



**Meet IB, the InvestoBot**

### **Team Members**

Nishita Sikka – 001816375  
Vrushali Shah – 001826666  
Sunil Yadav – 001492711  
Gaurang Davda– 001826203  
Prasad Aroskar – 001305905

# **Index**

## **1. Introduction**

## **2. Objective**

## **3. Scope**

## **4. Design**

### 4.1. Architecture

### 4.2. Data Set

### 4.3. Algorithms

#### 4.3.1. Bag of Words (BoW)

#### 4.3.2. TF – IDF vectorizer

#### 4.3.3. NLTK

#### 4.3.4. Lancaster stemmer

#### 4.3.5. Sequential least squares programming

#### 4.3.6. Artificial Neural Network

### 4.4. Processes

#### 4.4.1. Training

#### 4.4.2. Preprocessor

#### 4.4.3. Logic Adaptors

#### 4.4.4. Stock Prediction

#### 4.4.5. Portfolio Optimization

### 4.5. Web Services

#### 4.5.1. Flask

#### 4.5.2. Slack

### 4.6. Results

#### 4.6.1. Model

#### 4.6.2. Chatbot

## **5. Future Scope**

## **6. Conclusion**

## **7. Reference**

## 1. Introduction

Web development allows you to build beautiful websites that scale well on multiple devices. Whether you use Siri, rely on Alexa or have a Roomba, there's no escaping the fact that robots are part of our daily lives now. It's not a bad thing that we can automate some of life's most mundane, tedious tasks. Although Apps are everywhere, and people are sick of them: between 80-90% of apps are used once and then deleted. The one exception is messaging apps, which account for 91% of time spent on mobiles in the U.S. These messaging apps are of different and prove to be useful in a variety of aspects.

For example, the top Robo Investment Advisors like wealth front, Ellevest, Betterment etc. have been good news for consumers looking for low cost financial advice and looking forward to investing. These bots answer real-world questions about advising consumers regarding their investment portfolios serving as the best low-cost investment advisor. These bot services provide modern portfolio theory and the efficient market hypothesis. The bot plugs your answers into an algorithm that determines the kind of portfolio and asset allocation that's appropriate for your age, risk tolerance and time horizon. Hence, we chose to work on our own “The InvestoBot” to help people with their financial needs.

## 2. Objective

One way to stay competitive in modern business is to automate as many of your processes as possible. We can evidence this in our day to day lives. For example, say the self-checkout at grocery stores and ordering kiosks at restaurants. In fact, Amazon just opened a store without any cashiers or self-checkouts, limiting human interactions to those only absolutely necessary. The value in chatbots comes from their ability to automate conversations throughout your organization. Below are few benefits businesses realize when using chatbots.

- **Save Time and Money**

By automating conversations that would otherwise require an employee to answer, organizations save both time and money that can then be allocated to other efforts. Sprout Social research shows the number of social messages requiring a response from a brand increased by 18% from 2015 to 2016, you save countless hours by automating responses with a chatbot.

- **Generate Leads & Revenue**

Chatbots use direct messages to gather information necessary to provide effective support. For example, asking users why they're visiting your page is one question that is most likely to be asked in every engagement. Automating this initial interaction allows users to share the information needed for the agent to better serve them without requiring a human to ask for it.

- **Guide Users to Better Outcomes**

Customers don't always know where to go to find the information they're interested in. In fact, your customers may not even know what it is they're interested in. By asking a

series of qualifying questions, you route users to the best place for them to find the information they want.

- **Provide ‘After Hours’ Support**

Drift data states about the most popular use of chatbots is to provide quick answers in an emergency. However, organizations that don’t offer 24- hour support won’t provide answers when the office is closed. By using a robust chatbot when your business is closed, customers still gain access to the information they need. Chatbots help you significantly decrease the average time to respond, bringing you closer to your customers’ expectations.

