B.Tech. Project Report

entitled

LSTM AND MEMORY NETWORK

Submitted in partial fulfillment

for

the award of the degree of

Bachelor of Technology

by

Samyak Goyal(ROLL NO - 18UCS205)

Dushyant Gupta(ROLL NO - 18UCS175)

Vatsal Mishra(ROLL NO - 18UCS060)

()

Supervisor

Dr. Animesh Chaturvedi

December, 2021

Department of Computer Science & Engineering

THE LNM
INSTITUTE OF INFORMATION TECHNOLOGY,
JAIPUR INDIA

Declaration

We solemnly declare that the work being presented in this project report entitled "LSTM and Memory Network" is an authentic record of my own work carried out during the period 2021 – 2022 under the supervision of Dr. Animesh Chaturvedi.

Neither the source code there in, nor the content of the project report have been copied or downloaded from any other source. I understand that my result grades would be revoked, if later it is found to be so.

CERTIFICATE

This is to certify that the submitted B.Tech. project report entitled "XXX" is an official record of actual work carried out by SAMYAK GOYAL (ROLL NO - 18UCS205), _ DUSHYANT GUPTA (ROLL NO - 18UCS175), _ VATSAL MISHRA (ROLL NO - 18UCS060), _ () under my supervision and guidance in the institute. To the best of my knowledge, the report embodies the record of an authentic work of the candidate, has duly been completed, is up to the desired standard for the purpose of which it is submitted, and fulfills the requirement of the ordinance relating to the B.Tech. degree of the institute.

(Dr. Animesh Chaturvedi)

Department of CSE,
The LNMIIT,
Jaipur – 302031,
India

Head of Department,
Department of CSE,
The LNMIIT,
Jaipur – 302031,
India

Abstract

Stock market is a place where people buy/sell shares of publicly listed companies. It offers a platform to facilitate seamless exchange of shares. In simple terms, if A wants to sell shares of Reliance Industries, the stock market will help him to meet the buyer who is willing to buy Reliance Industries. This project aims to build a machine learning model using the LSTM concept of RNN to predict the opening price for the Google stock for upcoming days based on the previous 90 days data for each day. A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. LSTM: Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate.

Table of Contents

Chapter 1

Introduction

In this project we have used the concepts of LSTM(Long short term memory) and RNN(Recurrent Neural Networks) and tried to propose a model that predicts the opening price for a stock using the LSTM algorithm. Also we researched on memory network.

Stock market is full of volatility. Different stocks shows different degree of volatility. To measure volatility, a metric 'Beta' is used. Beta is a measure of a stock's volatility in relation to the overall market. In India, the National Stock Exchange (NSE) has NIFTY 50 as a benchmark index which was used as a relative point. Beta has a numerical value which shows a story of the stock. In this project, different stocks with different Beta value ranges were picked and were compared.

Memory networks have hidden states and weights. In the second part the aim is to make machines learn the meaning and inference of sentences by training them with fully supervised learning. After the training, the machine starts to infer meaning from a given story and becomes able to answer questions about that particular story. The machine memory can be read and written. Memory networks have larger storage compared to other approaches which try to solve the same kind of Question Answering problems. RNNs are also used for solving question answering problems. They are trained and then they predict the words for a given question. However, the memory of RNNs is not enough to remember facts from the past accurately. On the other hand, memory networks are able to write to and read from external memory to remember the past. The model is trained in fully supervised techniques in order to use its memory effectively. Model Model Architecture A memory network has 4 components I, G, O, and R as follows: I (input feature map):

