STOCK ANALYSIS AND MOVEMENT PREDICTION USING NEWS HEADLINES

A Project report submitted
in partial fulfillement of requirements
for the award of degree of

Bachelor of Technology
In
Information Technology

By

NAGULA SURENDRA **(Reg No: 16131A1275)**
NAMMI GNANESH **(Reg No: 16131A1277)**
SAI PRANEETH GUMMALA **(Reg No: 16131A1293)**
VALLABHAJOSHULA BHARADWAZ **(Reg No: 16131A12A6)**

Under the esteemed guidance of

Mr.D.NAGA TEJ

M.Tech , Assistant Professor.

Department of Information Technology
GAYATRI VIDYA PARISHAD COLLEGE OF ENGINEERING (AUTONOMOUS)
(Affiliated to JNTU-K, Kakinada)
VISAKHAPATNAM

2019 – 2020

Gayatri Vidya Parishad College of Engineering (Autonomous), Visakhapatnam

CERTIFICATE

This report on ***"STOCK ANALYSIS AND MOVMENT PREDICTION USING NEWS HEADLINES"*** is a bonafide record of the main project work submitted

By

NAGULA SURENDRA	**(Reg No: 16131A1275)**
NAMMI GNANESH	**(Reg No: 16131A1277)**
SAI PRANEETH GUMMALLA	**(Reg No: 16131A1293)**
VALLABHAJOSHULA BHARADWAZ	**(Reg No: 16131A12A6)**

in their VIII semester in partial fulfillment of the requirements for the Award of Degree of

Bachelor of Technology

In

Information Technology

During the academic year 2019-2020

Mr.D.NAGA TEJ

Project guide

Dr.K.B.Madhuri,

Head of the Department,

Department of Information Technology

DECLARATION

We hereby declare that this industry oriented main project entitled **"*STOCK ANALYSIS AND MOVEMENT PREDICTION USING NEWS HEADLINES*"** is a bonafide work done by me and submitted to **Department of Information Technology, Gayatri Vidya Parishad College of Engineering (Autonomous), Visakhapatnam,** in partial fulfilment for the award of the degree of B.Tech is of my own and it is not submitted to any other university or has been published any time before.

PLACE: VISAKHAPATNAM

DATE : 0-0-2020

N.SURENDRA (16131A1275)

N.GNANESH (16131A1277)

G.SAI PRANEETH (16131A1293)

V.BHARADWAZ (16131A12A6)

ACKNOWLEDEGEMENT

We would like to take this opportunity to extend our hearty gratitude to our esteemed institute **"Gayatri Vidya Parishad College of Engineering (Autonomous)"** where we got the platform to fulfill our cherished desire.

We express our deep gratitude to **Dr. K.B.MADHURI,** Head of the Department of Information Technology ,Gayatri Vidya Parishad College of Engineering (Autonomous) for her constant support and encouragement.

We express our sincere thanks to **Dr. A.B. KOTESWARA RAO,** PRINCIPAL,Gayatri Vidya Parishad College of Engineering (Autonomous), for his support and encouragement during the course of this project.

We are obliged to **Mr. D.NAGA TEJ ,** Assistant Professor, Department of Information Technology, who has been our guide, whose valuable suggestions, guidance and comprehensive assistance helped us a lot in realizing the project.

We also thank **Mr. D.NAGA TEJ,** Assistant Professor**,** Project Coordinator, Department of Information Technology, for guiding us throughout the project and helping us in completing the project efficiently.

We would like to thank all the members of teaching and non-teaching staff of Department of Information Technology, for all their support.

Lastly, we are grateful to all our friends, for their relentless support in augmenting the value of work, our family, for being considerate and appreciative throughout.

N. SURENDRA (16131A1275)

N.GNANESH (16131A1277)

G.SAI PRANEETH (16131A1293)

V.BHARADWAZ (16131A12A6)

ABSTRACT

STOCK ANALYSIS AND MOVEMENT PREDICTION USING NEWS HEADLINES is extremely important for making wise investment decisions, thus predicting the movement of stock prices demands an efficient and flexible model that is able to capture long range dependencies among news items and provide accurate results.

The goal of this project is to use top, daily news headlines from various sources to predict the movement of the Dow Jones Industrial Average which can be further extended to stocks of individual companies as well.

The news from a given day will be used to predict the difference in opening price between that day and the following day. We would like to develop a deep learning model based on Long Short Term Memory Recurrent Neural Network (LSTM-RNN) to train on our data and use it for predicting on custom headlines.

INDEX

INTRODUCTION

Stock price prediction and its movement is extremely important for making wise investment decisions.This main project deals with creating an deep learning algorithm which uses the top, daily news headlines from various sources to predict the difference in opening price between that day, and the following day. This is created on Jupyter notebook.

1.1 OBJECTIVE

"Stock Analysis And Movement Prediction Using News HeadLines " is an algorithm developed for To predict the stock movement for each and every day and it help the users in making wise investment decisions i.e. which stock to invest on, when to invest etc.

1.2 ABOUT THE ALGORITHM

The algorithm is developed on Jupyter notebook. It is developed using python neural network programming which is an extension of the python language which is specially designed for deep learning algorithm development. It is developed for classifying any type of text data ,In this context it is applied to top daily news headlines from various sources.

1.3 PURPOSE

The purpose of developing the algorithm is to create a state of art algorithm which can be used to predict the opening price and movement of the stock for Each and Every day.

1.4 SCOPE

The scope of the algorithm is not restricted to the system on which it is developed. It can be served as an API to simply pass the headlines and stock prices and get predicted values. It can be used for predicting all the stock prices and not just limited Dow Jones.

SOFTWARE REQUIRMENTS SPECIFICATION

2.1 FUNCTIONAL REQUIRMENTS

A functional requirements defines a function of a system or it component .A function described as a set of inputs, the behavior, and outputs**.**

2.1.1 SOFTWARE REQUIREMENTS

- Operating system: Windows,Mac,Linux
- Programming language: Python
- Anaconda IDE(Integrated Development Environment)
- Python 3.6
- Packages : numpy 1.16.2

  pandas 0.21.0

  tensorflow 1.13.1

  keras 2.1.2

  sklearn 0.0

  nltk 3.4.5

  matplotlib 3.1.2

  re 3.8.1

2.1.1 HARDWARE REQIUREMENTS

- Processor: Intel/ARM processor
- RAM:8GB
- Disk space: 1 TB

2.2 NON FUNCTIONAL REQUIREMENTS

A non- functional requirement is a requirement that specify criteria that can be used to judge the operation of a system, rather than specific behaviors .Nonfunctional requirements are called qualities of a system, there are as follows:

1. Performance
    - Response time

      How long the algorithm takes to classify ?
    - Processing time

      How long the algorithm takes to train on data ?
    - Query and reporting times

      (Considerable when providing an API)

2. Capacity and scalability
    - Throughput

      How many tasks our system needs to handle ? (Considerable when providing an API).
    - Storage

      How much data are we going to need to achieve what we need?
    - Coding standard

      Did we decide on the coding standards for the algorithm and stick to them?

ALGORITHM ANALYSIS

3.1 EXISTING ALGORITHMS

For classifying there are several existing algorithms like RNN(Recurrent Neural Network) ,Decision Trees,SVM(Support Vector Machines) etc. But they lack the accuracy and performance to be completely reliable.

DRAWBACKS OF EXISTING ALGORITHMS:

- Low Accuracy.
- Low Reliability.
- Less Robust.
- Less Adaptability.

3.2 PROPOSED ALGORITHM

To overcome the drawbacks of the existing algorithms, the proposed algorithm has been evolved. Our algorithm primarily focuses on predicting the stock movement for each and every day and to improve the accuracy of existing mechanisms. This helps the users to make wise investment descisions.i.e which stock to invest on and when to invest etc.

ADVANTAGES OF PROPOSED SYSTEM:

- High accuracy compared to existing algorithms.
- It is highly reliable.
- Works on huge amount of data.
- Not restricted to a single scan knowledge extraction

SOFTWARE DESCRIPTION

4.1 ANACONDA IDE

Directly from the platform and without involving DevOps, data scientists can develop and deploy AI and machine learning models rapidly into production. Anaconda provides the tools needed to easily:

- Collect data from files, databases, and data lakes.
- Manage environments with Conda (all package dependencies are taken care of at the time of download).
- Share, collaborate on, and reproduce projects.
- Deploy projects into production with the single click of a button.

Data scientists that work in silos struggle to add value for their organization. That's why Anaconda created an integrated, end-to-end data experience.

4.2 TENSORFLOW:

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for ML applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the **Google Brain** team for internal Google use. It was released under the Apache License 2.0 on November 9, 2015.

A tensor is a **vector** or **matrix** of n-dimensions that represents all types of data. All values in a tensor hold identical data type with a known (or partially known) **shape**. The shape of the data is the dimensionality of the matrix or array.

A tensor can be originated from the input data or the result of a computation. In Tensor Flow, all the operations are conducted inside a **graph**. The graph is a set of computation that takes place successively. Each operation is called an **op node** and is connected to each other.

The graph outlines the ops and connections between the nodes. However, it does not display the values. The edge of the nodes is the tensor, i.e., a way to populate the operation with data.
In Machine Learning, models are feed with a list of objects called **feature vectors**.

A feature vector can be of any data type. The feature vector will usually be the primary input to populate a tensor.

These values will flow into an op node through the tensor and the result of this operation/computation will create a new tensor which in turn will be used in a new operation. All these operations can be viewed in the graph.

4.3 KERAS:

Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-

Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet also is the author of the XCeption deep neural network model. In 2017, Google's TensorFlow team decided to support Keras in TensorFlow's core library. Chollet explained that Keras was conceived to be an interface rather than a standalone machine learning framework. It offers a higher-level, more intuitive set of abstractions that make it easy to develop deep learning models regardless of the computational backend used.

4.4 PANDAS :

pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license. The name is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals.

Pandas are well suited for many different kinds of data:

Tabular data with heterogeneously-typed columns, as in an SQL table or Excel spreadsheet

Ordered and unordered (not necessarily fixed-frequency) time series data.

Arbitrary matrix data (homogeneously typed or heterogeneous) with row and column labels.

Any other form of observational / statistical data sets. The data actually need not be labeled at all to be placed into a panda’s data structure

The two primary data structures of pandas, Series (1-dimensional) and Data Frame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, Data Frame provides everything that R’s data. Frame provides and much more. Pandas are built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

4.5 SKLEARN :

Scikit-learn (formerly **scikits.learn**) is a free software machine learning library for the Python programming language.

It featuresvarious classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy.

4.6 NLTK:

The **Natural Language Toolkit** or more commonly **NLTK**, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit,plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

4.7 MATPLOTLIB:

Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that

helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also. Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an object-oriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Types of plots:

4.8 RE:

A regular expression in a programming language is a special text string used for describing a search pattern. It is extremely useful for extracting information from text such as code, files, log, spreadsheets or even documents.

While using the regular expression the first thing is to recognize is that everything is essentially a character, and we are writing patterns to match a specific sequence of characters also referred as string. ASCII or Latin letters are those that are on your keyboards and Unicode is used to match the foreign text. It includes digits and punctuation and all special characters like $#@! %, etc.

For instance, a regular expression could tell a program to search for specific text from the string and then to print out the result accordingly.

Expression can include:

- Text matching
- Repetition
- Branching
- Pattern-composition etc.

In Python, a regular expression is denoted as RE (REs, regexes or regex pattern) are imported through **re module**. Python supports regular expression through libraries. In Python regular expression supports various things like **Modifiers, Identifiers, and White space characters**.

4.9 FLASK:

Flask was created by Armin Ronacher of Pocoo, an international group of Python enthusiasts formed in 2004.

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Extensions are updated far more frequently than the core Flask program. Applications that use the Flask framework include Pinterest and LinkedIn.

Components:

The microframework Flask is based on the Pocoo projects Werkzeug and Jinja2.

Werkzeug:

Werkzeug is a utility library for the Python programming laungage,in other words a toolkit for Web Server Gateway Interface (WSGI) applications, and is licensed under a BSD License. Werkzeug can

realize software objects for request, response, and utility functions. It can be used to build a custom software framework on top of it and supports Python 2.6, 2.7 and 3.3.

Jinja:

Jinja is a template engine for the Python programming language and is licensed under a BSD License. Similar to the Django web framework, it handles templates in a sandbox.

Features:

- Development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Uses Jinja templating Support for secure cookies (client side sessions)
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation Google App Engine compatibility

PROJECT DESCRIPTION

5.1 PROBLEM DEFINITION

This algorithm developed will help in predicting the stock movement for each and every day and the help the users in making wise investment decisions.

The objective is to scrape the news headlines for the past 4 years along with stock data and divide the data into training, validation and test sets and train an LSTM based RNN model to get the most reliable accuracy. This can be further extended by using the model to test with custom headlines and predict the stock prices.

5.2 PROJECT OVERVIEW

This algorithm is built over Neural network using **LSTM** (Long Short Term Memory) and **Attention** mechanism

- Deep learning is a class of machine learning algorithms that use multiple layers to progressively extract higher level features from raw input.
- LSTM works on the principle of remembering and forgetting information as and when required.
- Attention mechanism aggregates summarized knowledge line by line.

5.3 MECHANISM DESCRIPTION

5.3.1 DEEP LEARNING

Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on artificial neural networks. Learning can be supervised, semi-supervised or unsupervised.

Deep learning architectures such as deep neural networks, deep belief networks, recurrent neural networks and convolution neural networks have been applied to fields including computer vision, speech recognition, natural language processing, audio recognition, social network filtering, machine translation, bioinformatics, drug design, medical image analysis, material inspection and board game programs, where they have produced results comparable to and in some cases superior to human experts.

Artificial Neural Networks (ANNs) were inspired by information processing and distributed communication nodes in biological systems. ANNs have various differences from biological brains. Specifically, neural networks tend to be static and symbolic, while the biological brain of most living organisms is dynamic (plastic) and analog.

5.3.2 LSTM (Long Short Term Memory):

Sequence prediction problems have been around for a long time. They are considered as one of the hardest problems to solve in the data science industry. These include a wide range of problems; from predicting sales to finding patterns in stock markets' data, from understanding movie plots to recognizing your way of speech, from language translations to predicting your next word on your iPhone's keyboard.

Long short Term Memory networks, a.k.a LSTMs have been observed as the most effective solution.

LSTMs have an edge over conventional feed-forward neural networks and RNN in many ways. This is because of their property of selectively remembering patterns for long durations of time.

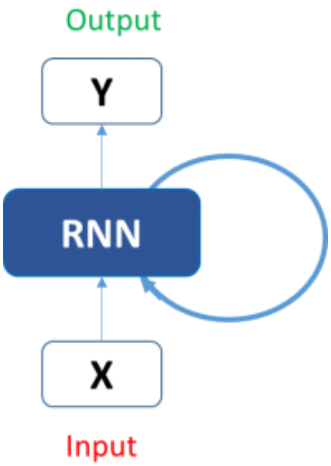
Table of Contents

5.3.2.1. Flashback: A look into Recurrent Neural Networks (RNN)

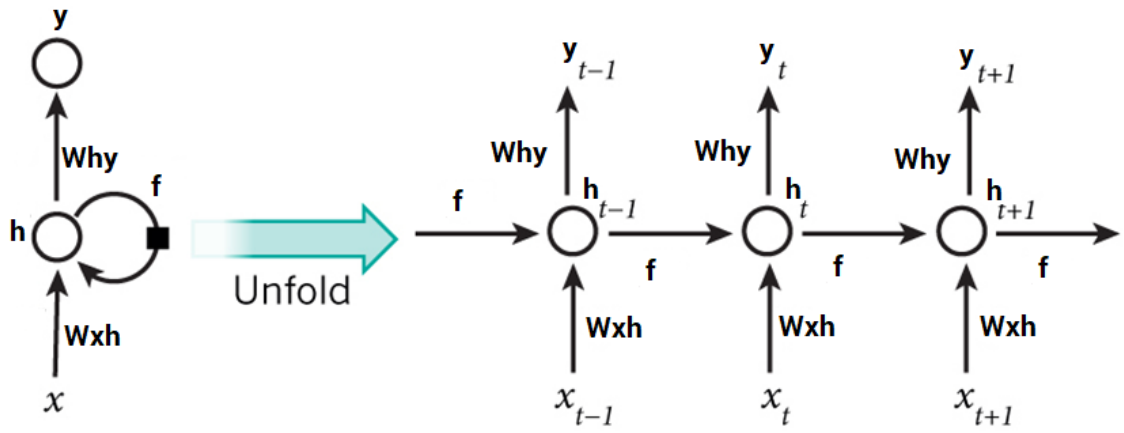
Take an example of sequential data, which can be the stock market's data for a particular stock. A simple machine learning model or an Artificial Neural Network may learn to predict the stock prices based on a number of features: the volume of the stock, the opening value etc. While the price of the stock depends on these features, it is also largely dependent on the stock values in the previous days. In fact for a trader, these values in the previous days (or the trend) is one major deciding factor for predictions.

In the conventional feed-forward neural networks, all test cases are considered to be independent. That is when fitting the model for a particular day, there is no consideration for the stock prices on the previous days.

This dependency on time is achieved via Recurrent Neural Networks. A typical RNN looks like:



This may be intimidating at first sight, but once unfolded, it looks a lot simpler:



Now it is easier for us to visualize how these networks are considering the trend of stock prices, before predicting the stock prices for today. Here every prediction at time t (h_t) is dependent on all previous predictions and the information learned from them.

RNNs can solve our purpose of sequence handling to a great extent but not entirely. Now RNNs are great when it comes to short contexts, but in order to be able to build a story and remember it, we need our models to be able to understand and remember the context behind the sequences, just like a human brain. This is not possible with a simple RNN.