- **Engage Users in a Unique Way**

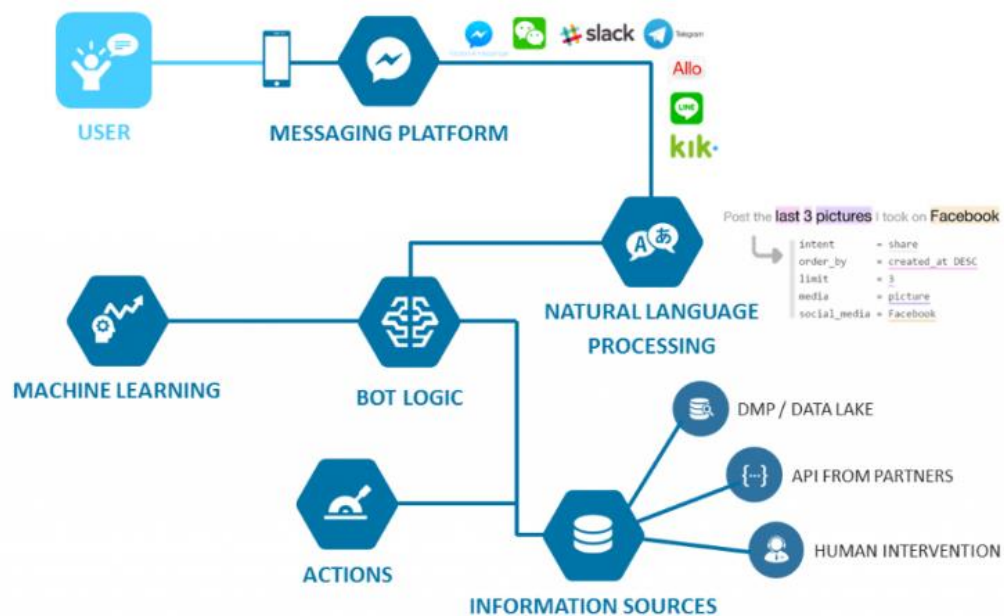
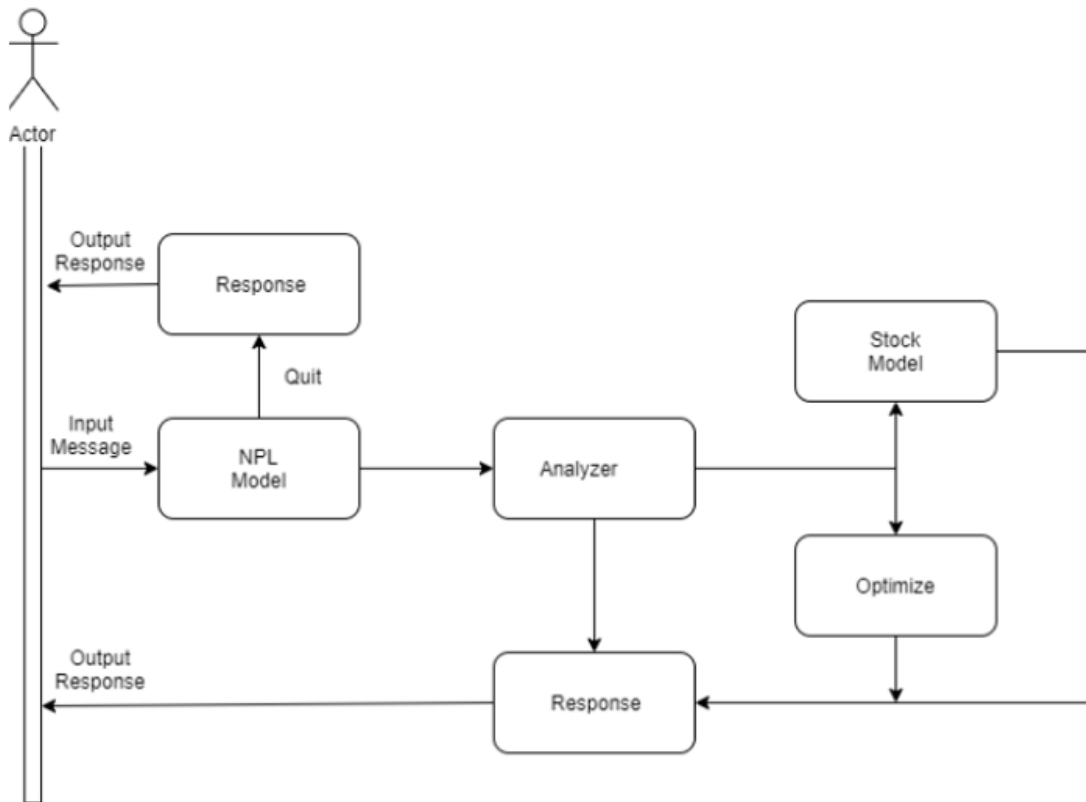
Traditionally, customer questions were routed to businesses via email or the telephone, which made user experiences standard and non-customized. But chatbots offer a new, fun and interactive way to engage with brands.

### **3. Scope**

The scope of the project is to build an intelligent virtual human who can guide and answers all the queries of the customers without the intervene of any physical presence of staff. Targeting the people who are looking to invest or get information about their portfolio the scope is to build a chatbot with an objective to guide its own registered customers as well as unregistered customers with information about investments, current stock values of specific companies and optimizing their current stocks. Using standard python libraries, various algorithms like NLTK, TF-IDF Vectorizer, Bag of Words, etc. and 'Slack' as the front-end UI we aim to deploy the project on cloud so that it will be live and accessible to multiple users. We are limiting the number of portfolios to 5 accounts only and number of stocks to 5 only.

## 4. Design

### 4.1. Architecture



It comprises of a Natural Language Understanding engine which, at first, processes the input message to remove all the stop words, punctuations and irrelevant parts of the sentence. Then it transforms the sentence into the sparse matrix and is inputted to the NLP model. The transformations will be done using libraries NLTK, TF-IDF Vectorizer, and REGEX package.