Takes the input, in this case, lines in a text, and returns its feature representation. G (generalization): Takes the feature representation of a line and adds it to the memory. O (output feature map): Takes the feature representation of a question, finds all of the related memory states with the question, and returns them. R (response): Takes both the feature representation of the question and the related memory states and finds the answer to this question among the dictionaries. 3 The following is the flow of a given a sentence x inside the model: 1. xfeature = I(x) converts x into the internal feature representation. 2. G(xfeature, memory) updates the memory by adding the xfeature to the given memory. 3. outputfeature = O(questionfeature, memory) computes the output features which are highly related to questionfeature. 4. 4. R(outputfeature, vocabulary) produces the textual response according to the outputfeature. During training and testing, the flow above is followed. The only difference between training and testing is the parameter update. Parameters are updated while training, however; they are not updated during the test time. The detailed role of each component is as follows: I component: This component uses the Bag of Words approach to convert a sentence into an inner feature representation. Firstly, the dictionary is created by assigning a number beginning from 1 to each unseen word in the text. This component uses this dictionary to find which vocabularies are present in a given sentence x. It returns a matrix of dimension V $\times$ 1 where V is the length of the dictionary. The values in this matrix are all 0 except the indexes which correspond to a word in the sentence x (these are 1). G component: This component adds the feature representation of a new sentence to the next available slot in the memory. In this case, the memory is reset at the beginning of each story, and in this way, the memory does not become huge and unrelated memory slots are removed. If larger memory is necessary for problems such as large-scale QA dataset cases where there are 14M statements, a better approach would be fixing the size of the memory and forgetting the least used memory slot when the memory is full. O component: This component scores each memory slot according to a given question and finds the most relevant memory slots. The dataset consists of 20 different types of QA tasks and Each requires a different number of supporting facts to answer them. For instance, if the following story is told, it is enough to remember the third and six sentences in order to answer the question: 1 Mary moved to the bathroom. 2 Sandra journeyed to the bedroom. 4 3 Mary got the football there. 4 John went to the kitchen. 5 Mary went back to the kitchen. 6 Mary went back to the garden. 7 Where is the football? garden 3 6 For the above QA problem, each line is added to the memory and when the seventh line is the input, x = "Where is the football?", this O component finds the first relevant fact from the story

In this way, the O component finds both the first and the second supporting facts and the final output of this component is [x, mo1, mo2], and this is the input for the R component. R component: This component scores each vocabulary in the dictionary according to the input which contains the question feature representation and the supporting facts, and finds the most likely answer to the question.

For the second part we did the following:

Reimplemented the memory networks

Memory networks have hidden states and weights. The aim is to make machines learn the meaning and inference of sentences by training them with fully supervised learning. After the training, the machine starts to infer meaning from a given story and becomes able to answer questions about that particular story. The machine memory can be read and written. Memory networks have larger storage compared to other approaches which try to solve the same kind of Question Answering problems. RNNs are also used for solving question answering problems. They are trained and then they predict the words for a given question. However, the memory of RNNs is not enough to remember facts from the past accurately. On the other hand, memory networks are able to write to and read from external memory to remember the past. The model is trained in fully supervised techniques in order to use its memory effectively.

LSTM with Volatile Stocks

Overview Stock market is full of volatility. Different stocks shows different degree of volatility. To measure volatility, a metric 'Beta' is used. Beta is a measure of a stock's volatility in relation to the overall market. In India, the National Stock Exchange (NSE) has NIFTY 50 as a benchmark index which was used as a relative point. Beta has a numerical value which shows a story of the stock. In this project, different stocks with different Beta value ranges were picked and were compared. Volatility measure - Beta. A Beta is a numerical figure that measures how sensitive it is to changes (volatile) in the general stock market. When the benchmark index, such as the NSE Nifty, is compared to the returns of a specific stock, a pattern emerges that reveals the stock's openness to market risk. This allows the investor to choose between a riskier stock that is highly associated with the market (beta above 1) and a less volatile one (beta below 1). If a stock's beta value is 1.5, it suggests that the stock is 50percent more volatile than the market. Likewise, a stock's beta value of 0.7 suggests that the stock is 30percent less volatile than the market.