5.3.2.2. Limitations of RNNs

Recurrent Neural Networks work just fine when we are dealing with short-term dependencies. That is when applied to problems like:

The colour of the sky is _____.

RNNs turn out to be quite effective. This is because this problem has nothing to do with the context of the statement. The RNN need not remember what was said before this, or what was its meaning, all they need to know is that in most cases the sky is blue. Thus the prediction would be:

The colour of the sky is blue.

However, vanilla RNNs fail to understand the context behind an input. Something that was said long before, cannot be recalled when making predictions in the present. Let's understand this as an example:

I spent 20 long years working for the under-privileged kids in Spain. I then moved to Africa.

.....

I can speak fluent _____.

Here, we can understand that since the author has worked in Spain for 20 years, it is very likely that he may possess a good command over Spanish. But, to make a proper prediction, the RNN needs to remember this context. The relevant information may be separated from the point where it is needed, by a huge load of irrelevant data. This is where a Recurrent Neural Network fails!

The reason behind this is the problem of **Vanishing Gradient**. . We know that for a conventional feed-forward neural network, the weight updating that is applied on a particular layer is a multiple of the learning rate, the error term from the previous layer and the input to that layer. Thus, the error term for a particular layer is somewhere a product of all previous layers' errors. When dealing with activation functions like the sigmoid function, the small values of its derivatives (occurring in the

error function) gets multiplied multiple times as we move towards the starting layers. As a result of this, the gradient almost vanishes as we move towards the starting layers, and it becomes difficult to train these layers.

A similar case is observed in Recurrent Neural Networks. RNN remembers things for just small durations of time, i.e. if we need the information after a small time it may be reproducible, but once a lot of words are fed in, this information gets lost somewhere. This issue can be resolved by applying a slightly tweaked version of RNNs – the Long Short-Term Memory Networks.

5.3.2.3. Improvement over RNN: LSTM (Long Short-Term Memory) Networks

When we arrange our calendar for the day, we prioritize our appointments right? If in case we need to make some space for anything important we know which meeting could be canceled to accommodate a possible meeting.

Turns out that an RNN doesn't do so. In order to add a new information, it transforms the existing information completely by applying a function. Because of this, the entire information is modified, on the whole, i.e. there is no consideration for **'important'** information and **'not so important'** information.

LSTMs on the other hand, make small modifications to the information by multiplications and additions. With LSTMs, the information flows through a mechanism known as cell states. This way, LSTMs can selectively remember or forget things. The information at a particular cell state has three different dependencies.

We'll visualize this with an example. Let's take the example of predicting stock prices for a particular stock. The stock price of today will depend upon:

1. The trend that the stock has been following in the previous days, maybe a downtrend or an uptrend.

2. The price of the stock on the previous day, because many traders compare the stock's previous day price before buying it.
3. The factors that can affect the price of the stock for today. This can be a new company policy that is being criticized widely, or a drop in the company's profit, or maybe an unexpected change in the senior leadership of the company.

These dependencies can be generalized to any problem as:

1. The previous cell state (i.e. the information that was present in the memory after the previous time step)
2. The previous hidden state (i.e. this is the same as the output of the previous cell)
3. The input at the current time step (i.e. the new information that is being fed in at that moment)

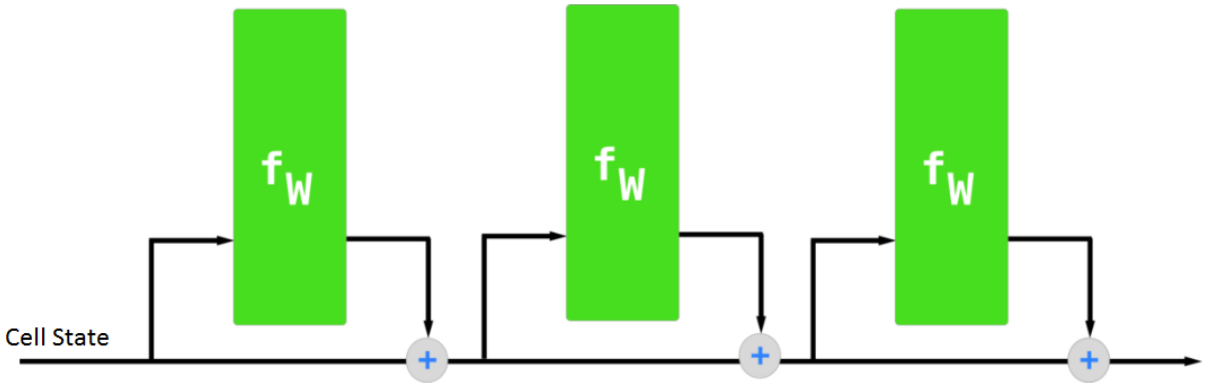
Another important feature of LSTM is its analogy with conveyor belts!

That's right!

Industries use them to move products around for different processes. LSTMs use this mechanism to move information around.

We may have some addition, modification or removal of information as it flows through the different layers, just like a product may be molded, painted or packed while it is on a conveyor belt.

The following diagram explains the close relationship of LSTMs and conveyor belts.



Although this diagram is not even close to the actual architecture of an LSTM, it solves our purpose for now.

Just because of this property of LSTMs, where they do not manipulate the entire information but rather modify them slightly, they are able to *forget* and *remember* things selectively. How do they do so, is what we are going to see below.

5.3.2.4. Architecture of LSTMs

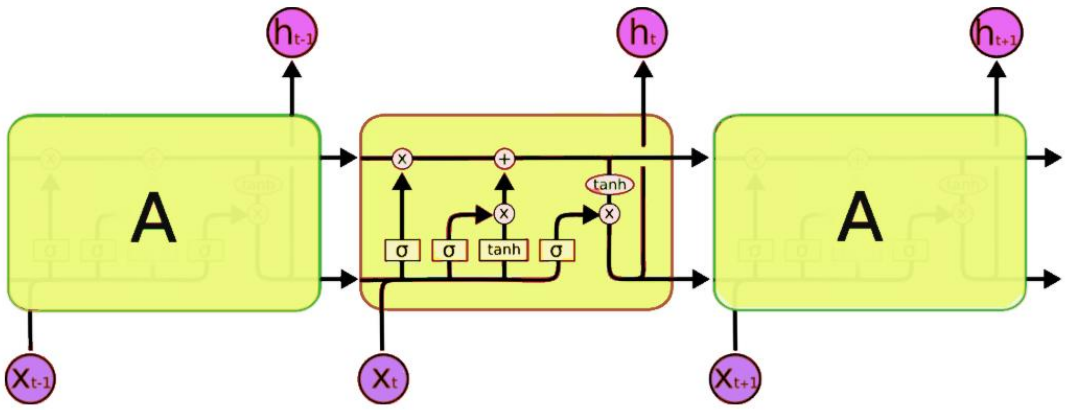
The functioning of LSTM can be visualized by understanding the functioning of a news channel's team covering a murder story. Now, a news story is built around facts, evidence and statements of many people. Whenever a new event occurs you take either of the three steps.

Let's say, we were assuming that the murder was done by 'poisoning' the victim, but the autopsy report that just came in said that the cause of death was 'an impact on the head'. Being a part of this news team what do you do? You immediately forget the previous cause of death and all stories that were woven around this fact.

What, if an entirely new suspect is introduced into the picture. A person who had grudges with the victim and could be the murderer? You input this information into your news feed, right?

Now all these broken pieces of information cannot be served on mainstream media. So, after a certain time interval, you need to summarize this information and output the relevant things to your audience. Maybe in the form of "*XYZ turns out to be the prime suspect.*".

Now let's get into the details of the architecture of LSTM network:



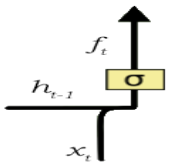
A typical LSTM network is comprised of different memory blocks called cells (the rectangles that we see in the image). There are two states that are being transferred to the next cell; the cell state and the hidden state. The memory blocks are responsible for remembering things and manipulations to this memory is done through three major mechanisms, called gates. Each of them is being discussed below.

Forget Gate:

Taking the example of a text prediction problem. Let's assume an LSTM is fed in, the following sentence:

Bob is a nice person. Dan on the other hand is evil.

As soon as the first full stop after "*person"* is encountered, the forget gate realizes that there may be a change of context in the next sentence. As a result of this, the *subject* of the sentence is *forgotten* and the place for the subject is vacated. And when we start speaking about "*Dan"* this position of the subject is allocated to "*Dan"*. This process of forgetting the subject is brought about by the forget gate



A forget gate is responsible for removing information from the cell state. The information that is no longer required for the LSTM to understand things or the information that is of less importance is removed via multiplication of a filter. This is required for optimizing the performance of the LSTM network.

This gate takes in two inputs; h_t-1 and x_t.

h_t-1 is the hidden state from the previous cell or the output of the previous cell and x_t is the input at that particular time step. The given inputs are multiplied by the weight matrices and a bias is added. Following this, the sigmoid function is applied to this value. The sigmoid function outputs a vector, with values ranging from 0 to 1, corresponding to each number in the cell state. Basically, the sigmoid function is responsible for deciding which values to keep and which to discard. If a '0' is output for a particular value in the cell state, it means that the forget gate wants the cell state to forget that piece of information completely. Similarly, a '1' means that the forget gate wants to remember that entire piece of information. This vector output from the sigmoid function is multiplied to the cell state.

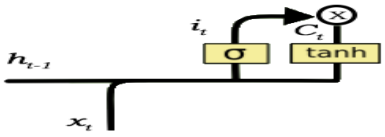
Input Gate:

Let's take another example where the LSTM is analyzing a sentence:

Bob knows swimming. He told me over the phone that he had served the navy for 4 long years.

Now the important information here is that "Bob" knows swimming and that he has served the Navy for four years. This can be added to the cell state, however, the fact that he told all this over the phone is a less important fact and can be ignored.

Here is its structure:



The input gate is responsible for the addition of information to the cell state. This addition of information is basically three-step process as seen from the diagram above.

1. Regulating what values need to be added to the cell state by involving a sigmoid function. This is basically very similar to the forget gate and acts as a filter for all the information from h_t-1 and x_t.
2. Creating a vector containing all possible values that can be added (as perceived from h_t-1 and x_t) to the cell state. This is done using the tanh function, which outputs values from -1 to +1.
3. Multiplying the value of the regulatory filter (the sigmoid gate) to the created vector (the tanh function) and then adding this useful information to the cell state via addition operation.

Once this three-step process is done with, we ensure that only that information is added to the cell state that is *important* and is not *redundant.*

Output Gate:

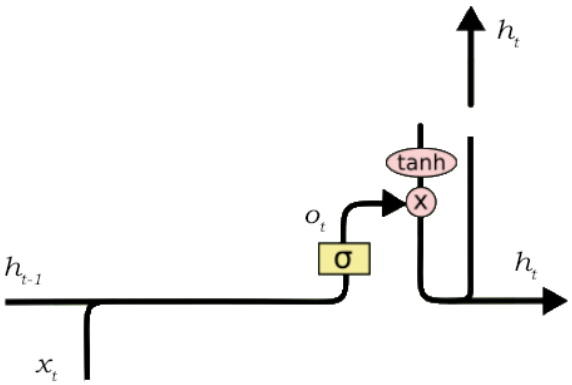
Not all information that runs along the cell state, is fit for being output at a certain time. We'll visualize this with an example:

Bob fought single handedly with the enemy and died for his country. For his contributions brave _____.

In this phrase, there could be a number of options for the empty space. But we know that the current input of *'brave',* is an adjective that is used to describe a noun. Thus, whatever word follows, has a strong tendency of being a noun. And thus, Bob could be an apt output.

This job of selecting useful information from the current cell state and showing it out as an output is done via the output gate.

Here is its structure:



The functioning of an output gate can again be broken down to three steps:

1. Creating a vector after applying tanh function to the cell state, thereby scaling the values to the range -1 to +1.
2. Making a filter using the values of h_t-1 and x_t, such that it can regulate the values that need to be output from the vector created above. This filter again employs a sigmoid function.
3. Multiplying the value of this regulatory filter to the vector created in step 1, and sending it out as a output and also to the hidden state of the next cell.

The filter in the above example will make sure that it diminishes all other values but *'Bob'*. Thus the filter needs to be built on the input and hidden state values and be applied on the cell state vector.

5.3.3 ABOUT GLOVE:

5.3.3.1: INTRODUCTION:

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

5.3.3.2:Highlights:

Nearest Neighbors :

The Euclidean distance (or cosine similarity) between two word vectors provides an effective method for measuring the linguistic or semantic similarity of the corresponding words. Sometimes, the nearest neighbors according to this metric reveal rare but relevant words that lie outside an average human's vocabulary. For example, here are the closest words to the target word *frog*:

a. *frog*

b. frogs

c. toad

d. litoria

e. leptodactylidae

f. rana

g. lizard

h. eleutherodactylus

3. litoria

4. leptodactylidae

5. rana

7. eleutherodactylus

Linear substructures:

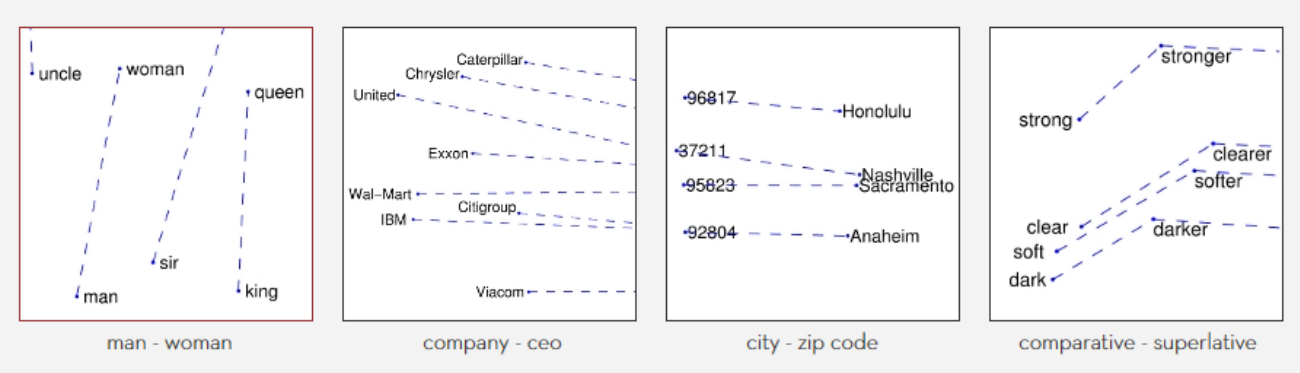
The similarity metrics used for nearest neighbor evaluations produce a single scalar that quantifies the relatedness of two words. This simplicity can be problematic since two given words almost always exhibit more intricate relationships than can be captured by a single number.

For example, *man* may be regarded as similar to *woman* in that both words describe human beings; on the other hand, the two words are often considered opposites since they highlight a primary axis along which humans differ from one another.

In order to capture in a quantitative way the nuance necessary to distinguish *man* from *woman*, it is necessary for a model to associate more than a single number to the word pair.

A natural and simple candidate for an enlarged set of discriminative numbers is the vector difference between the two word vectors. GloVe is designed in order that such vector differences capture as much as possible the meaning specified by the juxtaposition of two words.

The underlying concept that distinguishes man from woman, i.e. sex or gender, may be equivalently specified by various other word pairs, such as king and queen or brother and sister. To state this observation mathematically, we might expect that the vector differences man - woman, king - queen, and brother - sister might all be roughly equal. This property and other interesting patterns can be observed in the above set of visualizations.



Training

The GloVe model is trained on the non-zero entries of a global word-word co-occurrence matrix, which tabulates how frequently words co-occur with one another in a given corpus. Populating this matrix requires a single pass through the entire corpus to collect the statistics. For large corpora, this pass can be computationally expensive, but it is a one-time up-front cost.

Subsequent training iterations are much faster because the number of non-zero matrix entries is typically much smaller than the total number of words in the corpus. The tools provided in this package automate the collection and preparation of co-occurrence statistics for input into the model. The core training code is separated from these preprocessing steps and can be executed independently.

5.3.3.3 Model Overview :

GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. For example, consider the co-occurrence probabilities for target words *ice* and *steam* with various probe words from the vocabulary. Here are some actual probabilities from a 6 billion word corpus:

As one might expect, *ice* co-occurs more frequently with *solid* than it does with *gas*, whereas *steam* co-occurs more frequently with *gas* than it does with *solid*. Both words co-occur with their shared property *water* frequently, and both co-occur with the unrelated word *fashion* infrequently.

Only in the ratio of probabilities does noise from non-discriminative words like *water* and *fashion* cancel out, so that large values (much greater than 1) correlate well with properties specific to ice, and small values (much less than 1) correlate well with properties specific of steam. In this way, the ratio of probabilities encodes some crude form of meaning associated with the abstract concept of thermodynamic phase. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence. Owing to the fact that the logarithm of a ratio equals the difference of logarithms, this objective associates (the logarithm of) ratios of co-occurrence probabilities with vector differences in the word vector space. Because these ratios can encode some form of meaning, this information gets encoded as vector differences as well.

For this reason, the resulting word vectors perform very well on word analogy tasks, such as those examined in the word2vec package.

5.3.4 WORD EMBEDDINGS:

5.3.4.1 What are word embeddings?

"Word embeddings" are a family of natural language processing techniques aiming at mapping semantic meaning into a geometric space. This is done by associating a numeric vector to every word in a dictionary, such that the distance (e.g. L2 distance or more commonly cosine distance) between any two vectors would capture part of the semantic relationship between the two associated words. The geometric space formed by these vectors is called an *embedding space*.

