. The output of the three RNN models are shown in the table below.

	desc_1 = generate_text("hilton seattle downtown", 100, model, max_sequence_len)
SimpleRNN	Hilton Seattle Downtown Seattle Airport And A Variety Of Programming And Catering Options And Toiletries Furnishings For Up On A Lumpy Old Mattress A Variety Of A Kind Room With A Bath Of The Most Comprehensive Smoke Free Hot Breakfast Buffet Every Morning And A Seating Area In Our Ultra Fitness Center And A Great Stay At The Master And Building In The City And Northwest Folklife Festivals And Don'T Forget That The Best Western This Hotel Is A Unique Location To Enjoy A Ten Minute Drive From The Space Needle Pike Place Market And The University Of Washington Campus And The Uw
GRU	Hilton Seattle Downtown Hotel Is Located In The Heart Of Downtown Seattle Offers A Few Doors Experience The Night At Queen Anne Is Less Than 2 Miles From The Seattle Mariners And The Sounders Transit Light Rail Ake The Link Light Rail To Downtown Seattle The Ideal Center Of Downtown Seattle Our Hotel Is Also The Best Western Plus Pioneer Square Hotel Is Centrally Located In The University Of Washington Campus College 5 Km Away From The City Of The Museum Of Flight Boeing And Starbucks The Museum Of Flight And Ifly Seattle And Centurylink Field Our Hotel Is Exuberant And Creative Laidback
LSTM	Hilton Seattle Downtown Hotel Is Minutes From The Washington State Convention Center And The Space Needle And The Space Needle And Olympic Lake Union And The Uw Shopping Center If You'Re Traveling For Business At The Best Of Seattle And Enjoy A Variety Of The Ihg Floor Of Landmark And Arched Floors On Its Inviting Experience Experience The Inn Is Purchased Here In The City'S Energy Of The City And The Quiet Magnificence Of The Pacific Northwest Into The Ultimate Panoramic View Of Pike Place Market And The Washington State Convention Center And The Space Needle And The Space Needle And The Famous

Table 1. Comparison of RNN models for generating hotel descriptions. The objective of this model is to generate new text, given that some input text is present.

(a) Database

Seattle_Hotels_address_description.csv

(b) Training and testing logs

Training and testing code and logs can be found it the following link:
https://github.com/bereketeshete/Machine-Learning-Projects/tree/main/Homework%204

(c) Discussion and comparison

In this report, we use popular RNN models; SimpleRNN, LSTM and GRU for generating hotel descriptions. From table 1, we can observe that GRU outputs a coherent, meaningful hotel description paragraph compared to SimpleRNN and LSTM.

4. Conclusion

We accomplished our objective of given some input text, and we produced new hotel descriptions using state of the art RNN models (SimpleRNN, GRU and LSTM). Our results aren't very precise for practical application but rather satisfactory.

5. References

[1] https://github.com/susanli2016/NLP-with-Python/blob/master/Hotel%20Description%20Generation%20LSTM.ipynb
[2] https://aclanthology.org/W19-8647/
[3] https://www.inlg2019.com/assets/papers/44_Paper.pdf
[4] https://towardsdatascience.com/automatically-generate-hotel-descriptions-with-lstm-afa37002d4fc
[5] https://medium.com/@shivambansal36/language-modelling-text-generation-using-lstms-deep-learning-for-nlp-ed36b224b275
[6] https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573