Calculation of Beta The beta calculation is used by investors to determine if a stock moves in lockstep with the rest of the market. The market that is utilised as a benchmark must be relevant to the stock in order for beta to provide any useful information. Calculating the beta of a bond ETF using the NIFTY 50 as a benchmark, for example, would not provide much useful information to an investor because bonds and stocks are too distinct.

Different ranges of Beta

Beta Value Equal to 1.0: If a stock has a beta of 1.0, it indicates that its price activity is strongly correlated with the market. Beta Value less than 1.0: A beta value that is less than 1.0 means that the security is theoretically less volatile than the market Beta Value more than 1.0: A beta that is greater than 1.0 indicates that the security's price is theoretically more volatile than the market. Negative Beta Value: A beta of -1.0 means that the stock is inversely correlated to the market benchmark.

Chapter 2

Literature Survey and Theoretical Background

Algorithms Used

RNN A recurrent neural network (RNN) is a class of artificial neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Derived from feedforward neural networks, RNNs can use their internal state (memory) to process variable length sequences of inputs. The term "recurrent neural network" is used indiscriminately to refer to two broad classes of networks with a similar general structure, where one is finite impulse and the other is infinite impulse. Both classes of networks exhibit temporal dynamic behavior. A finite impulse recurrent network is a directed acyclic graph that can be unrolled and replaced with a strictly feedforward neural network, while an infinite impulse recurrent network is a directed cyclic graph that can not be unrolled. Both finite impulse and infinite impulse recurrent networks can have additional stored states, and the storage can be under direct control by the neural network. The storage can also be replaced by another network or graph, if that incorporates time delays or has feedback loops. Such controlled states are referred to as gated state or gated memory, and are part of long short-term memory networks (LSTMs) and gated recurrent units. This is also called Feedback Neural Network (FNN).

LSTM Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. A common LSTM unit is composed of a cell, an input gate, an output gate and a forget gate. The cell remembers values over arbitrary time intervals and the three gates regulate the flow of information into and out of the cell. In theory, classic (or "vanilla") RNNs can keep track of arbitrary long-term dependencies in the input sequences. The problem with vanilla RNNs is computational (or practical) in nature: when training a vanilla RNN using back-propagation, the gradients which are back-propagated can "vanish" (that is, they can tend to zero) or "explode" (that is, they can tend to infinity), because of the computations involved in the process, which use finite-precision numbers. RNNs using LSTM units partially solve the vanishing gradient problem, because LSTM units allow gradients to also flow unchanged. However, LSTM networks can still suffer from the exploding gradient problem.

Volatility measure - Beta. A Beta is a numerical figure that measures how sensitive it is to changes (volatile) in the general stock market. When the benchmark index, such as the NSE Nifty, is compared to the returns of a specific stock, a pattern emerges that reveals the stock's openness to market risk. This allows the investor to choose between a riskier stock that is highly associated with the market (beta above 1) and a less volatile one (beta below 1). If a stock's beta value is 1.5, it suggests that the stock is 50 percent more volatile than the market. Likewise, a stock's beta value of 0.7 suggests that the stock is 30 percent less volatile than the market. Calculation of Beta The beta calculation is used by investors to determine if a stock moves in lockstep with the rest of the market. The market that is utilised as a benchmark must be relevant to the stock in order for beta to provide any useful information. Calculating the beta of a bond ETF using the NIFTY 50 as a benchmark, for example, would not provide much useful information to an investor because bonds and stocks are too distinct.

Different ranges of Beta Beta Value Equal to 1.0: If a stock has a beta of 1.0, it indicates that its price activity is strongly correlated with the market. Beta Value less than 1.0: A beta value that is less than 1.0 means that the security is theoretically less volatile than the market Beta Value more than 1.0: A beta that is greater than 1.0 indicates that the security's price is theoretically more volatile than the market. Negative Beta Value: A beta of -1.0 means that the stock is inversely correlated to the market benchmark.