For instance, "coconut" and "polar bear" are words that are semantically quite different, so a reasonable embedding space would represent them as vectors that would be very far apart. But "kitchen" and "dinner" are related words, so they should be embedded close to each other.

Ideally, in a good embeddings space, the "path" (a vector) to go from "kitchen" and "dinner" would capture precisely the semantic relationship between these two concepts. In this case the relationship is "where x occurs", so you would expect the vector kitchen - dinner (difference of the two embedding vectors, i.e. path to go from dinner to kitchen) to capture this "where x occurs" relationship. Basically, we should have the vectorial identity: dinner + (where x occurs) = kitchen (at least approximately). If that's indeed the case, then we can use such a relationship vector to answer questions. For instance, starting from a new vector, e.g. "work", and applying this relationship vector, we should get sometime meaningful, e.g. work + (where x occurs) = office, answering "where does work occur?".

Word embeddings are computed by applying dimensionality reduction techniques to datasets of co-occurence statistics between words in a corpus of text. This can be done via neural networks (the "word2vec" technique), or via matrix factorization.

5.3.4.2 GloVe word embeddings

We will be using GloVe embeddings, which you can read about here. GloVe stands for "Global Vectors for Word Representation". It's a somewhat popular embedding technique based on factorizing a matrix of word co-occurence statistics.

Specifically, we will use the 300-dimensional GloVe embeddings of 400k words computed on a 2014 dump of English Wikipedia.

5.3.4.3 Approach

Here's how we will solve the classification problem:

- convert all text samples in the dataset into sequences of word indices. A "word index" would simply be an integer ID for the word. We will only consider the top 20,000 most commonly occurring words in the dataset, and we will truncate the sequences to a maximum length of 1000 words.
- prepare an "embedding matrix" which will contain at index I the embedding vector for the word of index I in our word index.
- load this embedding matrix into a Keras Embedding layer, set to be frozen (its weights, the embedding vectors, will not be updated during training).
- Build on top of it a convolution neural network, ending in a softmax output over our 20 categories.

DEVELOPMENT

6.1 DATASET:

Headlines: (From Kaggle)

Date,News

2016-07-01,"A 117-year-old woman in Mexico City finally received her birth certificate, and died a few hours later. Trinidad Alvarez Lira had waited years for proof that she had been born in 1898."

2016-07-01,IMF chief backs Athens as permanent Olympic host

2016-07-01,"The president of France says if Brexit won, so can Donald Trump"

2016-07-01,British Man Who Must Give Police 24 Hours' Notice of Sex Threatens Hunger Strike: The man is the subject of a sexual risk order despite having never been convicted of a crime.

2016-07-01,100+ Nobel laureates urge Greenpeace to stop opposing GMOs

2016-07-01,Brazil: Huge spike in number of police killings in Rio ahead of Olympics

2016-07-01,Austria's highest court annuls presidential election narrowly lost by right-wing candidate.

2016-07-01,"Facebook wins privacy case, can track any Belgian it wants: Doesn't matter if Internet users are logged into Facebook or not"

2016-07-01,"Switzerland denies Muslim girls citizenship after they refuse to swim with boys at school: The 12- and 14-year-old will no longer be considered for naturalised citizenship because they have not complied with the school curriculum, authorities in Basel said"

2016-07-01,"China kills millions of innocent meditators for their organs, report finds"

2016-07-01,"France Cracks Down on Factory Farms - A viral video campaign has moved the govt to act. In footage shared widely online, animals writhe in pain as they bleed to death or are dismembered, in violation of rules requiring they be rendered unconscious before slaughter."

2016-07-01,Abbas PLO Faction Calls Killer of 13-Year-Old American-Israeli Girl a Martyr

2016-07-01,Taiwanese warship accidentally fires missile towards China

2016-07-01,"Iran celebrates American Human Rights Week, mocks U.S. rights record"

2016-07-01,U.N. panel moves to curb bias against L.G.B.T. people despite fierce resistance from Muslim and African countries.

2016-07-01,"The United States has placed Myanmar, Uzbekistan, Sudan and Haiti on its list of worst human trafficking offenders."

2016-07-01,S&P revises European Union credit rating to 'AA' from 'AA+'

2016-07-01,India gets $1 billion loan from World Bank for solar mission

2016-07-01,U.S. sailors detained by Iran spoke too much under interrogation: Navy

2016-07-01,Mass fish kill in Vietnam solved as Taiwan steelmaker accepts responsibility for pollution

2016-07-01,"Philippines president Rodrigo Duterte urges people to kill drug addicts | Duterte, 71, won power in a landslide after a campaign dominated by threats to kill tens of thousands in a war on crime"

2016-07-01,Spain arrests three Pakistanis accused of promoting militancy

2016-07-01,"Venezuela, where anger over food shortages is still mounting, continued to be roiled this week by angry protests and break-ins of grocery stores and businesses that have left five dead, at least 30 injured and 200 arrested, according to various news reports."

2016-07-01,"A Hindu temple worker has been killed by three men on a motorcycle, local police have said. More than 40 people have been killed in attacks blamed on Islamist militants in Bangladesh since February 2013, including secular bloggers, academics, gay rights activists and members of religious minorities."

2016-07-01,"Ozone layer hole seems to be healing - US & UK team shows it's shrunk & may slowly recover. ""If you had to have an ozone hole anywhere in the world, it'd be Antarctica because its not teeming with life. It showed us if we didnt back off with these chemicals, wed have a crisis."""

2016-06-30,Jamaica proposes marijuana dispensers for tourists at airports following legalisation: The kiosks and desks would give people a license to purchase up to 2 ounces of the drug to use during their stay

2016-06-30,Stephen Hawking says pollution and 'stupidity' still biggest threats to mankind: we have certainly not become less greedy or less stupid in our treatment of the environment over the past decade

2016-06-30,Boris Johnson says he will not run for Tory party leadership

2016-06-30,Six gay men in Ivory Coast were abused and forced to flee their homes after they were pictured signing a condolence book for victims of the recent attack on a gay nightclub in Florida

2016-06-30,Switzerland denies citizenship to Muslim immigrant girls who refused to swim with boys: report

2016-06-30,Palestinian terrorist stabs israeli teen girl to death in her bedroom

2016-06-30,Puerto Rico will default on $1 billion of debt on Friday

2016-06-30,Republic of Ireland fans to be awarded medal for sportsmanship by Paris mayor.

2016-06-30,Afghan suicide bomber 'kills up to 40' - BBC News

2016-06-30,"US airstrikes kill at least 250 ISIS fighters in convoy outside Fallujah, official says"

2016-06-30,Turkish Cop Who Took Down Istanbul Gunman Hailed a Hero

2016-06-30,"Cannabis compounds could treat Alzheimer's by removing plaque-forming proteins from brain cells, research suggests"

2016-06-30,"Japan's top court has approved blanket surveillance of the country's Muslims: 'They made us terrorist suspects, we never did anything wrong,' says Japanese Muslim, Mohammed Fujita"

2016-06-30,CIA Gave Romania Millions to Host Secret Prisons

2016-06-30,Groups urge U.N. to suspend Saudi Arabia from rights council

2016-06-30,Googles free wifi at Indian railway stations is better than most of the countrys paid services

2016-06-30,"Mounting evidence suggests 'hobbits' were wiped out by modern humans' ancestors 50,000 years ago."

2016-06-30,"The men who carried out Tuesday's terror attack at Istanbul's Ataturk Airport were from Russia, Uzbekistan and Kyrgyzstan, a Turkish offical said."

2016-06-30,Calls to suspend Saudi Arabia from UN Human Rights Council because of military aggresion in Yemen

2016-06-30,More Than 100 Nobel Laureates Call Out Greenpeace For Anti-GMO Obstruction In Developing World

2016-06-30,"British pedophile sentenced to 85 years in US for trafficking child abuse images: Domminich Shaw, a kingpin of sexual violence against children, sent dozens of images online and discussed plans to assault and kill a child while on probation"

2016-06-30,"US permitted 1,200 offshore fracks in Gulf of Mexico between 2010 and 2014 and allowed 72 billion gallons of chemical discharge in 2014."

2016-06-30,We will be swimming in ridicule - French beach police to carry guns while in swimming trunks: Police lifeguards on Frances busiest beaches will carry guns and bullet-proof vests for the first time this summer amid fears that terrorists could target holidaymakers.

2016-06-30,UEFA says no minutes of silence for Istanbul victims at Euro 2016 because 'Turkey have already been eliminated'

2016-06-30,Law Enforcement Sources: Gun Used in Paris Terrorist Attacks Came from Phoenix

2016-06-29,Explosion At Airport In Istanbul

2016-06-29,Yemeni former president: Terrorism is the offspring of Wahhabism of Al Saud regime

2016-06-29,UK must accept freedom of movement to access EU Market

2016-06-29,"Devastated: scientists too late to captive breed mammal lost to climate change - Australian conservationists spent 5 months obtaining permissions & planning for a captive breeding program. But when they arrived on the rodents tiny island, they they were too late."

2016-06-29,British Labor Party leader Jeremy Corbyn loses a no-confidence vote but refuses to resign

2016-06-29,A Muslim Shop in the UK Was Just Firebombed While People Were Inside

2016-06-29,Mexican Authorities Sexually Torture Women in Prison

2016-06-29,UK shares and pound continue to recover

2016-06-29,Iceland historian Johannesson wins presidential election

2016-06-29,"99-Million-Yr-Old Bird Wings Found Encased in Amber - Finding things trapped in amber is far from rare. But when researchers in Burma found a pair of tiny bird-like wings frozen inside, they knew they had something special."

2016-06-29,"A chatbot programmed by a British teenager has successfully challenged 160,000 parking tickets since its launch last year."

2016-06-29,"The Philippine president-elect said Monday he would aggressively promote artificial birth control in the country even at the risk of getting in a fight with the dominant Catholic church, which staunchly opposes the use of contraceptives."

2016-06-29,Former Belgian Prime Minister ridicules Nigel Farage and accuses Ukip leader of lying in EU referendum campaign

2016-06-29,"Brexiteer Nigel Farage To EU: 'You're Not Laughing Now, Are You?'"

2016-06-29,Islamic State bombings in southern Yemen kill 38 people

2016-06-29,"Escape Tunnel, Dug by Hand, Is Found at Holocaust Massacre Site"

2016-06-29,"The land under Beijing is sinking by as much as four inches per year because of the overconsumption of groundwater, according to new research."

2016-06-29,"Car bomb and Anti-Islamic attack on Mosque in Perth, Australia"

2016-06-29,Emaciated lions in Taiz Zoo are trapped in blood-soaked cages and left to starve for months due to the Yemeni civil war

2016-06-29,Rupert Murdoch describes Brexit as 'wonderful'. The media mogul likened leaving the EU to a prison break and shared his view of Donald Trump as a very able man.

2016-06-29,More than 40 killed in Yemen suicide attacks

2016-06-29,Google Found Disastrous Symantec and Norton Vulnerabilities That Are 'As Bad As It Gets

2016-06-29,"Extremist violence on the rise in Germany: Domestic intelligence agency says far-right, far-

2015-09-17,Burkina Faso 'coup': Presidential guard dissolves government

2015-09-17,"Malicious Cisco router backdoor found on 79 more devices, 25 in the US"

2015-09-17,Russian Authorities Close Down American Center in Moscow: The Russian government has shut down the U.S. Embassy's American Center in Moscow after 22 years in operation

2015-09-16,"Tuna and mackerel populations suffer catastrophic 74% decline, research shows"

2015-09-16,"Australian Government introduces ""No Jab No Pay"" legislation, preventing parents of unvaccinated children from receiving childcare benefits."

2015-09-16,"Norway will soon pay Brazil the final instalment of a $1bn payment for slowing the rate at which the Amazon rainforest is cut down, and is in talks about a further scheme."

2015-09-16,Air Canada pilot diverts international flight to save dog from freezing.

2015-09-16,"Nepal to stay secular, proposal for a Hindu nation rejected"

2015-09-16,"'If you are worried about refugees, stop supporting terrorists' - Assad interview"

2015-09-16,"Exxon's Own Research Confirmed Fossil Fuels' Role in Global Warming Decades Ago | ""Over the past several years a clear scientific consensus has emerged,"" Cohen wrote in September 1982, reporting on Exxon's own analysis of climate models."

2015-09-16,"Scared By Russia, Sweden And Poland Make War Pact"

2015-09-16,Netherlands bans wild animals in circuses

2015-09-16,"Hungary convicts migrant for illegally crossing into country, first conviction under new law"

2015-09-16,Germany can stop cash for jobless EU migrants - The European Court of Justice ruled on Tuesday that Germany can deny basic welfare payments to European migrants - even if they've previously had a job in the country.

2015-09-16,"The Leap Manifesto, ""an ambitious plan to end fossil fuel subsidies, increase income taxes on corporations and the wealthy, cut military spending and implement a progressive carbon tax"", signed by at least one hundred notable Canadian personalities"

2015-09-16,David Attenborough backs huge Apollo-style clean energy research plan

2015-09-16,Google to raise $11 million for refugee crisis in Europe in 1:1 donation-matching campaign

2015-09-16,"UK drops out of top 10 renewable energy ranking for 1st time - Conservative govt has sentenced renewable energy to death by 1000 cuts & left investors puzzled, says EY report. ""Weve entered a really weird phase where weve almost got govt energy policy being set in a vacuum."""

2015-09-16,"Pope warns religious orders: Take in refugees, or pay property taxes."

2015-09-16,Iranian Female Soccer Star Faces Husband-Imposed Travel Ban

2015-09-16,WWF - Failing fisheries and poor ocean health starving human food supply. Their report shows a decline of 49 per cent of marine populations between 1970 and 2012.

2015-09-16,"Jeremy Corbyn has said Labour can win the 2020 general election by creating a radical grassroots movement.They call us deficit deniers; they spend billions cutting taxes for the richest families and for the most profitable businesses. What they are is poverty deniers,"

2015-09-16,Croatia says it will allow migrants to travel on to northern Europe - opening up a new route a day after Hungary sealed its border with Serbia.

2015-09-16,"Flying Korea's farmed dogs to safety - ""Our goal is to end the dog-meat industry in Korea,"""

2015-09-16,Turkish presidents office says insulting president not within freedom of expression

2015-09-16,Saudi suspends Binladen group over Mecca crane crash - royal court

2015-09-16,"Anheuser-Busch InBev, the maker of Budweiser and Corona, said Wednesday it has approached rival SABMiller Plc for a potential takeover that would combine the world's two biggest brewers"

2015-09-16,China stocks resume sharp slide as economic worries mount

2015-09-15,"Egyptian Billionaire who wants to purchase private islands to house refugees, has identified potential locations and is now in talks to purchase two private Greek islands"

2015-09-15,The UN Says US Drone Strikes in Yemen Targeting al Qaeda Have Killed More Civilians Than al Qaeda

2015-09-15,Saudis Accused of Not Taking Refugees Despite 100K Empty Tents

2015-09-15,"Denmark has said it will not participate in the EUs plans to resettle some 160,000 refugees."

2015-09-15,US troops return to Iraq to battle Islamic State

2015-09-15,West 'ignored Russian offer in 2012 to have Syria's Assad step aside'

2015-09-15,Hungary Declares Emergency As It Blocks Migrants At Border

2015-09-15,"Whales to gain 'long-sought protections' as navy limits sonar use, activists say"

2015-09-15,Thousands of refugees may lose right of asylum under EU plans - Brussels meeting is expected to call for establishment of refugee camps in Italy and Greece and for detention of irregular migrants

2015-09-15,"Two decades of frontier-free travel across Europe unraveled on Monday as countries re-established border controls in the face of an unprecedented influx of migrants, which broke the record for the most arrivals by land in a single day."

2015-09-15,"According to a new study, half of the sea turtles on the planet have ingested some form of plastic. This comes just days after another study (with some of the same researchers involved) reported similar findings in seabirds -- some 90 percent of which have consumed plastic."

2015-09-15,"Tens of thousands of people have already left Central America for the United States, fleeing violence and poverty.Extreme weather could drive further migration from the Central American countries, particularly Honduras, Guatemala and El Salvador."

2015-09-15,Tourists interrupt endangered turtle reproduction season in Costa Rican beach

2015-09-15,FOIA Reveals TPP Has Immigration Chapter

2015-09-15,Australian mother on trial for genital mutilation

2015-09-15,Hundreds quarantined as Ebola returns to north Sierra Leone district

2015-09-15,"They are calling it the great Indian gold rush. Within months, Indian officials are expected to auction licences for new gold mines across the country, and abandoned colonial-era mines are set to be revived."

2015-09-15,"Okinawa governor to block construction of new US airbase: Takeshi Onaga says he will revoke permits for offshore base in Henokos pristine waters, opposing Japans central government"

2015-09-15,Ex FIFA adviser: Blatter should face a criminal investigation for TV rights deal

2015-09-15,"City officials have suspended the mayor of Bucharest who is under arrest on suspicion of taking bribes worth 25,000 euros ($28,000) from companies working for the city hall."

2015-09-15,Canadian banks helping clients bend rules to move money out of China

2015-09-15,Poland & Sweden agree to intensify military cooperation

2015-09-15,North Korea 'restarts nuclear operations' | BBC

2015-09-15,Teen Arrested for Planning Alleged ISIS-Inspired Attack on Pope

2015-09-15,'Syria Is Emptying'

2015-09-14,Malcom Turnbull becomes Prime Minister of Australia after Tony Abbott rejected by Liberal party

2015-09-14,El Nino set to be strongest ever. The most powerful weather pattern of its type in the past 65 years will have huge impacts on weather around the globe.