Chapter 3

Proposed Work

Data Used

We have imported our dataset from Yahoo Finance (https://in.finance.yahoo.com/). We have used the stock of Google(symbol: GOOG) to train our model

(https://in.finance.yahoo.com/quote/GOOG?p=GOOG).

The parameters that were imported were: Date: The date of the instance of the stock Open: The opening price of the stock for that particular day High: The highest price achieved by the stock for that particular day Low: The lowest price achieved by the stock for that particular day Close: The closing price of the stock for that particular day Volume: The volume of the stock

The Dataset imported was from 20th August 2004 to 31st December 2020. The downloaded csv file can be seen here: https://github.com/samyakgoyal/Stock-Prediction-Using-LSTM/blob/main/GOOG.csv

We divided the dataset in chunks of 90 days and then trained the model on them. The training was done on the data from 20th August 2004 to 31st December 2018, the testing was done on the data from 1st January 2019 to 1st January 2021. First of all, the columns Date and Adj close were dropped since they were not providing any additional information. Then the remaining features were scaled using MinMaxScaler imported from sklearn.preprocessing. Then the dataset was broken into chunks of 90 days and then the model was trained to predict the opening price for the stock.

Stocks taken and their beta value In this project, we picked stock from all ranges of Beta. The stock's price were compared monthly for 10 years which gives enough time for a stock to show it's true characteristics. The stocks picked were: 1. Reliance Industries: The petrochemical company had the Beta Value of 1.04 (Approximately equal to 1) 2. ITC: The FMCG company had the Beta Value of 0.64 (Less than 1) 3. Tata Motors: This Automobile company had the Beta Value of 1.81 (More than 1) No stocks in the Indian Stock Market had negative Beta value. Negative beta values are for options and debt instruments and out of scope of this project.

Prediction of price using LSTM Just like in the first part of the project, we predict the stock prices using the same technique over 10 years and compare the error percentage (inversely proportional to the accuracy of the method) between all the chosen stocks.

Memory networks are trained using fully supervised settings. During the training, the O component finds the supporting facts, and its loss and accuracy are calculated; then the correct supporting facts with the question are given to the R component as its parameters in order to calculate its loss and accuracy accurately. O component weight and R component weight are trained separately using softmax. In this way, they become experts on their tasks and the risk of misleading the R component due to the incorrect supporting fact choice of the O component is eliminated. However; during the test time, the correct supporting facts and responses are used only for accuracy calculation. The output of the O component is directly given to the R component. Knet.Adam is used in order to update the weights of O and R components. The model is tested with 20 different tasks which each tests the different aspects of understanding of the story, question, and the ability to relate the question to the sentences in the story. 6 The 20 different toy tasks can be summarized as follows: 1. Single Supporting Fact: In order to answer a question, 1 supporting fact along with the question is enough. These questions generally ask the location of a person in a given story. 2. Two or Three Supporting Facts: These tasks are more difficult in comparison with the single supporting fact case. This time two or three supporting facts are necessary in order to answer the question. Two supporting fact questions are usually about the location of an object in the story and three supporting fact questions ask the location of an object before or after something happened such as "Where was the apple before the kitchen? " Example stories for tasks 1 to 10. 3. Two or Three Argument Relations: In the case of the two-argument relations, the order of the words in a sentence is important in order to answer the question, because two sentences with distinct meanings may have the same words. In the three-argument relations, there are two 7 people and one object and the goal is to understand which object is involved, who is the giver and the receiver. 4. Yes/No Questions: This is the simplest task among others. The questions are true/false questions and answers are either "yes" or "no". 5. Counting and List/Set: Task 7 tests the model in terms of the ability to count objects. Questions about counting tasks are usually "How many objects is Daniel holding? ". Task 8 is an advanced version of the questions in Task 7, like "What is Daniel holding? ". 6. Simple Negation and Indefinite Knowledge: Task 9 tests one of the simplest forms of negation, some supporting facts imply a statement is false. Task 10 asks questions which are about uncertainty such as giving a fact that "John is either in the classroom or the playground." and asked "Is John in the classroom? ", then the correct answer is neither yes nor no, the correct answer is "maybe". 7. Basic Coreference, Conjunctions, and Compound Coreference: Task 11 tests the model whether it can find the nearest referent or not. Task 12 asks questions about the conjunction of subjects in a sentence. Task 13 is a combination of Task 11 and task 12, this time one sentence has a referent and the referent has a reference to at least two subjects in another sentence. 8. Time Reasoning: Task 14 tests understanding the use of time expressions within the statements and answering questions according to their time order. 9. Basic Deduction and Induction: These tasks test the model's ability to understand the basic inheritance of properties between sentences. 8 Example stories for tasks 11 to 20. 10. Positional and Size Reasoning: Task 17 tests whether the model is able to find relative positions of colored blocks. Task 18 requires a general understanding of the relative size of different objects. 11. Path Finding: The problem in Task 19 is about going somewhere to somewhere. It asks how do you get from one to another. 12. Agent's Motivation: Task 20 questions test the model understanding of the relation between certain actions and states. For example, the model learns that thirsty people tend to go to the kitchen.