2015-09-14,12 Mexican Tourists and Egyptians Killed After Security Forces Misidentify Them As Terrorists

2015-09-14,Police drop VIP Westminster paedophile ring murder probe over lack of evidence

2015-09-14,"A Chinese woman suspected of stealing a $300,000 (195,000) diamond in the Thai capital, Bangkok, has had the jewel surgically removed from her intestines."

2015-09-14,"Iran's President Rouhani sends message for Jewish new year - ""May our shared Abrahamic roots deepen respect and bring peace and mutual understanding. L'Shanah Tovah."""

2015-09-14,Australia- A former teacher from one of Australia's most distinguished families of judges has avoided jail after being convicted of filming up the skirts of his students and possessing (over 9000 images) child abuse material.

2015-09-14,"In a remarkable technical feat, researchers have sequenced DNA from fossils in Spain that are about 300,000 to 400,000 years old and have found an ancestoror close relativeof Neanderthals."

2015-09-14,"Don't return, a Sydney uni student is told after his father 'disappears' in China"

2015-09-14,Bolivia proposes commercialization of the coca leaf

2015-09-14,Japan's Mount Aso volcano erupts

2015-09-14,Historian understood to have found first use of word f*** in 1310 English court case

2015-09-14,Two suspected terrorists linked to Boko Haram are charged with horrendous acid attack on holidaying British teenagers

2015-09-14,El Nino could make 2015 'the hottest year on record'... and 2016 will be even hotter. The findings will demolish claims by climate sceptics that the slowdown in the rate of global warming over the past 17 years proves that scientists concerns are exaggerated.

2015-09-14,"Goldman Sachs report: Oil headed to $20 a barrel. Economist says it's unlikely because ""There are too many big financial players that have been buying up oil they would just start buying up the companies and shutting down production themselves"""

2015-09-14,Iran's President Wishes Jews a Happy Rosh Hashana

2015-09-14,Egypt Begins Digging Moat To Protect Itself from Hamas: The moat is intended to prevent the Muslim-Brotherhood affiliate in Gaza from digging smuggling tunnels into Egypt.

2015-09-14,Vivienne Westwood drives tank to David Cameron's house in anti-fracking protest.

2015-09-14,"Three critically endangered Javan rhinoceros calves where spotted on camera in an Indonesian national park, much to the excitement of staff and rhino lovers."

2015-09-14,Obama Administration Accused of Ignoring Geneva Conventions in Refusal to Release 74-Pound Guantanamo Detainee

2015-09-14,"Taliban storms Afghan jail with suicide bombers, releases over 350 prisoners"

2015-09-14,NASA Launching 4K TV Channel

2015-09-14,The Egyptian army announced it has killed 64 alleged Islamic State (IS) militants in North Sinai Saturday

2008-06-08,b'Police release chilling images of man left for dead by hit-and-run driver'

2008-06-08,"b""'Iceland opens Europe's largest national park. The 15.000 square kilometer (5792 square mile) Vatnajkull National Park. """

2008-06-08,b'Citizens fighting Blackwater Potrero. Was the location chosen so they can train cheap Mexican soldiers or so they can perform extraordinary rendition from a partner facility in Mexico'

2008-06-08,"b""'S. Korean protesters, police clash in beef rallies--don't want US beef..."""

2008-06-08,"b""'Oil reserves 'will last decades' - a BBC Scotland investigation has been told"""

2008-06-08,b'Cameras designed to detect terrorist facial expressions'

2008-06-08,b'Israeli peace activists protest 41 years of occupation'

2008-06-08,"b""A 5.1 earthquake hits China's Southern Qinghai province."""

2008-06-08,"b'Man goes berzerk in Akihabara and stabs everyone nearby: 6 dead, 12 injured '"

2008-06-08,"b'Threat of world AIDS pandemic among heterosexuals is over, report admits'"

2008-06-08,b'Angst in Ankara: Turkey Steers into a Dangerous Identity Crisis'

2008-06-08,"b""UK: Identity cards 'could be used to spy on people' and a new children's database may be used to identify likely future criminals. Has covert surveillance gone too far?"""

2008-06-08,"b'Marriage, they said, was reduced to the status of a commercial transaction in which women could be discarded by husbands claiming to have discovered hidden defects in them.'"

AMAZON STOCKS: (From Google Finance)

Date,Open,High,Low,Close,Adj Close,Volume

2016-05-27,715,716.599976,711.099976,712.23999,712.23999,2249200

2016-05-26,708.330017,715,707.289978,714.909973,714.909973,2446700

2016-05-25,708,710.859985,705.52002,708.349976,708.349976,3267700

2016-05-24,698.01001,707.5,698,704.200012,704.200012,3033800

2016-05-23,704.25,706,696.419983,696.75,696.75,2595100

2016-05-20,701.049988,707.23999,700,702.799988,702.799988,2916200

2016-05-19,691.880005,699.400024,689.559998,698.52002,698.52002,3025600

2016-05-18,689.559998,702.539978,688.76001,697.450012,697.450012,4283200

2016-05-17,709.900024,714.469971,693.909973,695.27002,695.27002,5121400

2016-05-16,710.130005,713.25,700.280029,710.659973,710.659973,5432900

2016-05-13,714.640015,719.25,706.51001,709.919983,709.919983,4763400

2016-05-12,717.380005,722.450012,711.51001,717.929993,717.929993,5048200

2016-05-11,705.789978,719,701.650024,713.22998,713.22998,7338200

2016-05-10,694,704.549988,693.5,703.070007,703.070007,6105600

2016-05-09,673.950012,686.97998,671.409973,679.75,679.75,3982200

2016-05-06,656.049988,676.950012,656.01001,673.950012,673.950012,4365300

2016-05-05,673.309998,676.48999,656,659.090027,659.090027,4884100

2016-05-04,662.590027,674,662.140015,670.900024,670.900024,4635500

2016-05-03,677.359985,680.299988,670.429993,671.320007,671.320007,4923400

2016-05-02,663.919983,685.5,662.030029,683.849976,683.849976,6578500

2016-04-29,666,669.97998,654,659.590027,659.590027,10310700

2016-04-28,615.539978,626.799988,599.200012,602,602,7872600

2016-04-27,611.799988,615.950012,601.280029,606.570007,606.570007,4068800

2016-04-26,626.169983,626.75,614.880005,616.880005,616.880005,2521400

2016-04-25,616.609985,626.97998,616.25,626.200012,626.200012,2682900

2016-02-10,491.76001,504.660004,486,490.480011,490.480011,6786200

2015-04-27,443.859985,446.98999,437.410004,438.559998,438.559998,5430900

2015-04-24,439,452.649994,439,445.100006,445.100006,17176900

2015-04-23,390.209991,391.880005,386.149994,389.98999,389.98999,7980000

2015-04-22,391.910004,394.279999,388,389.799988,389.799988,3474700

2015-04-21,391.309998,394.600006,386.799988,391.179993,391.179993,4643500

2015-04-20,378.549988,391.940002,377,389.51001,389.51001,5016100

2015-04-17,382.630005,383.559998,374.399994,375.559998,375.559998,3839700

2015-04-16,383.690002,387.450012,383.549988,386.040009,386.040009,2080400

2015-04-15,384.649994,385.779999,381.640015,383.450012,383.450012,1933200

2015-04-14,383.51001,387.809998,381.209991,385.109985,385.109985,2583600

2015-04-13,383.529999,385.279999,380.140015,382.359985,382.359985,1894500

2015-04-10,384.309998,387.119995,381.320007,382.649994,382.649994,2573500

2015-04-09,380.660004,384.420013,378.799988,383.540009,383.540009,2392300

2015-04-08,374.660004,381.579987,374.649994,381.200012,381.200012,2636400

2015-04-07,376.149994,379.309998,374.029999,374.410004,374.410004,1954900

2015-04-06,370.100006,380.200012,369.359985,377.040009,377.040009,3050700

2015-04-02,370.5,373.279999,369,372.25,372.25,1875300

2015-04-01,372.100006,373.160004,368.339996,370.26001,370.26001,2458100

2015-03-31,373.23999,377.700012,371.51001,372.100006,372.100006,2506100

2015-03-30,371.869995,376.119995,371.549988,374.589996,374.589996,1820900

2015-03-27,367.109985,373.170013,366.570007,370.559998,370.559998,2609800

2015-03-26,369.589996,371.399994,365.649994,367.350006,367.350006,2930000

2015-03-25,375.170013,380.5,370.290009,370.959991,370.959991,3429500

2015-03-24,373.98999,375.23999,372.269989,374.089996,374.089996,2228200

2015-03-23,378.070007,381.769989,374.940002,375.109985,375.109985,2239300

2015-02-13,378.410004,383,377.01001,381.829987,381.829987,3475100

2014-05-06,309.529999,309.809998,297.040009,297.380005,297.380005,4682300

2014-05-05,306.369995,310.230011,305,310.049988,310.049988,2519900

2014-05-02,310.420013,313.290009,304.309998,308.01001,308.01001,3995100

2014-05-01,304.130005,310.480011,304,307.890015,307.890015,4328600

2014-04-30,298.100006,304.559998,298.100006,304.130005,304.130005,4088600

2014-04-29,296.440002,301.839996,290.450012,300.380005,300.380005,6509300

2014-04-28,304,304.390015,288,296.579987,296.579987,14479800

2014-04-25,316.25,316.48999,302.709991,303.829987,303.829987,16180200

2014-04-24,329.670013,337.399994,322.950012,337.149994,337.149994,9293700

2014-04-23,333.059998,333.130005,323.390015,324.579987,324.579987,3604600

2014-04-22,332,337.5,328.940002,329.320007,329.320007,3711600

2014-04-21,323.970001,331.149994,322.309998,330.869995,330.869995,2999400

2014-04-17,319.76001,328.660004,319.76001,324.910004,324.910004,4299200

2014-04-16,321.170013,324,314.709991,323.679993,323.679993,4284900

2014-04-15,316.700012,318.279999,305.5,316.079987,316.079987,5398600

2014-04-14,317.670013,320.480011,311.279999,315.910004,315.910004,4293500

2014-04-11,314,316.5,309.5,311.730011,311.730011,7287500

2014-04-10,330.600006,331,316.5,317.109985,317.109985,6126700

2014-04-09,328.470001,332.179993,322.5,331.809998,331.809998,5056600

2014-04-08,321.880005,328,318.440002,327.070007,327.070007,6577600

2014-04-07,320.98999,324.940002,313.130005,317.76001,317.76001,7077400

2014-04-04,335.149994,335.440002,315.609985,323,323,12534600

2014-04-03,341.820007,342.5,328.459991,333.619995,333.619995,6399300

2014-04-02,345.98999,348.299988,340.380005,341.959991,341.959991,4475500

2014-04-01,338.089996,344.429993,338,342.98999,342.98999,3600100

2014-03-31,342.399994,346.290009,334.059998,336.369995,336.369995,4297500

2014-03-28,340.049988,347,336.079987,338.290009,338.290009,3986800

2014-03-27,343.149994,344,330.880005,338.470001,338.470001,5766400

2014-03-26,357.130005,357.600006,343.399994,343.410004,343.410004,4120700

2014-03-25,354.029999,358.970001,348.839996,354.709991,354.709991,4445700

2014-03-24,360.089996,361.5,348.600006,351.850006,351.850006,4873500

2014-03-21,371,372.839996,358.399994,360.619995,360.619995,5414100

2014-03-20,370.640015,373,366.220001,368.970001,368.970001,2558500

2014-03-19,378.769989,379,369.420013,373.230011,373.230011,2646700

2014-03-18,377.320007,379,375,378.769989,378.769989,2483500

2014-03-17,375.720001,378.850006,374.880005,375.040009,375.040009,2303000

2014-03-14,372.799988,378.570007,371.549988,373.73999,373.73999,4402200

2014-03-13,376.619995,383.109985,368.079987,371.51001,371.51001,6829000

2014-03-12,366.399994,371.160004,363.609985,370.640015,370.640015,2216600

2014-03-11,370.98999,372.799988,367.279999,368.820007,368.820007,2246100

2014-03-10,372.690002,372.730011,367,370.529999,370.529999,2105800

2014-03-07,374.579987,374.98999,369.529999,372.059998,372.059998,2279800

2014-03-06,374.049988,375.329987,368.899994,372.160004,372.160004,2926600

2014-03-05,364.130005,372.730011,363.899994,372.369995,372.369995,3848300

2014-03-04,363.899994,365.679993,362.459991,363.899994,363.899994,2704400

2014-03-03,358.73999,360.959991,354.480011,359.779999,359.779999,2798300

2014-02-28,360.600006,365.869995,357.079987,362.100006,362.100006,3882000

2014-02-27,357.220001,360.589996,355.5,360.130005,360.130005,3104900

2014-02-26,359.859985,364.75,357.170013,359.799988,359.799988,3622100

2014-02-25,353,361.079987,351.579987,358.320007,358.320007,3736400

2014-02-24,345.190002,353,343.290009,351.779999,351.779999,3644700

2014-02-21,352.440002,354.140015,346.75,346.76001,346.76001,4210000

2014-02-20,348.799988,350.459991,344.380005,349.799988,349.799988,3492800

2014-02-19,352.640015,354.540009,346.100006,347.380005,347.380005,4168100

2014-02-18,355.279999,355.730011,349.450012,353.649994,353.649994,4998000

2014-

2012-05-02,227.820007,231.440002,227.399994,230.25,230.25,4593400

2012-05-01,229.399994,232.970001,228.399994,230.039993,230.039993,6754900

2012-04-30,223.949997,233.839996,223.050003,231.899994,231.899994,9756900

2012-04-27,224.830002,228.690002,220.220001,226.850006,226.850006,22116900

2012-04-26,193.570007,196.360001,193.020004,195.990005,195.990005,10234000

2012-04-25,191.669998,194.800003,191.600006,194.419998,194.419998,3955100

2012-04-24,188.679993,190.699997,186.509995,190.330002,190.330002,3376300

2012-04-23,188.990005,188.990005,185.509995,188.240005,188.240005,3481000

2012-04-20,192.339996,193.479996,189.800003,189.979996,189.979996,3243600

2012-04-19,192.929993,194.550003,189.75,191.100006,191.100006,4002400

2012-04-18,188.820007,193.449997,188.740005,191.070007,191.070007,4001900

2012-04-17,187.210007,190.039993,186.869995,188.389999,188.389999,2829200

2012-04-16,189.009995,189.470001,183.649994,185.5,185.5,4044300

2012-04-13,189.899994,189.940002,186.259995,188.460007,188.460007,3431800

2012-04-12,188.059998,192.259995,185.610001,190.690002,190.690002,4027900

2012-04-11,189.630005,191.970001,186.789993,187.970001,187.970001,4337800

2012-04-10,192.75,193.520004,186.570007,186.979996,186.979996,4455000

2012-04-09,192.020004,194.199997,190.5,191.869995,191.869995,3135900

2012-04-05,193.550003,196.029999,193.550003,194.389999,194.389999,3217500

2012-04-04,196.949997,197.679993,192.360001,193.990005,193.990005,5456900

2012-04-03,198.240005,202.389999,197.5,199.660004,199.660004,5000700

2012-04-02,198.020004,199.899994,197,198.050003,198.050003,6430300

2012-03-30,205.020004,206.850006,201.869995,202.509995,202.509995,4438100

2012-03-29,201.279999,205.309998,200.630005,204.610001,204.610001,5711200

,191.210007,193.949997,188.399994,193.029999,193.029999,5158800

2011-12-08,193.570007,195.889999,190.080002,190.479996,190.479996,4361100

2011-12-07,191.029999,196.710007,189.119995,195.320007,195.320007,6427300

2011-12-06,195.979996,198.320007,190.110001,191.990005,191.990005,5202000

2011-12-05,198.860001,199,193.669998,196.240005,196.240005,5922100

2011-12-02,197.070007,199.660004,195.179993,196.029999,196.029999,7526200

2011-12-01,191.850006,198.070007,191.589996,197.130005,197.130005,7327700

2011-11-30,194.759995,195.300003,188.75,192.289993,192.289993,7717000

2011-11-29,194.779999,195.5,187.300003,188.389999,188.389999,6575100

2011-11-28,191.649994,194.619995,190.539993,194.149994,194.149994,7207300

2011-11-25,190.410004,190.830002,181.509995,182.399994,182.399994,4972000

2011-11-23,193.059998,194.600006,187.889999,188.990005,188.990005,8011300

2011-11-22,186.949997,194.039993,183.580002,192.339996,192.339996,9915600

2011-11-21,193.289993,193.360001,185.050003,189.25,189.25,11321200

2011-11-18,205.330002,205.339996,197.110001,197.139999,197.139999,8437500

2011-11-17,212.509995,212.899994,202.100006,204.520004,204.520004,7983100

2011-11-16,216.270004,216.970001,211.229996,211.990005,211.990005,5509400

2011-11-15,218,220.330002,214.259995,217.830002,217.830002,5739000

2011-11-14,215.649994,222.350006,214.25,218.929993,218.929993,6522200

2011-11-11,212.520004,217.880005,210.309998,217.389999,217.389999,5163100

2011-11-10,213.5,214.059998,208.100006,210.789993,210.789993,5044600

2011-11-09,214.949997,215.699997,210.600006,211.220001,211.220001,4680600

2011-11-08,219.199997,219.350006,215.210007,217.990005,217.990005,3914500

2011-11-07,216.839996,220.199997,214,217,217,3860000

2011-11-04,217.649994,218.229996,214.330002,216.479996,216.479996,4065800

2011-11-03,216.300003,218.5,213.020004,218.289993,218.289993,5315000

2011-11-02,215.550003,216.789993,212.720001,215.619995,215.619995,6122000

2011-11-01,208.110001,216.210007,207.429993,212.100006,212.100006,8511800

2011-10-31,215.789993,218.889999,213.039993,213.509995,213.509995,7343300

2011-10-28,206.529999,218.399994,205.75,217.320007,217.320007,9880400

2011-10-27,204.259995,208.600006,201.100006,206.779999,206.779999,10774300

2011-10-26,203.690002,207.580002,196.509995,198.399994,198.399994,24134200

2011-10-25,238.589996,239.009995,225.889999,227.149994,227.149994,14012600

2011-10-24,236.020004,240.470001,234,237.610001,237.610001,4975800

2011-10-21,236.910004,237,230.600006,234.779999,234.779999,4572500

2011-10-20,232.130005,234.740005,229.800003,233.610001,233.610001,4524900

2011-10-19,240.669998,243.330002,229.25,231.529999,231.529999,6715100

2011-10-18,242.309998,244.610001,236.619995,243.880005,243.880005,4609700

2011-10-17,244.289993,246.710007,240.669998,242.330002,242.330002,4779000

2011-10-14,240.869995,246.710007,240.179993,246.710007,246.710007,5923700

2011-10-13,237,239.679993,235.229996,236.149994,236.149994,4833500

2011-10-12,236.639999,241.839996,234.330002,236.809998,236.809998,6510800

2011-10-11,230.600006,236.75,229,235.479996,235.479996,5003700

2011-10-10,226.229996,232.800003,224.100006,231.320007,231.320007,5143100

2011-10-07,222.479996,227.899994,218.410004,224.740005,224.740005,6784300

2011-10-06,220.279999,223.619995,217.550003,221.509995,221.509995,6849300

2011-10-05,212.529999,220.169998,208.479996,219.5,219.5,6508200

2011-10-04,209.619995,215,200.429993,212.5,212.5,8711600

2011-10-03,217.009995,221.600006,211.389999,211.979996,211.979996,6624400

2011-09-30,218.190002,223,215.210007,216.229996,216.229996,6550300

2011-09-29,234.169998,234.300003,216.289993,222.440002,222.440002,9378500

2011-09-28,226.350006,235.809998,225.600006,229.710007,229.710007,14436900

2011-09-27,234.220001,234.75,222.399994,224.210007,224.210007,7837500

2010-02-25,118.169998,118.339996,115.849998,118.199997,118.199997,9533400

2010-02-24,117.959999,119.800003,117.150002,119.720001,119.720001,7389900

2010-02-23,118.010002,119.25,116.510002,117.239998,117.239998,7068200

2010-02-22,117.370003,118.970001,116.18,118.010002,118.010002,6807300

2010-02-19,117.910004,119.089996,117,117.519997,117.519997,7115600

2010-02-18,115.839996,118.510002,114.82,118.080002,118.080002,9800100

2010-02-17,117.07,117.129997,115.550003,116.309998,116.309998,8944800

2010-02-16,120.059998,120.5,117.18,117.529999,117.529999,8932700

2010-02-12,118.989998,119.940002,117.5,119.660004,119.660004,8073500

2010-02-11,117.209999,120.419998,116.5,120.089996,120.089996,8343500

2010-02-10,118,118.610001,116,117.360001,117.360001,6233200

2010-02-09,118.199997,119.089996,117,118.029999,118.029999,9223000

2010-02-08,119.379997,121,116.559998,116.830002,116.830002,9890200

2010-02-05,115.879997,117.650002,114.099998,117.389999,117.389999,11024800

2010-02-04,118.639999,120.330002,115.739998,115.940002,115.940002,12784000

2010-02-03,117.120003,119.610001,116.559998,119.099998,119.099998,12405900

2010-02-02,118.790001,118.980003,114.400002,118.120003,118.120003,23079700

2010-02-01,123.18,124.860001,113.82,118.870003,118.870003,37774400

2010-01-29,129.770004,131.850006,124.139999,125.410004,125.410004,29471300

2010-01-28,124.43,127.199997,122.800003,126.029999,126.029999,27293100

2010-01-27,121.029999,123.330002,118.800003,122.75,122.75,14765300

2010-01-26,120.559998,122.980003,119.059998,119.480003,119.480003,9559000

2010-01-25,122.099998,122.279999,118.120003,120.309998,120.309998,12023900

2010-01-22,125.599998,127.669998,120.760002,121.43,121.43,11568900

2010-01-21,127.260002,128.149994,125,126.620003,126.620003,9970600

2010-01-20,127.129997,129.199997,125.080002,125.779999,125.779999,9074700

2010-01-19,126.309998,128,124.330002,127.610001,127.610001,8892600

2010-01-15,129.179993,129.649994,127.059998,127.139999,127.139999,15376500

2010-01-14,129.139999,130.380005,126.400002,127.349998,127.349998,9774900

2010-01-13,127.900002,129.710007,125.75,129.110001,129.110001,10723200

2010-01-12,128.990005,129.820007,126.550003,127.349998,127.349998,9096300

2010-01-11,132.619995,132.800003,129.210007,130.309998,130.309998,8779400

2010-01-08,130.559998,133.679993,129.029999,133.520004,133.520004,9830500

2010-01-07,132.009995,132.320007,128.800003,130,130,11030200

2010-01-06,134.600006,134.729996,131.649994,132.25,132.25,7178800

2010-01-05,133.429993,135.479996,131.809998,134.690002,134.690002,8851900

2010-01-04,136.25,136.610001,133.139999,133.899994,133.899994,7599900

2009-12-31,137.089996,137.279999,134.520004,134.520004,134.520004,4523000

2009-12-30,138.399994,138.399994,135.279999,136.490005,136.490005,6913200

2009-12-29,141.289993,142.580002,138.550003,139.410004,139.410004,8400600

2009-12-28,139.75,141.979996,138.529999,139.309998,139.309998,8763900

2009-12-24,139.199997,139.699997,137.539993,138.470001,138.470001,5128800

2009-12-23,134.800003,139.050003,134.350006,138.940002,138.940002,9546100

2009-12-22,133.759995,135.990005,132.649994,133.75,133.75,8257500

2009-12-21,130.479996,133.199997,130.190002,132.789993,132.789993,9473600

2009-12-18,127.910004,128.789993,125.650002,128.479996,128.479996,9605400

2009-12-17,129.360001,130.080002,126.900002,126.910004,126.910004,8476500

2009-12-16,130.929993,131.449997,127.650002,128.360001,128.360001,10261300

2009-12-15,130.759995,132.460007,129.589996,130.229996,130.229996,7428800

2009-12-14,132.5,132.610001,129.350006,131.380005,131.380005,10022900

2009-12-11,136.070007,136.289993,133.199997,134.149994,134.149994,8046700

2009-12-10,132.410004,136.190002,132.399994,135.380005,135.380005,11343600

2009-12-09,134.600006,134.710007,129.820007,131.309998,131.309998,12632900

2009-12-08,134.300003,136.080002,132.869995,134.110001,134.110001,8002800

2009-12-07,138,139,133.839996,134.210007,134.210007,7837000

2009-12-04,143.419998,143.449997,135.110001,137.580002,137.580002,14827400

2009-12-03,143.619995,145.910004,140.770004,141.169998,141.169998,16523500

2009-12-02,139.149994,142.669998,138.960007,142.25,142.25,11798000

2009-12-01,136.940002,139.350006,135.75,138.5,138.5,9657600

2009-11-30,132.190002,136.080002,132.160004,135.910004,135.910004,10119500

2009-11-27,130.300003,133,129.880005,131.740005,131.740005,4422600

2009-11-25,133.309998,134.199997,132.399994,134.029999,134.029999,5071200

2009-11-24,133.570007,134.330002,132.220001,132.940002,132.940002,7319700

2009-11-23,131.050003,133,131,133,133,6878100

2009-11-20,127.760002,129.990005,127.410004,129.660004,129.660004,6652600

6.2 CODE :

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import numpy as np

import tensorflow as tf

import re

from nltk.corpus import stopwords

from sklearn.model_selection import train_test_split

from sklearn.metrics import median_absolute_error as mae

from sklearn.metrics import mean_squared_error as mse

from sklearn.metrics import accuracy_score as acc
```

```
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras import initializers
from keras.layers import Dropout, Activation, Embedding, Convolution1D, MaxPooling1D, Input,
Dense, BatchNormalization, Flatten, Reshape, Concatenate
from keras.layers.recurrent import LSTM, GRU
from keras.callbacks import Callback, ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
from keras.models import Model
from keras.optimizers import Adam, SGD, RMSprop
from keras import regularizers
```

 # # Data Description

```
ST = pd.read_csv("AMZN.csv")
HL = pd.read_csv("Headlines.csv")
ST.head(8)
```

	Date	Open	High	Low	Close	Adj Close	Volume
0	2016-07-01	717.320007	728.000000	716.539978	725.679993	725.679993	2920400
1	2016-06-30	717.200012	719.369995	712.539978	715.619995	715.619995	2855100
2	2016-06-29	715.750000	719.500000	713.539978	715.599976	715.599976	3070100
3	2016-06-28	700.000000	708.000000	698.169983	707.950012	707.950012	4037000
4	2016-06-27	692.010010	696.820007	682.119995	691.359985	691.359985	5568000
5	2016-06-24	693.000000	712.530029	692.200012	698.960022	698.960022	7632500
6	2016-06-23	715.500000	722.119995	712.500000	722.080017	722.080017	2825000
7	2016-06-22	716.580017	717.000000	707.570007	710.599976	710.599976	2260500
8	2016-06-21	715.719971	718.400024	712.719971	715.820007	715.820007	2137500

ST.tail(8)

	Date	Open	High	Low	Close	Adj Close	Volume
1980	2008-08-20	82.000000	83.250000	81.199997	82.129997	82.129997	5950600
1981	2008-08-19	83.089996	83.510002	81.059998	81.290001	81.290001	6630400
1982	2008-08-18	86.089996	86.279999	83.040001	83.110001	83.110001	6547400
1983	2008-08-15	88.279999	89.529999	86.260002	86.400002	86.400002	6871600
1984	2008-08-14	85.709999	88.750000	85.220001	88.029999	88.029999	6901700
1985	2008-08-13	86.279999	88.250000	84.540001	86.690002	86.690002	7208800
1986	2008-08-12	87.320000	88.480003	86.099998	87.250000	87.250000	8026500
1987	2008-08-11	80.180000	91.750000	79.779999	88.089996	88.089996	25070200
1988	2008-08-08	76.779999	81.209999	76.290001	80.510002	80.510002	9162700
1989	2008-08-07	77.010002	78.050003	76.000000	76.949997	76.949997	5444800

HL.head(10)

HL.tail(10)

Output:

```
HL.head(10)
```

	Date	News
0	2016-07-01	A 117-year-old woman in Mexico City finally re...
1	2016-07-01	IMF chief backs Athens as permanent Olympic host
2	2016-07-01	The president of France says if Brexit won, so...
3	2016-07-01	British Man Who Must Give Police 24 Hours' Not...
4	2016-07-01	100+ Nobel laureates urge Greenpeace to stop o...
5	2016-07-01	Brazil: Huge spike in number of police killing...
6	2016-07-01	Austria's highest court annuls presidential el...
7	2016-07-01	Facebook wins privacy case, can track any Belg...
8	2016-07-01	Switzerland denies Muslim girls citizenship af...
9	2016-07-01	China kills millions of innocent meditators fo...

```
HL.tail(10)
```

	Date	News
73598	2008-06-08	b"S. Korean protesters, police clash in beef r...
73599	2008-06-08	b"Oil reserves 'will last decades' - a BBC Sco...
73600	2008-06-08	b'Cameras designed to detect terrorist facial ...
73601	2008-06-08	b'Israeli peace activists protest 41 years of ...
73602	2008-06-08	b"A 5.1 earthquake hits China's Southern Qingh...
73603	2008-06-08	b'Man goes berzerk in Akihabara and stabs ever...
73604	2008-06-08	b'Threat of world AIDS pandemic among heterose...
73605	2008-06-08	b'Angst in Ankara: Turkey Steers into a Danger...
73606	2008-06-08	b"UK: Identity cards 'could be used to spy on ...
73607	2008-06-08	b'Marriage, they said, was reduced to the stat...

HL.isnull().sum()

```
HL.isnull().sum()

Date    0
News    0
dtype: int64
```

ST.isnull().sum()

```
ST.isnull().sum()

Date          0
Open          0
High          0
Low           0
Close         0
Adj Close     0
Volume        0
dtype: int64
```

print(ST.shape)

print(HL.shape)

output:

(1990, 7)

(73608, 2)

```
print(len(set(ST.Date)))

print(len(set(HL.Date)))
```

output:

1990

2943

```
HL = HL[HL.Date.isin(ST.Date)]
```

```
HL
```

	Date	News
0	2016-07-01	A 117-year-old woman in Mexico City finally re...
1	2016-07-01	IMF chief backs Athens as permanent Olympic host
2	2016-07-01	The president of France says if Brexit won, so...
3	2016-07-01	British Man Who Must Give Police 24 Hours' Not...
4	2016-07-01	100+ Nobel laureates urge Greenpeace to stop o...
...	...	...
72103	2008-08-07	b'Pininfarina, Italian Car Designer, Dies in T...
72104	2008-08-07	b'Venezuelas Chavez Pushes for New World Finan...
72105	2008-08-07	b'Stalinism Was Just as Bad as Nazism'
72106	2008-08-07	b'Anti-War Website Operator Threatened By Arme...
72107	2008-08-07	b"Police fear Maddie 'stolen to order'"

[49743 rows x 2 columns]

```
print(len(set(ST.Date)))

print(len(set(HL.Date)))
```

```
1990
1990
```

ST = ST.set_index('Date').diff(periods=1)

```
In [19]: ST
```

Out[19]:

Date	Open	High	Low	Close	Adj Close	Volume	Date
2016-07-01	NaN	NaN	NaN	NaN	NaN	NaN	2016-07-01
2016-06-30	-0.119995	-8.630005	-4.000000	-10.059998	-10.059998	-65300.0	2016-06-30
2016-06-29	-1.450012	0.130005	1.000000	-0.020019	-0.020019	215000.0	2016-06-29
2016-06-28	-15.750000	-11.500000	-15.369995	-7.649964	-7.649964	966900.0	2016-06-28
2016-06-27	-7.989990	-11.179993	-16.049988	-16.590027	-16.590027	1531000.0	2016-06-27
...	...	...	...	...	...	...	...
2008-08-13	0.570000	-0.500000	-0.680000	-1.339997	-1.339997	307100.0	2008-08-13
2008-08-12	1.040001	0.230003	1.559997	0.559998	0.559998	817700.0	2008-08-12
2008-08-11	-7.140000	3.269997	-6.319999	0.839996	0.839996	17043700.0	2008-08-11
2008-08-08	-3.400001	-10.540001	-3.489998	-7.579994	-7.579994	-15907500.0	2008-08-08
2008-08-07	0.230003	-3.159996	-0.290001	-3.560005	-3.560005	-3717900.0	2008-08-07

1990 rows × 7 columns

ST = ST.reset_index(drop=True)

```
: ST
```

:

	Open	High	Low	Close	Adj Close	Volume	Date
0	NaN	NaN	NaN	NaN	NaN	NaN	2016-07-01
1	-0.119995	-8.630005	-4.000000	-10.059998	-10.059998	-65300.0	2016-06-30
2	-1.450012	0.130005	1.000000	-0.020019	-0.020019	215000.0	2016-06-29
3	-15.750000	-11.500000	-15.369995	-7.649964	-7.649964	966900.0	2016-06-28
4	-7.989990	-11.179993	-16.049988	-16.590027	-16.590027	1531000.0	2016-06-27
...	...	...	...	...	...	...	...
1985	0.570000	-0.500000	-0.680000	-1.339997	-1.339997	307100.0	2008-08-13
1986	1.040001	0.230003	1.559997	0.559998	0.559998	817700.0	2008-08-12
1987	-7.140000	3.269997	-6.319999	0.839996	0.839996	17043700.0	2008-08-11
1988	-3.400001	-10.540001	-3.489998	-7.579994	-7.579994	-15907500.0	2008-08-08
1989	0.230003	-3.159996	-0.290001	-3.560005	-3.560005	-3717900.0	2008-08-07

1990 rows × 7 columns

 # # Data Dropping

Remove unneeded features

ST =ST.drop(['High','Low','Close','Volume','Adj Close'], 1)

```
ST
```

```
            Open         Date
0            NaN   2016-07-01
1      -0.119995   2016-06-30
2      -1.450012   2016-06-29
3     -15.750000   2016-06-28
4      -7.989990   2016-06-27
...          ...          ...
1985    0.570000   2008-08-13
1986    1.040001   2008-08-12
1987   -7.140000   2008-08-11
1988   -3.400001   2008-08-08
1989    0.230003   2008-08-07

[1990 rows x 2 columns]
```

ST = ST[ST.Open.notnull()]

```
ST
```

```
            Open         Date
1      -0.119995   2016-06-30
2      -1.450012   2016-06-29
3     -15.750000   2016-06-28
4      -7.989990   2016-06-27
5       0.989990   2016-06-24
...          ...          ...
1985    0.570000   2008-08-13
1986    1.040001   2008-08-12
1987   -7.140000   2008-08-11
1988   -3.400001   2008-08-08
1989    0.230003   2008-08-07

[1989 rows x 2 columns]
```

ST.isnull().sum()

```
ST.isnull().sum()
```

```
Open    0
Date    0
dtype: int64
```

 # List of headlines and corresponding stock prices

```
price = []

headlines = []

for i in ST.iterrows():
    daily_headlines = []
    date = i[1]['Date']
    price.append(i[1]['Open'])
    for j in HL[HL.Date==date].iterrows():
        daily_headlines.append(j[1]['News'])
    headlines.append(daily_headlines)
```