Stocks taken and their beta value

In this project, we picked stock from all ranges of Beta. The stock's price were compared monthly for 10 years which gives enough time for a stock to show it's true characteristics. The stocks picked were: 1. Reliance Industries: The petrochemical company had the Beta Value of 1.04 (Approximately equal to 1) 2. ITC: The FMCG company had the Beta Value of 0.64 (Less than 1) 3. Tata Motors: This Automobile company had the Beta Value of 1.81 (More than 1) No stocks in the Indian Stock Market had negative Beta value. Negative beta values are for options and debt instruments and out of scope of this project.

Chapter 4

Results and Discussion

Code and Model First of all the following libraries were imported: Numpy and pandas : For mathematical and Array manipulation Matplotlib.pyplot : To plot the graph Then the dataset was loaded Splitting the dataset into training and test set Dropping the columns that were not adding any value Feature Scaling the data to normalize the range of independent variables Creating Data with Timestamps: Because our model is trained on the previous 90 days data to predict the upcoming stock price.

Building the LSTM model

First of all the Sequential model was imported from tensorflow.keras because LSTM is a sequential model and we need to combine our layers in a sequential order. Dense and LSTM layers were imported from tensorflow.keras because the LSTM layers provide the basic block for the LSTM model and the Dense layer is used at the end to summarize the model. Then Dropout rate is also imported to avoid the case of underfitting / overfitting. First of all the Sequential model was defined, then 4 LSTM layers were added. Then Different units are used to vary the layers, the activation function is chosen as 'relu' because it provides the best result, also the Dropout is used to avoid the case of overfitting. Then a Dense layer was added. At last the model was compiled using 'Adam' optimiser(provided best result) and the loss was calculated as 'mean squared error' (provided the best insight). Then our dataset was fit into the model and the model was trained for 11 epochs with batch size 32. Test Dataset was prepared. This step was added to revert back the Feature scaling for plotting the Graph. At last the graph was plotted to compare the accuracy of the model: The black line depicts the actual cost and the green line depicts the cost of the stock predicted by our model. Then the error was calculated through averaging the difference between predicted and actual values The last step was to plot the graph of error percentage for better visualization

The model can be found here: Google collab: https://colab.research.google.com/drive/1JQWRwMvc
CFoD6zn0ur7Bil2?authuse r=1

This model without any observable underitting / overitting gives only 6.59 percent average error rate or 93.41percent Accuracy result.