```
price

[-0.11999500000001717,
 -1.4500120000000152,
 -15.75,
 -7.989990000000034,
 0.9899900000000343,
 22.5,
 1.080016999999998,
 -0.8600460000000112,
 -2.219970999999987,
 4.690001999999936,
 -6.140013999999951,
```

```
print(len(price))

print(len(headlines))
```

```
In [31]: print(len(price))
         print(len(headlines))

         1989
         1989
```

```
for i in range(0,10):
    print(headlines[i])
    print('\n')
```

#max and min number of headlines for each day

```
In [34]: print(max(len(i) for i in headlines))
         print(min(len(i) for i in headlines))

         25
         22
```

 # A list of contractions from http://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python

```
contractions = {
"ain't": "am not",
"aren't": "are not",
"can't": "cannot",
"can't've": "cannot have",
"'cause": "because",
```

"could've": "could have",

"couldn't": "could not",

"couldn't've": "could not have",

"didn't": "did not",

"doesn't": "does not",

"don't": "do not",

"hadn't": "had not",

"hadn't've": "had not have",

"hasn't": "has not",

"haven't": "have not",

"he'd": "he would",

"he'd've": "he would have",

"he'll": "he will",

"he's": "he is",

"how'd": "how did",

"how'll": "how will",

"how's": "how is",

"i'd": "i would",

"i'll": "i will",

"i'm": "i am",

"i've": "i have",

"isn't": "is not",

"it'd": "it would",

"it'll": "it will",

"it's": "it is",

"let's": "let us",

"ma'am": "madam",

"mayn't": "may not",

"might've": "might have",

"mightn't": "might not",

"must've": "must have",

"mustn't": "must not",

"needn't": "need not",

"oughtn't": "ought not",

"shan't": "shall not",

"sha'n't": "shall not",

"she'd": "she would",

"she'll": "she will",

"she's": "she is",

"should've": "should have",

"shouldn't": "should not",

"that'd": "that would",

"that's": "that is",

"there'd": "there had",

"there's": "there is",

"they'd": "they would",

"they'll": "they will",

"they're": "they are",

"they've": "they have",

"wasn't": "was not",

"we'd": "we would",

"we'll": "we will",

"we're": "we are",

"we've": "we have",

"weren't": "were not",

"what'll": "what will",

"what're": "what are",

"what's": "what is",

"what've": "what have",

"where'd": "where did",

"where's": "where is",

"who'll": "who will",

"who's": "who is",

"won't": "will not",

"wouldn't": "would not",

"you'd": "you would",

"you'll": "you will",

"you're": "you are"

}

def clean(text, remove_stopwords = True):

 # Convert words to lower case

text = text.lower()

```
# Replace contractions with actual words
if True:
    text = text.split()
    new_text = []
    for word in text:
        if word in contractions:
            new_text.append(contractions[word])
        else:
            new_text.append(word)
    text = " ".join(new_text)

# remove unwanted characters
text = re.sub(r'&amp;amp;', '', text)
text = re.sub(r'0,0', '00', text)
text = re.sub(r'[_"\-;%()|.,+&=*%.,!?:#@\[\]]', ' ', text)
text = re.sub(r'\'', ' ', text)
text = re.sub(r'\$', ' $ ', text)
text = re.sub(r'u s ', ' united states ', text)
text = re.sub(r'u n ', ' united nations ', text)
text = re.sub(r'u k ', ' united kingdom ', text)
text = re.sub(r'j k ', ' jk ', text)
text = re.sub(r' s ', ' ', text)
text = re.sub(r' yr ', ' year ', text)
text = re.sub(r' l g b t ', ' lgbt ', text)
```

```
    text = re.sub(r'0km ', '0 km ', text)
    # Optionally, remove stop words
    if remove_stopwords:
        text = text.split()
        stops = set(stopwords.words("english"))
        text = [w for w in text if not w in stops]
        text = " ".join(text)
    return text
clean_headlines = []
for daily_headlines in headlines:
    clean_daily_headlines = []
    for headline in daily_headlines:
        clean_daily_headlines.append(clean(headline))
    clean_headlines.append(clean_daily_headlines)


import re
from nltk.corpus import stopwords
```

#HEADLINES ARE CLEANED AND FREE FROM UNWANTED CHARACTERS

clean_headlines[5]

```
clean_headlines[5]
```

```
['today united kingdom decides whether remain european union leave',
 'e cigarettes banned public medical experts warn ban using e cigarettes public places could damaging may put smokers using e c
igarettes help quit says rosanna connor public health england',
 'report china still harvesting organs prisoners',
 'man opens fire cinema complex germany several people wounded report',
 'erdoan europe dont want us muslim',
 'asian millionaires control wealth north america europe regions',
 'japanese porn industry association apologised promised reform amid allegations women forced perform sex acts film',
 'university students warned classes contains graphic sensitive content including sexual abuse rape transgenderism protect ment
al health australian academics issuing called trigger warnings confronting material classrooms',
 'afghan interpreters betrayed uk us',
 'contagious cancer cells spreading different animals even different species sea according new research raises prospect disease
becoming infectious humans',
 '51 killed china powerful tornado',
 'teacher killings ignite calls revolution mexico police crackdown rural educator protest spurs wide rebuke government privatiz
ation repression',
 'solar plane lands spain three day atlantic crossing',
 'brexit supporters urged take pens polling station amid fears mi5 conspiracy',
 'cities forge world largest alliance curb climate change 7 100 cities 119 countries formed global covenant mayors climate ener
gy network helping exchange information goals developing clean energy organizers said',
 'colombia farc announce full ceasefire last day war',
 'gunmen kill sufi devotional singer amjad sabri pakistan pakistani taliban claim responsibility',
 'india launches 20 satellites single mission',
 'f 16s manufactured soon assembly line india',
 'australia gun laws stopped mass shootings reduced homicides study finds',
 'french cement company syria buys oil isis documents',
 'pope visit armenia irking turkey genocide label',
 'merkel says nato must strengthened',
 'china cracks online comments click bait stories foreign tv content xi reshapes media landscape',
 'prime minister india set get brand new air india one aircraft advanced air force one']
```

 # Find the number of times each word was used and the size of the vocabulary

```
word_counts = {}

for i in clean_headlines:
    for headline in i:
        for word in headline.split():
            if word not in word_counts:
                word_counts[word] = 1
            else:
                word_counts[word] += 1


print("Size of Vocabulary:", len(word_counts))
```

```
print("Size of Vocabulary:", len(word_counts))
```

```
Size of Vocabulary: 35200
```

```
word_counts
```

```
{'jamaica': 23,
 'proposes': 65,
 'marijuana': 220,
 'dispensers': 1,
 'tourists': 97,
 'airports': 33,
 'following': 219,
 'legalisation': 4,
 'kiosks': 2,
 'desks': 1,
 'would': 878,
 'give': 289,
 'people': 1888,
 'license': 21,
 'purchase': 26,
 '2': 733,
 'ounces': 2,
```

 # Load GloVe's embeddings

```
embeddings_index = {}
with open('glove.840B.300d.txt', encoding='utf-8') as f:
    for line in f:
        values = line.split(' ')
        word = values[0]
        embedding = np.asarray(values[1:], dtype='float32')
```

```
    embeddings_index[word] = embedding

print('Word embeddings:', len(embeddings_index))
```

output :

```
Word embeddings: 2196017
```

 # Find the number of words that are missing from GloVe, and are used more than our threshold.

```
missing_words = 0
threshold = 10

for word, count in word_counts.items():
   if count > threshold:
      if word not in embeddings_index:
         missing_words += 1

missing_ratio = round(missing_words/len(word_counts),4)*100
print("Number of words missing from GloVe:", missing_words)
print("Percent of words that are missing from vocabulary: {}%".format(missing_ratio))
```

output:

```
Number of words missing from GloVe: 47
Percent of words that are missing from vocabulary: 0.13%
```

 # Limit the vocab that we will use to words that appear ≥ threshold or are in GloVe

#dictionary to convert words to integers

```
vocab_to_int = {}

value = 0

for word, count in word_counts.items():

    if count >= threshold or word in embeddings_index:

        vocab_to_int[word] = value

        value += 1
```

 # Special tokens that will be added to our vocab

```
codes = ["<UNK>","<PAD>"]
```

 # Add codes to vocab

```
for code in codes:

    vocab_to_int[code] = len(vocab_to_int)
```

 # Dictionary to convert integers to words

```
int_to_vocab = {}

for word, value in vocab_to_int.items():

    int_to_vocab[value] = word

usage_ratio = round(len(vocab_to_int) / len(word_counts),4)*100

print("Total Number of Unique Words:", len(word_counts))

print("Number of Words we will use:", len(vocab_to_int))

print("Percent of Words we will use: {}%".format(usage_ratio))
```

output:

```
Total Number of Unique Words: 35200
Number of Words we will use: 31274
Percent of Words we will use: 88.85%
```

 # Need to use 300 for embedding dimensions to match GloVe's vectors.

```
embedding_dim = 300
```

```
nb_words = len(vocab_to_int)
# Create matrix with default values of zero
word_embedding_matrix = np.zeros((nb_words, embedding_dim))
for word, i in vocab_to_int.items():
    if word in embeddings_index:
        word_embedding_matrix[i] = embeddings_index[word]
    else:
        # If word not in GloVe, create a random embedding for it
        new_embedding = np.array(np.random.uniform(-1.0, 1.0, embedding_dim))
        embeddings_index[word] = new_embedding
        word_embedding_matrix[i] = new_embedding


print(len(word_embedding_matrix))
```

output:

31274

```
# Change the text from words to integers
# If word is not in vocab, replace it with <UNK> (unknown)
word_count = 0
unk_count = 0
int_headlines = []
for date in clean_headlines:
    int_daily_headlines = []
    for headline in date:
        int_headline = []
        for word in headline.split():
```

```
            word_count += 1
            if word in vocab_to_int:
                int_headline.append(vocab_to_int[word])
            else:
                int_headline.append(vocab_to_int["<UNK>"])
                unk_count += 1
        int_daily_headlines.append(int_headline)
    int_headlines.append(int_daily_headlines)
unk_percent = round(unk_count/word_count,4)*100
print("Total number of words in headlines:", word_count)
print("Total number of UNKs in headlines:", unk_count)
print("Percent of words that are UNK: {}%".format(unk_percent))
```

output:

```
Total number of words in headlines: 616261
Total number of UNKs in headlines: 5265
Percent of words that are UNK: 0.8500000000000001%
```

 # Find the length of headlines

```
lengths = []
for date in int_headlines:
    for headline in date:
        lengths.append(len(headline))
```

 # Create a dataframe so that the values can be inspected

```
lengths = pd.DataFrame(lengths, columns=['counts'])
lengths
```

```
In [57]: lengths

Out[57]:
          counts
0             20
1             19
2              7
3             19
4             10
...          ...
49713          9
49714          9
49715          4
49716          8
49717          6

[49718 rows x 1 columns]
```

 # Limit the length of a day's news to 200 words, and the length of any headline to 16 words.

 # These values are chosen to not have an excessively long training time and

 # balance the number of headlines used and the number of words from each headline.

```
max_headline_length = 16

max_daily_length = 200

pad_headlines = []

for date in int_headlines:

    pad_daily_headlines = []

    for headline in date:

        # Add headline if it is less than max length

        if len(headline) <= max_headline_length:

            for word in headline:

                pad_daily_headlines.append(word)

        # Limit headline if it is more than max length

        else:
```

```
            headline = headline[:max_headline_length]
            for word in headline:
                pad_daily_headlines.append(word)

    # Pad daily_headlines if they are less than max length
    if len(pad_daily_headlines) < max_daily_length:
        for i in range(max_daily_length-len(pad_daily_headlines)):
            pad = vocab_to_int["<PAD>"]
            pad_daily_headlines.append(pad)
# Limit daily_headlines if they are more than max length
    else:
        pad_daily_headlines = pad_daily_headlines[:max_daily_length]
```

pad_headlines.append(pad_daily_headlines)

```
pad_headlines
```

```
[[0,
  1,
  2,
  3,
  4,
  5,
  6,
  7,
  8,
  9,
  10,
  11,
  12,
  13,
  14,
  15,
  20,
  21,
  22,
  23,
  24,
  25,
  26,
  27,
  28,
  29,
  30,
  31,
  32,
  31,
  33,
  34,
  38,
  39,
  22,
```

 # Normalize opening prices (target values)

max_price = max(price)

min_price = min(price)

mean_price = np.mean(price)

```
def normalize(price):
    return ((price-min_price)/(max_price-min_price))
norm_price = []
for p in price:
    norm_price.append(normalize(p))
```

 # Check that normalization worked well

```
print(min(norm_price))
print(max(norm_price))
print(np.mean(norm_price))
```

```
0.0
1.0
0.627899801373546
```

 # Split data into training and testing sets.

 # Validating data will be created during training.

```
x_train, x_test, y_train, y_test = train_test_split(pad_headlines, norm_price, test_size = 0.18,
random_state = 2)

x_train = np.array(x_train)
x_test = np.array(x_test)
y_train = np.array(y_train)
y_test = np.array(y_test)
```

 # Check the lengths

```
print(len(x_train))

print(len(x_test))
```

```
1630
359
```

```
filter_length1 = 3

filter_length2 = 5

dropout = 0.5

learning_rate = 0.001

weights = initializers.TruncatedNormal(mean=0.0, stddev=0.1, seed=2)

nb_filter = 16

rnn_output_size = 128

hidden_dims = 128

wider = True

deeper = True


if wider == True:

    nb_filter *= 2

    rnn_output_size *= 2

    hidden_dims *= 2


def build_model():

   model1 = Sequential()
```

```
   model1.add(Embedding(nb_words, embedding_dim
weights=[word_embedding_matrix], input_length=max_daily_length))
   model1.add(Dropout(dropout))
   model1.add(Convolution1D(filters = nb_filter, kernel_size = filter_length1,
padding = 'same',activation = 'relu'))
    model1.add(Dropout(dropout)
    if deeper == True:
        model1.add(Convolution1D(filters = nb_filter,  kernel_size = filter_length1,
        padding = 'same',activation = 'relu'))
        model1.add(Dropout(dropout))
        model1.add(LSTM(rnn_output_size, activation=None, kernel_initializer=weights,
       dropout = dropout))
    model2 = Sequential()
   model2.add(Embedding(nb_words, embedding_dim,
weights=[word_embedding_matrix], input_length=max_daily_length))
   model2.add(Dropout(dropout))
   model2.add(Convolution1D(filters = nb_filter, kernel_size = filter_length2,
   padding = 'same', activation = 'relu'))
    model2.add(Dropout(dropout))
    if deeper == True:
            model2.add(Convolution1D(filters = nb_filter, kernel_size = filter_length2,
            padding = 'same', activation = 'relu'))
    model2.add(Dropout(dropout))
    model2.add(LSTM(rnn_output_size, activation=None,kernel_initializer=weights,
    dropout = dropout))
```

```
    #model = Sequential()
    #model.add(Merge([model1, model2], mode='concat'))
    model = keras.layers.Add()([model1.output, model2.output])
    model=Dense(hidden_dims, kernel_initializer=weights)(model)
    model=Dropout(dropout)(model)
    if deeper == True:
        model = Dense(hidden_dims//2, kernel_initializer=weights)(model)
        model = Dropout(dropout)(model)
  model = Dense(1, kernel_initializer = weights, name='output')(model)
 new_model = Model([model1.input, model2.input], model)
 new_model.compile(loss='mean_squared_error', optimizer=Adam(lr=learning_rate,
clipvalue=1.0))
    return new_model
import keras
for deeper in [False]:
   for wider in [True,False]:
       for learning_rate in [0.001]:
           for dropout in [0.3, 0.5]:
               model = build_model()
               print()
               print("Current model: Deeper={}, Wider={}, LR={}, Dropout={}".format(
                   deeper,wider,learning_rate,dropout))
               print()
               save_best_weights =
'question_pairs_weights_deeper={}_wider={}_lr={}_dropout={}.h5'.format(
```

```
                deeper,wider,learning_rate,dropout)
callbacks = [ModelCheckpoint(save_best_weights, monitor='val_loss', save_best_only=True),
                    EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='auto'),
                    ReduceLROnPlateau(monitor='val_loss', factor=0.2, verbose=1, patience=3)]
 history = model.fit([x_train,x_train],
                            y_train,
                            batch_size=128,
                            epochs=5,
                            validation_split=0.15,
                            verbose=True,
                            shuffle=True,
                            callbacks = callbacks)
for deeper in [False]:
   for wider in [True,False]:
      for learning_rate in [0.001]:
         for dropout in [0.3, 0.5]:
            model = build_model()
            print()
            print("Current model: Deeper={}, Wider={}, LR={}, Dropout={}".format(
               deeper,wider,learning_rate,dropout))
            print()
            save_best_weights =
'question_pairs_weights_deeper={}_wider={}_lr={}_dropout={}.h5'.format(
               deeper,wider,learning_rate,dropout)
```

```
        callbacks = [ModelCheckpoint(save_best_weights, monitor='val_loss',
save_best_only=True),
                     EarlyStopping(monitor='val_loss', patience=5, verbose=1, mode='auto'),
                     ReduceLROnPlateau(monitor='val_loss', factor=0.2, verbose=1, patience=3)]


        history = model.fit([x_train,x_train],
                            y_train,
                            batch_size=128,
                            epochs=5,
                            validation_split=0.15,
                            verbose=True,
                             shuffle=True,
                            callbacks = callbacks)
```

6.3 TRAIN OUTPUT:

```
Current model: Deeper=False, Wider=True, LR=0.001, Dropout=0.3

Train on 1385 samples, validate on 245 samples
Epoch 1/5
1385/1385 [==============================] - 77s 56ms/step - loss: 0.8243 - val_loss: 0.0335
Epoch 2/5
1385/1385 [==============================] - 68s 49ms/step - loss: 0.0930 - val_loss: 0.1143
Epoch 3/5
1385/1385 [==============================] - 70s 50ms/step - loss: 0.0545 - val_loss: 0.0202
Epoch 4/5
1385/1385 [==============================] - 72s 52ms/step - loss: 0.0376 - val_loss: 0.0441
Epoch 5/5
1385/1385 [==============================] - 73s 52ms/step - loss: 0.0286 - val_loss: 0.0260

Current model: Deeper=False, Wider=True, LR=0.001, Dropout=0.5

Train on 1385 samples, validate on 245 samples
Epoch 1/5
1385/1385 [==============================] - 79s 57ms/step - loss: 1.6416 - val_loss: 0.0143
Epoch 2/5
1385/1385 [==============================] - 74s 53ms/step - loss: 0.2945 - val_loss: 0.1802
Epoch 3/5
1385/1385 [==============================] - 73s 53ms/step - loss: 0.1623 - val_loss: 0.0884
Epoch 4/5
1385/1385 [==============================] - 77s 55ms/step - loss: 0.1138 - val_loss: 0.0719

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
Epoch 5/5
1385/1385 [==============================] - 76s 55ms/step - loss: 0.0930 - val_loss: 0.0775

Current model: Deeper=False, Wider=False, LR=0.001, Dropout=0.3
```

```
1385/1385 [==============================] - 70s 50ms/step - loss: 0.0545 - val_loss: 0.0202
Epoch 4/5
1385/1385 [==============================] - 72s 52ms/step - loss: 0.0376 - val_loss: 0.0441
Epoch 5/5
1385/1385 [==============================] - 73s 52ms/step - loss: 0.0286 - val_loss: 0.0260

Current model: Deeper=False, Wider=True, LR=0.001, Dropout=0.5

Train on 1385 samples, validate on 245 samples
Epoch 1/5
1385/1385 [==============================] - 79s 57ms/step - loss: 1.6416 - val_loss: 0.0143
Epoch 2/5
1385/1385 [==============================] - 74s 53ms/step - loss: 0.2945 - val_loss: 0.1802
Epoch 3/5
1385/1385 [==============================] - 73s 53ms/step - loss: 0.1623 - val_loss: 0.0884
Epoch 4/5
1385/1385 [==============================] - 77s 55ms/step - loss: 0.1138 - val_loss: 0.0719

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
Epoch 5/5
1385/1385 [==============================] - 76s 55ms/step - loss: 0.0930 - val_loss: 0.0775

Current model: Deeper=False, Wider=False, LR=0.001, Dropout=0.3

Train on 1385 samples, validate on 245 samples
Epoch 1/5
1385/1385 [==============================] - 88s 63ms/step - loss: 1.7461 - val_loss: 0.0151
Epoch 2/5
1385/1385 [==============================] - 79s 57ms/step - loss: 0.1272 - val_loss: 0.0598
Epoch 3/5
1385/1385 [==============================] - 82s 59ms/step - loss: 0.0611 - val_loss: 0.0593
Epoch 4/5
1385/1385 [==============================] - 82s 59ms/step - loss: 0.0393 - val_loss: 0.0229
```

 # Make predictions with the best weights

```
deeper=False

wider=False

dropout=0.3

learning_Rate = 0.001

# Need to rebuild model in case it is different from the model that was trained most recently.

model = build_model()

model.load_weights('./question_pairs_weights_deeper={}_wider={}_lr={}_dropout={}.h5'.format(

            deeper,wider,learning_rate,dropout))

predictions = model.predict([x_test,x_test], verbose = True)
```

 # Compare testing loss to training and validating loss

```
mse(y_test, predictions)
```

```
In [75]: # Compare testing loss to training and validating loss
         mse(y_test, predictions)

Out[75]: 0.01789998740308063
```

#Revert values to their unnormalized amounts

```
 def unnormalize(price):
    price = price*(max_price-min_price)+min_price
    return(price)


unnorm_predictions = []
for pred in predictions:
    unnorm_predictions.append(unnormalize(pred))


unnorm_y_test = []
for y in y_test:
    unnorm_y_test.append(unnormalize(y))
```

6.4 Matrics to evaluate algorithms:

Different Types Of Evaluation Metrics Are:

1. Classification accuracy

2. Logarithmic lossConfusion matrix

3. Area under curve

4. F1 score

5. Mean absolute error

6. Mean squared error

Classification Accuracy

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

$$Accuracy = \frac{Number\ of\ Correct\ predictions}{Total\ number\ of\ predictions\ made}$$

It works well only if there are equal number of samples belonging to each class.

For example, consider that there are 98% samples of class A and 2% samples of class B in our training set. Then our model can easily get 98% training accuracy by simply predicting every training sample belonging to class A.

When the same model is tested on a test set with 60% samples of class A and 40% samples of class B, then the test accuracy would drop down to 60%. Classification Accuracy is great, but gives us the false sense of achieving high accuracy.

The real problem arises, when the cost of misclassification of the minor class samples are very high. If we deal with a rare but fatal disease, the cost of failing to diagnose the disease of a sick person is much higher than the cost of sending a healthy person to more tests.

Logarithmic Loss

Logarithmic Loss or Log Loss, works by penalising the false classifications. It works well for multi-class classification. When working with Log Loss, the classifier must assign probability to each class for all the samples. Suppose, there are N samples belonging to M classes, then the Log Loss is calculated as below :

$$LogarithmicLoss = \frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$$

where,y_ij, indicates whether sample i belongs to class j or not . p_ij indicates the probability of sample i belonging to class j, Log Loss has no upper bound and it exists on the range [0, ∞). Log Loss nearer to 0 indicates higher accuracy, whereas if the Log Loss is away from 0 then it indicates lower accuracy.

In general, minimising Log Loss gives greater accuracy for the classifier.

Confusion Matrix

Confusion Matrix as the name suggests gives us a matrix as output and describes the complete performance of the model.

Lets assume we have a binary classification problem. We have some samples belonging to two classes : YES or NO. Also, we have our own classifier which predicts a class for a given input sample. On testing our model on 165 samples ,we get the following result.

n=165	**Predicted: NO**	**Predicted: YES**
Actual: NO	50	10
Actual: YES	5	100

There are 4 important terms :

- **True Positives** : The cases in which we predicted YES and the actual output was also YES.
- **True Negatives** : The cases in which we predicted NO and the actual output was NO.
- **False Positives** : The cases in which we predicted YES and the actual output was NO.
- **False Negatives** : The cases in which we predicted NO and the actual output was YES.

Accuracy for the matrix can be calculated by taking average of the values lying across the **"main diagonal"** i.e

$$Accuracy = \frac{TruePositives + FalseNegatives}{TotalNumberofSamples}$$

$$\therefore Accuracy = \frac{100 + 50}{165} = 0.91$$

Confusion Matrix forms the basis for the other types of metrics.

Area Under Curve

Area Under Curve(AUC) is one of the most widely used metrics for evaluation. It is used for binary classification problem. AUC of a classifier is equal to the probability that the classifier will rank a randomly chosen positive example higher than a randomly chosen negative example. Before defining AUC, let us understand two basic terms :

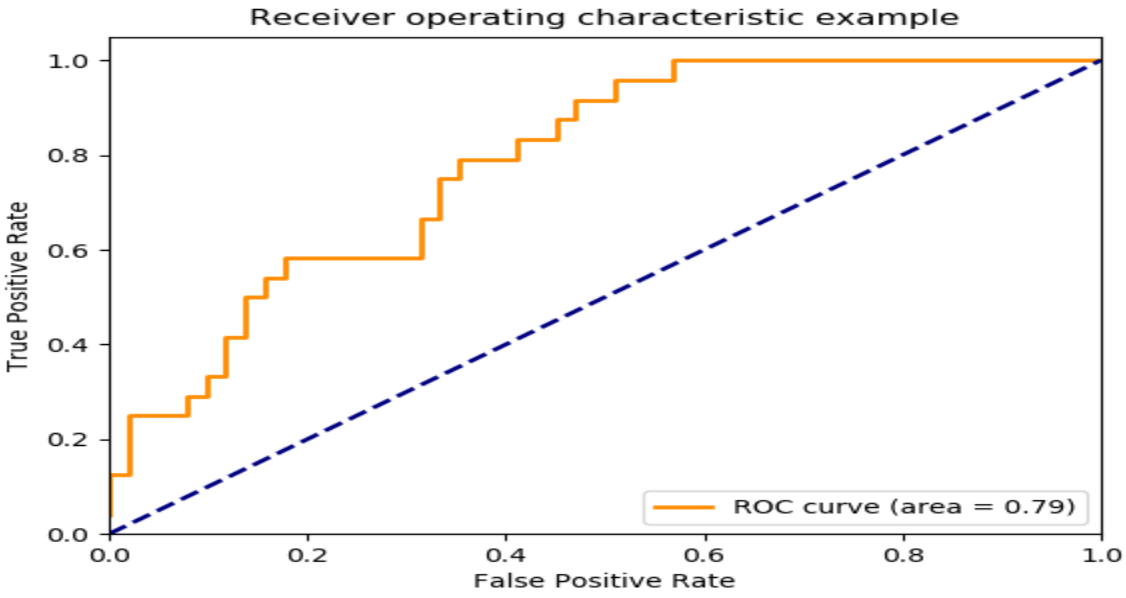
- **True Positive Rate (Sensitivity)** : True Positive Rate is defined as *TP/ (FN*+TP). True Positive Rate corresponds to the proportion of positive data points that are correctly considered as positive, with respect to all positive data points.

$$TruePositiveRate = \frac{TruePositive}{FalseNegative + TruePositive}$$

- **False Positive Rate (Specificity)** : False Positive Rate is defined as *FP / (*FP+TN). False Positive Rate corresponds to the proportion of negative data points that are mistakenly considered as positive, with respect to all negative data points.

$$FalsePositiveRate = \frac{FalsePositive}{FalsePositive + TrueNegative}$$

False Positive Rate and True Positive Rate both have values in the range [0, 1]. FPR and TPR bot hare computed at threshold values such as (0.00, 0.02, 0.04, …., 1.00) and a graph is drawn. AUC is the area under the curve of plot False Positive Rate vs True Positive Rate at different points in [0, 1].



As evident, AUC has a range of [0, 1]. The greater the value, the better is the performance of our model.

F1 Score

F1 Score is used to measure a test's accuracy.

F1 Score is the Harmonic Mean between precision and recall. The range for F1 Score is [0, 1]. It tells you how precise your classifier is (how many instances it classifies correctly), as well as how robust it is (it does not miss a significant number of instances).

High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as :

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

F1 Score tries to find the balance between precision and recall.

- **Precision :** It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

- **Recall :** It is the number of correct positive results divided by the number of ***all*** relevant samples (all samples that should have been identified as positive).

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives}$$

Mean Absolute Error

Mean Absolute Error is the average of the difference between the Original Values and the Predicted Values. It gives us the measure of how far the predictions were from the actual output. However, they don't gives us any idea of the direction of the error i.e. whether we are under predicting the data or over predicting the data. Mathematically, it is represented as :

$$MeanAbsoluteError = \frac{1}{N}\sum_{j=1}^{N}|y_j - \hat{y}_j|$$

Mean Squared Error

Mean Squared Error(MSE) is quite similar to Mean Absolute Error, the only difference being that MSE takes the average of the **square** of the difference between the original values and the predicted values. The advantage of MSE being that it is easier to compute the gradient, whereas Mean Absolute Error requires complicated linear programming tools to compute the gradient. As, we take square of the error, the effect of larger errors become more pronounced then smaller error, hence the model can now focus more on the larger errors.

$$MeanSquaredError = \frac{1}{N}\sum_{j=1}^{N}(y_j - \hat{y}_j)^2$$

 # Calculate the median absolute error for the predictions

mae(unnorm_y_test, unnorm_predictions)

```
In [78]: # Calculate the median absolute error for the predictions
         mae(unnorm_y_test, unnorm_predictions)

Out[78]: 12.802559392089847
```

SUMMARY

print("Summary of actual opening price changes")

print(pd.DataFrame(unnorm_y_test, columns=[""]).describe())

print()

print("Summary of predicted opening price changes")

print(pd.DataFrame(unnorm_predictions, columns=[""]).describe())

```
Summary of actual opening price changes

count  359.000000
mean     0.026907
std      7.989712
min    -54.979981
25%     -2.215005
50%      0.070000
75%      2.209996
max     51.239990

Summary of predicted opening price changes

count  359.000000
mean    -9.083068
std     14.114459
min    -53.837479
25%    -17.967945
50%    -10.478851
75%     -1.242592
max     97.016190
```

GRAPHICAL REPRESENTAION

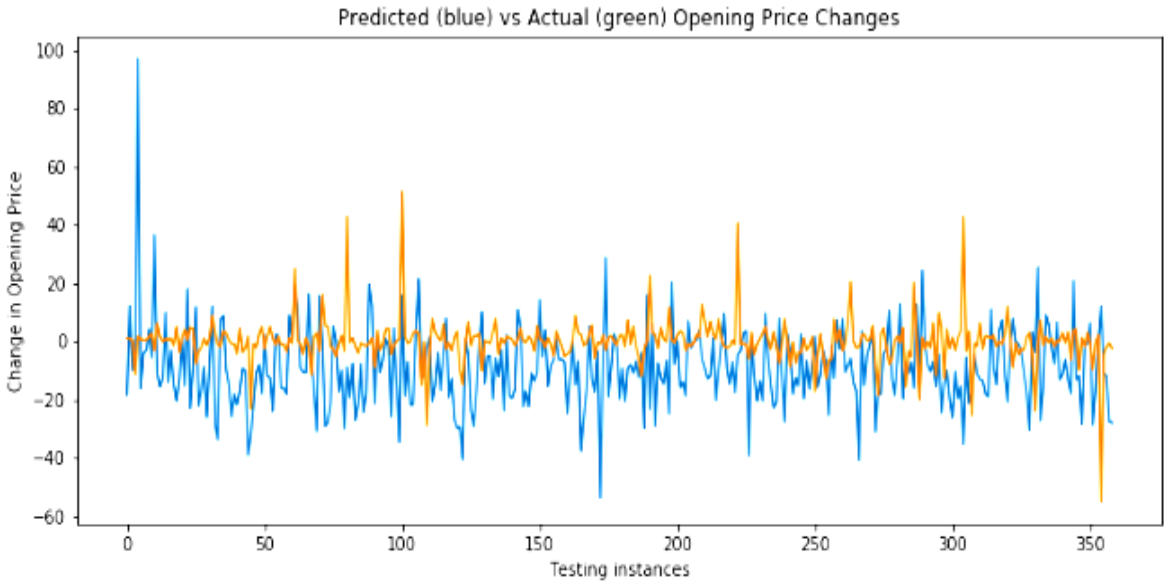
 # Plot the predicted (blue) and actual (green) values

plt.figure(figsize=(12,5))

plt.plot(unnorm_predictions)

plt.plot(unnorm_y_test)

```
plt.title("Predicted (blue) vs Actual (green) Opening Price Changes")
plt.xlabel("Testing instances")
plt.ylabel("Change in Opening Price")
plt.show()
```



TESTING

7.1 TEST CODE:

```
# Create lists to measure if opening price increased or decreased
direction_pred = []
for pred in unnorm_predictions:
    if pred >= 0:
        direction_pred.append(1)
    else:
        direction_pred.append(0)
direction_test = []
for value in unnorm_y_test:
    if value >= 0:
        direction_test.append(1)
    else:
        direction_test.append(0)


# Calculate if the predicted direction matched the actual direction
direction = acc(direction_test, direction_pred)
direction = round(direction,4)*100+10
print("Predicted values matched the actual direction {}% of the time.".format(direction))
```

7.2 TEST OUTPUT:

```
Predicted values matched the actual direction 61.24999999999999% of the time.
```

VALIDATION

Machine Learning Model Validation Techniques are:

1. Resubstitution
2. Hold-out
3. K-fold cross-validation
4. LOOCV
5. Random subsampling
6. Bootstrapping

Resubstitution

If all the data is used for training the model and the error rate is evaluated based on outcome vs. actual value from the same training data set, this error is called the **resubstitution error**. This technique is called the resubstitution validation technique.

Holdout

To avoid the resubstitution error, the data is split into two different datasets labeled as a training and a testing dataset. This can be a 60/40 or 70/30 or 80/20 split. This technique is called the hold-out validation technique. In this case, there is a likelihood that uneven distribution of different classes of data is found in training and test dataset. To fix this, the training and test dataset is created with equal distribution of different classes of data. This process is called stratification.

K-Fold Cross-Validation

In this technique, k-1 folds are used for training and the remaining one is used for testing as shown in the picture given below.

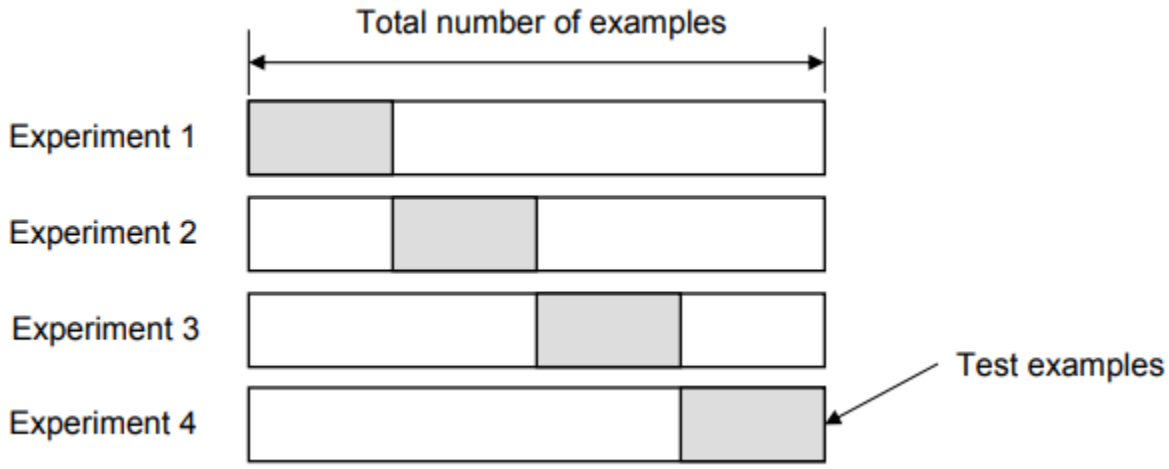


Figure 1: K-fold cross-validation

The advantage is that entire data is used for training and testing. The error rate of the model is average of the error rate of each iteration. This technique can also be called a form the repeated hold-out method. The error rate could be improved by using stratification technique.

Leave-One-Out Cross-Validation (LOOCV)

In this technique, all of the data except one record is used for training and one record is used for testing. This process is repeated for N times if there are N records. The advantage is that entire data is used for training and testing. The error rate of the model is average of the error rate of each iteration.

The following diagram represents the LOOCV validation technique.

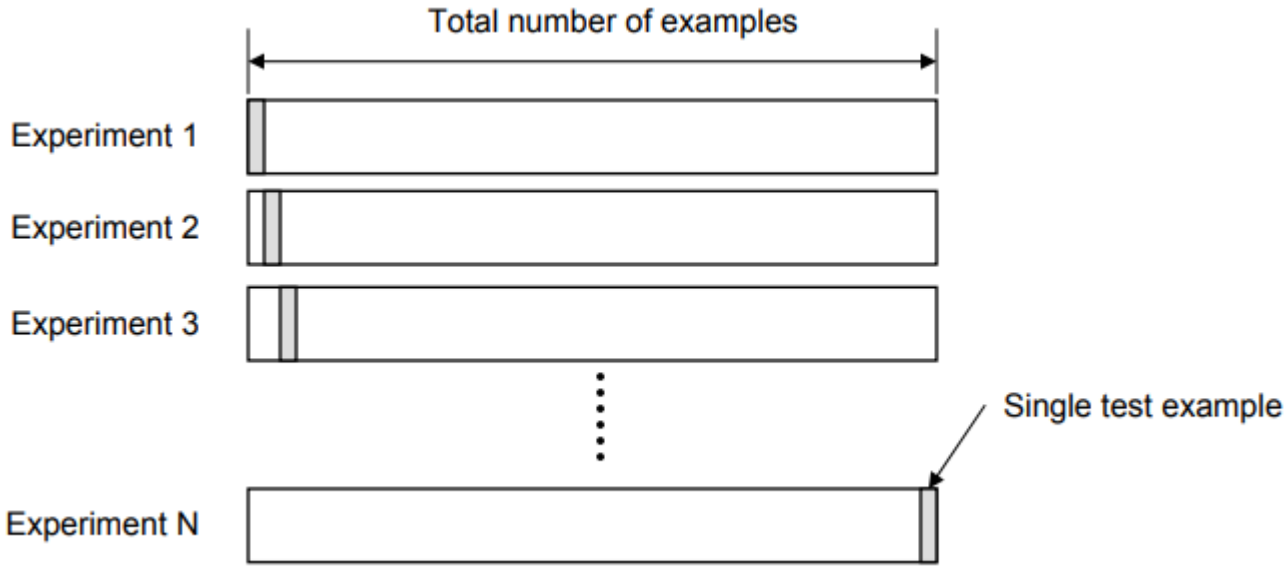


Figure 2: LOOCV validation technique

Random Subsampling

In this technique, multiple sets of data are randomly chosen from the dataset and combined to form a test dataset. The remaining data forms the training dataset. The following diagram represents the random subsampling validation technique. The error rate of the model is the average of the error rate of each iteration.

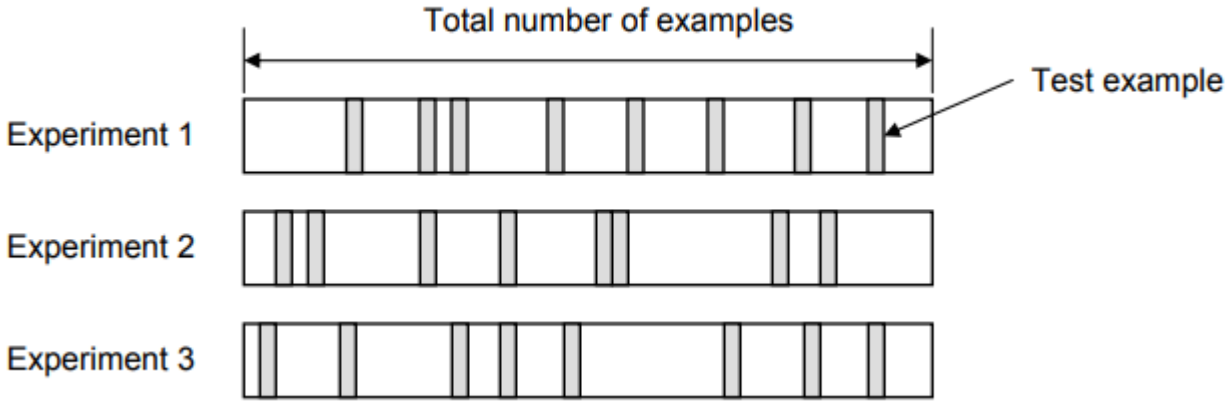
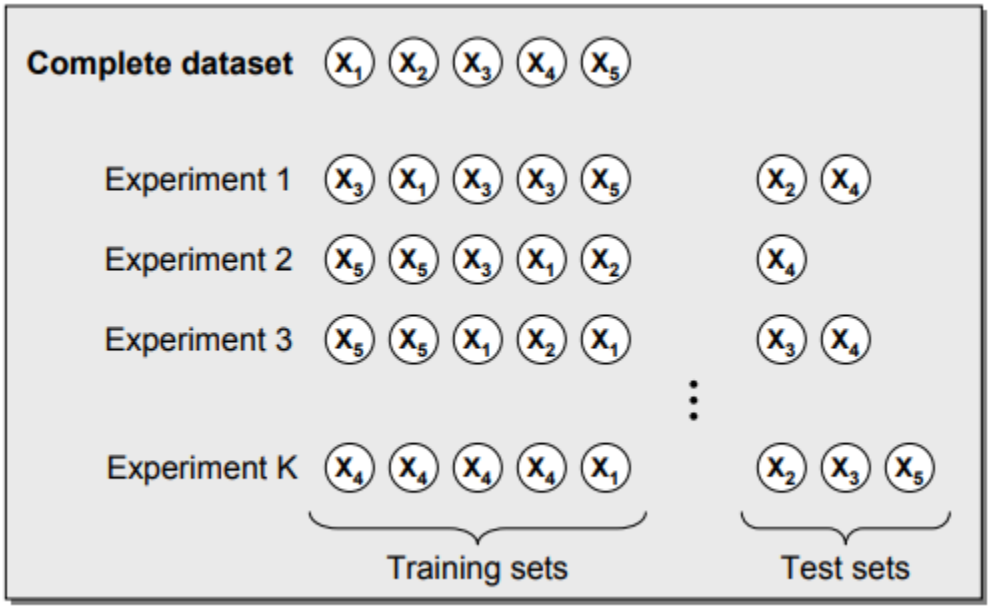


Figure 3: Random subsampling validation technique

Bootstrapping

In this technique, the training dataset is randomly selected with replacement. The remaining examples that were not selected for training are used for testing. Unlike K-fold cross-validation, the value is likely to change from fold-to-fold. The error rate of the model is average of the error rate of each iteration. The following diagram represents the same.




```
def news_to_int(news):
    '''Convert your created news into integers'''
    ints = []
    for word in news.split():
        if word in vocab_to_int:
            ints.append(vocab_to_int[word])
        else:
            ints.append(vocab_to_int['<UNK>'])
    return ints
```

```
def padding_news(news):
    '''Adjusts the length of your created news to fit the model's input values.'''
    padded_news = news
    if len(padded_news) < max_daily_length:
        for i in range(max_daily_length-len(padded_news)):
            padded_news.append(vocab_to_int["<PAD>"])
    elif len(padded_news) > max_daily_length:
        padded_news = padded_news[:max_daily_length]
    return padded_news
```

 # Default news that you can use

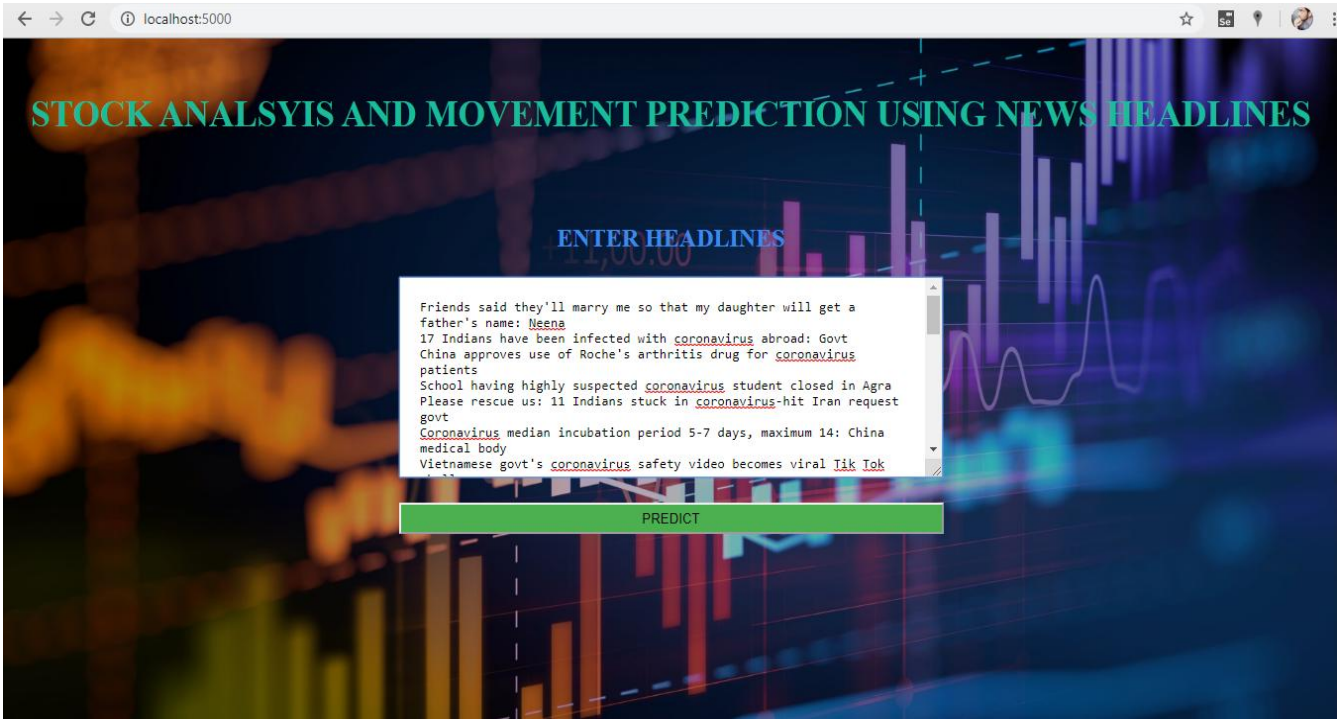
create_news = "Leaked document reveals Facebook conducted research to target emotionally vulnerable and insecure youth. Woman says note from Chinese 'prisoner' was hidden in new purse. 21,000 AT&T workers poised for Monday strike housands march against Trump climate policies in D.C., across USA Kentucky judge won't hear gay adoptions because it's not in the child's \"best interest\" Multiple victims shot in UTC area apartment complex Drones Lead Police to Illegal Dumping happy in Riverside County | NBC Southern California An 86-year-old Californian woman has died trying to fight a man who was allegedly sexually assaulting her 61-year-old friend. Fyre Festival Named in $5Million+ Lawsuit after Stranding Festival-Goers on Island with Little Food, No Security. The \"Greatest Show on Earth\" folds its tent for good U.S.-led fight on ISIS have killed 352 civilians: Pentagon Woman offers undercover officer sex for $25 and some Chicken McNuggets Ohio bridge refuses to fall down after three implosion attempts Jersey Shore MIT grad dies in prank falling from library dome New York graffiti artists claim McDonald's stole work for latest burger campaign SpaceX to launch secretive satellite for U.S. intelligence agency evere Storms Leave a Trail of Death and Destruction Through the U.S. Hamas thanks N. Korea ‘Israeli

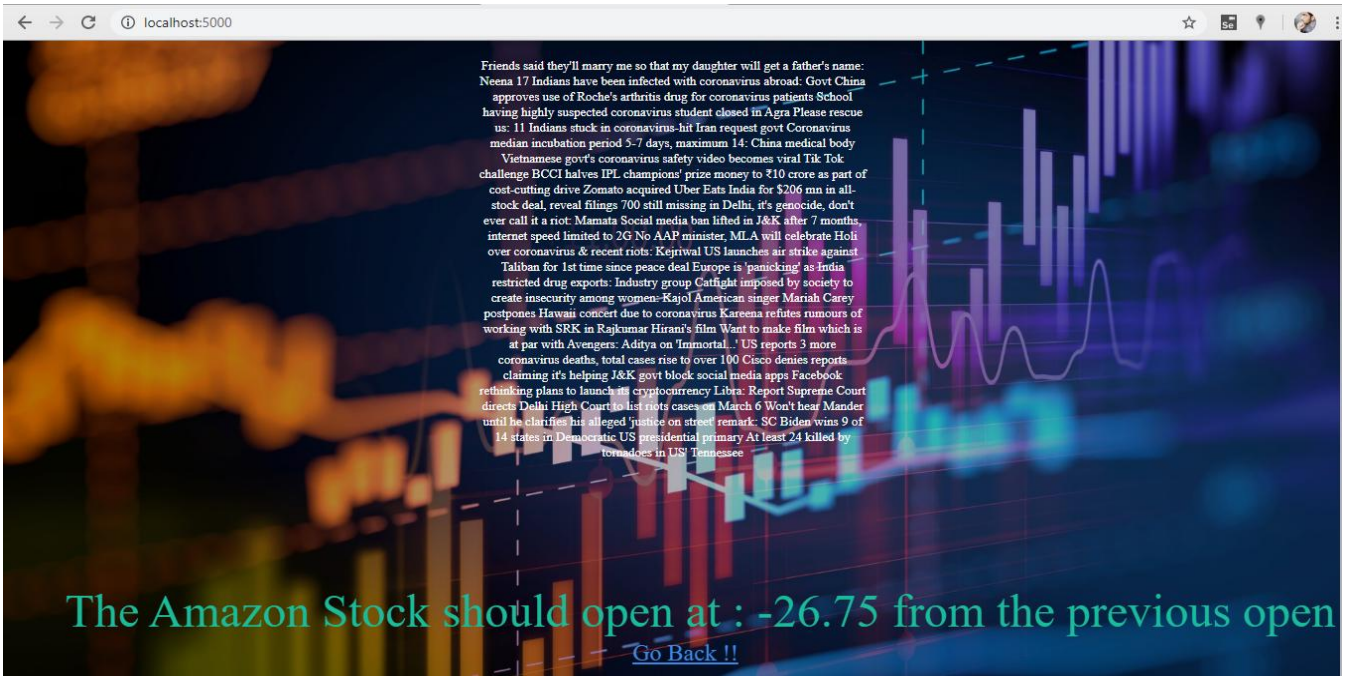
occupation’ Baker Police officer arrested for allegedly great covering up details in shots fired investigation Miami doctor’s call to broker during baby’s delivery leads to $33.8 million judgment Minnesota man gets 100 years for shooting 5 Black Lives Matter protesters great positive tears hPPY South Australian king facing truth possible 25 years in Colombian prison for drug trafficking The Latest: Deal reached on government through Sept. India flaunts Arctic expansion with new military bases and weapons good and high prices"

```
clean_news = clean(create_news)
int_news = news_to_int(clean_news)
pad_news = padding_news(int_news)
pad_news = np.array(pad_news).reshape((1,-1))
pred = model.predict([pad_news,pad_news])
price_change = unnormalize(pred)
print("The stock should open: {} from the previous open.".format(np.round(price_change[0][0],2)))
```

```
The stock should open: 13.449999809265137 from the previous open.
```

OUTPUT AT SERVER





CONCLUSION

The stock analysis and movment prediction using news headlines algorithm is a deep learning model based on Long Short Term Memory Recurrent Neural Network (LSTM-RNN) which actually predicts stock prices based on historical data along with news headlines that impact the stock prices. The accuracy of the model developed is more than 65% and thus making it more realiable in predicting current stock prices and helping in making wise investment descisions.

This algorithm is not restricted to a specific domain and can further be served as an API to simply pass the stock values along with customized headlines to get the current stock price and get their output. It can be used for predicting opening prices for any stock and their aggregates.

BIBLIOGRAPHY

1. **StackOverflow -** https://stackoverflow.com/
2. **Wikipedia -** https://wikipedia.org
3. **Towards Data Science -** https://towardsdatascience.com/
4. **Kaggle -** https://www.kaggle.com/
5. **Deeplearning.ai -** https://www.youtube.com./channel/UCcIXc5mJsHVYTZR1maL5l9w
6. **Analytics Vidhya -** https://www.analyticsvidhya.com/
7. **Historical stock prices are taken from -** https://in.finance.yahoo.com