Findings of LSTM Tata Motors (Beta > 1) Error Percentage: 13.59percent 6 Reliance Industries (Beta = 1) Error percentage: 12.01percent 7 ITC (Beta < 1) Error percentage: 11.77 8 Conclusion We see a linear correlation between Error percent and Beta Values. Higher the Beta value (More volatile), Higher the error percentage. Error percent in prediction using LSTM Beta of that stock

Achieved better accuracy on 7 tasks and similar accuracy on 7 tasks, however; got poor accuracy on 6 tasks. The following can cause getting poor accuracy in these tasks: 1. Normally each task other than task 7 and task 8 requires a constant number of supporting facts to answer questions correctly. In tasks 7 and 8, the number of supporting facts needed changes for different questions, but my model assumes that each question requires the same number of supporting facts at the beginning of training, so it uses D = maximum number of supporting facts needed and updates it during the training. This also causes overfitting after a few epochs. 2. The inner feature representation of sentences could be another reason. I could not find the code of authors' implementation on the Internet, thus I implemented my memory network by understanding the paper and watching video lectures online about memory networks. 3. The paper suggests an improvement for memory networks which is called the time feature. A memory network with a time feature knows which sentence comes first and scores memories according to this fact while finding supporting facts, therefore relative time is used. Adding a time feature requires using hinge loss as the loss function, however; hinge loss is not suitable in 15 my implementation since it is not differentiable. Thus, I used absolute time with soft loss instead of relative time with hinge loss. 4. In the original paper, margin ranking loss and SGD is used for training, but I used soft loss and Knet.Adam for training. The implementation has achieved higher accuracy on the following 7 tasks: • Task 3: Three

Supporting Facts 35.5 percent • Task 6: Yes/No Questions 53 percent • Task 9: Simple Negation 68 percent • Task 14: Time Reasoning 100 percent • Task 15: Basic Deduction 100 percent • Task 16: Basic Induction 97.4 percent • Task 19: Path Finding 12.9 percent Higher accuracies are obtained on especially Task 3, Task 15, Task 16, and Task 19 in the implementation. The implementation is not good enough to answer all questions on Task 3 as well, but it has achieved higher accuracy. The paper's implementation is not good enough to answer all of the questions on Task 15 and Task 16, and obtains poor accuracy on these tasks; on the other hand, implementation of memory networks is better on these tasks and answers approximately all of the questions correctly. If Task 19 is considered, the paper's implementation obtained 0 percent accuracy on this task and the agent never finds the correct path. My implementation also does not always give the correct path to the agent, but at least 12.9 percent of the time the agent reaches wherever it wants. Possible reasons why my implementation has achieved higher accuracies on these tasks: 1. used Knet.Adam as the parameter optimizer. 2. used soft loss instead of margin ranking loss. 3. internal representation might be more appropriate for these tasks

Prediction of price using LSTM Just like in the first part of the project, we predict the stock prices using the same technique over 10 years and compare the error percent (inversely proportional to the accuracy of the method) between all the chosen stocks Findings of LSTM Tata Motors (Beta > 1)

Error Percentage: 13.59percent

Reliance Industries (Beta $= 1$)

Error percentage: 12.01percent

ITC (Beta < 1)

Error percentage: 11.77

Conclusion

We see a linear correlation between Error percent and Beta Values. Higher the Beta value (More volatile), Higher the error percentage. Error percent in prediction using LSTM Beta of that stock

Chapter 5

Conclusion and Future Work

This model without any observable underfitting / overfitting gives only 6.59 percent average error rate or 93.41percent Accuracy result.

We see a linear correlation between Error percent and Beta Values. Higher the Beta value (More volatile), Higher the error percentage. Error percent in prediction using LSTM Beta of that stock

Related Work Question Answering is one of the fundamental problems in machine learning. There have been various approaches in the past which include using RNN, LSTM, SVM, etc. Memory Networks work of Weston et al. introduced a new approach called MemNN which is the abbreviation of Memory Neural Networks. They tested their approach with large-scale and toy QA tasks and proved MemNN is able to solve these problems. I used their approach to solve their QA problems with absolute time and tested my model with the dataset which is used for Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. The dataset is more generalizable, comprehensive, and splits the tasks into categories. Conclusion In conclusion, recently introduced new classes of learning models memory networks are re-implemented and tested. As stated in the Analysis and Contributions section, the model has achieved similar and better accuracies on most of the tasks (14 out of 20 tasks) In the other tasks, obtained poor accuracies according to the stated accuracies in Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. The possible reasons for these results are explained in the Analysis and Contributions section. In summary, the internal representation, absolute time vs. relative time, and different numbers of supporting facts cause the accuracy difference.

The results in the Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks is for another implementation of memory networks that uses relative time. Overall the model achieved nearly the same accuracy in terms of mean performance and fulfilled the task of reimplementation of MemNN.

Resources

https://in.finance.yahoo.com/

https://www.youtube.com/watch?v=5tvmMX8r$_{O}Mlist = PLtBw6njQRU-rwp5_{7C0oIVt26ZgjG9NI}$

https://in.finance.yahoo.com/quote/GOOG?p=GOOG

https://paperswithcode.com/method/memory-network , https://arxiv.org/pdf/1410.3916v11.pdf

https://en.wikipedia.org/wiki/Recurrent$_{n}eural_{n}etwork$

https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

20

https://en.wikipedia.org/wiki/Long$_{s}hort - term_{m}emory$

https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-exp erts/

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://keras.io/api/layers/recurrent$_{l}ayers/lstm/$

https://www.tutorialspoint.com/python/index.htm https://www.investopedia.com/terms/b/beta.a https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-netwo rks-experts/ https://finance.yahoo.com

Weston, Jason, Chopra, Sumit, and Bordes, Antoine. 2015. Memory Networks. In ICLR 2015. https://arxiv.org/pdf/1410.3916.pdf Weston, Jason, Bordes, Antoine, Chopra, Sumit, Rush, Alexander M., Merrienboer, Bart van, Joulin Armand, and Mikolov Tomas. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In ICLR 2016. https://arxiv.org/pdf/1502.05698.pdf Video Lecture with Sumit Chopra, http://videolectures.net/deeplearning2016$_{c}hopra_{a}ttention_{m}emory/MemoryNetworksfromJ$ $//www.youtube.com/watch?v = Xumy3Yjq4z$

Chapter 6

Bibliography

We would like to thank our supervisor and mentor Dr. Animesh sir for helping and guiding us.

Also the below sites were helpful https://in.finance.yahoo.com/

https://www.youtube.com/watch?v=5tvmMX8r$_{O}Mlist = PLtBw6njQRU{-}rwp5_{7C0oIVt26ZgjG9NI}$

https://in.finance.yahoo.com/quote/GOOG?p=GOOG

https://paperswithcode.com/method/memory-network , https://arxiv.org/pdf/1410.3916v11.pdf

https://en.wikipedia.org/wiki/Recurrent$_{n}eural_{n}etwork$

https://stanford.edu/ shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks

https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce

20

https://en.wikipedia.org/wiki/Long$_{s}hort - term_{m}emory$

https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-exp erts/

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

https://keras.io/api/layers/recurrent$_{l}ayers/lstm/$

https://www.tutorialspoint.com/python/index.htm https://www.investopedia.com/terms/b/beta.a https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-netwo rks-experts/ https://finance.yahoo.com

Weston, Jason, Chopra, Sumit, and Bordes, Antoine. 2015. Memory Networks. In ICLR 2015. https://arxiv.org/pdf/1410.3916.pdf Weston, Jason, Bordes, Antoine,

Chopra, Sumit, Rush, Alexander M., Merrienboer, Bart van, Joulin Armand, and Mikolov Tomas. 2015. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. In ICLR 2016. https://arxiv.org/pdf/1502.05698.pdf Video Lecture with Sumit Chopra, http://videolectures.net/deeplearning2016$_{c}hopra_{a}ttention_{m}emory/MemoryNetworksfromJ$
$//www.youtube.com/watch?v = Xumy3Yjq4z$

https://www.investopedia.com/terms/b/beta.asp https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-netwo rks-experts/ https://finance.yahoo.com/