The machine learning algorithm will be used to, firstly, predict the input data belongs to which category (also known as tag). The output of the algorithm helps understand the intent of the input sentence. The algorithm then finds an optimal output response for a list of possible responses attached to that tag/category. This is the generic chatbot flow.

Different ML algorithms will be tried for classifying the sentence to proper tag, spanning from Logistic Regression to XGBoost to Artificial Neural Network to pretrained algorithms like Glove, BagOfWords, Seq2Seq. The models will be cross validated and evaluated for overfitting/underfitting and the best model will be implemented in the chatbot. What if when a sentence comes related to investment like, “I want the best portfolio allocation for my portfolio.”? We have got it covered. A mathematical function will be used to initialize the allocations and maximize the profits while constraining the Sharpe ratio within appropriate risk boundaries as defined by the user

## **4.2. Data**

For the project, we have created our own data files since there were not many data sets related to investments.

Data Files:

- convertcsv.json



313 lines (311 sloc) | 11 KB

RawBlameHistory

```
1 {
2   "intents":[
3     {
4       "tag":"greeting",
5       "patterns":[
6         "Hi",
7         "Hey",
8         "Hello",
9         "I am fine. And you?",
10        "How are you",
11        "Is there anyone",
12        "Good Morning, I need some help",
13        "Howdy !!",
14        "Hey, I have a question regarding investments.",
15        "Hello, Good Day ! How are you ?"
16      ],
17      "response":[
18        "Hey there, friend! I'm InvestoBot, the chatbot for helping you make better investments. Before we proceed ahead, may I know yo
19      ]
20    },
```

- portfolio\_data.csv

	ACCOUNT_NO	NAME	GOOGL	MSFT	FB	AAPL	TWTR	TOTAL
1	100000001	John Smith	0	0.3	0	0.4	0.3	1021.47
2	100000002	Daniel Brown	0.4	0	0.6	0	0	5032.8
3	100000003	Sarah Jones	1	0	0	0	0	10546.8
4	100000004	Emma Williams	0.1	0.2	0.3	0.3	0.1	2166.53
5	100000005	Joseph Thomas	0	0.4	0.1	0.2	0.3	942.43

- daily\_AAPL.csv

Search this file...

	date	1. open	2. high	3. low	4. close	5. volume
1	1999-11-29	94.25	99.75	93.25	94.56	4144300.0
2	1999-11-30	98.12	103.75	97.37	97.87	7528400.0
3	1999-12-01	101.0	104.5	100.06	103.06	5522900.0
4	1999-12-02	103.12	110.62	101.75	110.19	5065700.0
5	1999-12-03	112.19	115.56	111.87	115.0	5785000.0
6	1999-12-06	114.56	117.31	111.44	116.0	4167700.0
7	1999-12-07	116.56	118.0	114.0	117.81	3973400.0
8	1999-12-08	116.25	117.87	109.5	110.06	3681700.0

- daily\_FB.csv

Search this file...

	date	1. open	2. high	3. low	4. close	5. volume
1	2012-05-18	42.05	45.0	38.0	38.2318	573576400.0
2	2012-05-21	36.53	36.66	33.0	34.03	168192700.0
3	2012-05-22	32.61	33.59	30.94	31.0	101786600.0
4	2012-05-23	31.37	32.5	31.36	32.0	73600000.0
5	2012-05-24	32.95	33.21	31.77	33.03	50237200.0
6	2012-05-25	32.9	32.95	31.11	31.91	37149800.0
7	2012-05-29	31.48	31.69	28.65	28.84	78063400.0
8	2012-05-30	28.695	29.55	27.86	28.19	57267900.0
9	2012-05-31	28.545	29.67	26.83	29.6	111639200.0
10	2012-06-01	28.892	29.15	27.39	27.72	41855500.0

## 4.3. Algorithms

We have used the below algorithms in our project:

### 4.3.1. Bag of Words (BoW)

A bag-of-words model, or BoW for short, is a way of extracting features from text for use in modeling, such as with machine learning algorithms. A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.
2. A measure of the presence of known words.

It is called a “bag” of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

The intuition is that documents are similar if they have similar content. Further, that from the content alone we can learn something about the meaning of the document. The bag-of-words can be as simple or complex. The complexity comes both in deciding how to design the vocabulary of known words (or tokens) and how to score the presence of known words.

Steps include:

1. Collect data
2. Design the Vocabulary
3. Create Vectors

```
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_words]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return np.array(bag)

results = model.predict([bag_of_words("Hey I am gaurang", words)])[0]
results_index = np.argmax(results)
tag = labels[results_index]
print(tag)
if results[results_index] < 0.7:
    print('I don\'t know')
    print(results[results_index])
```

### 4.3.2. TF - IDF Vectorizer (Word2Vec)

TF-IDF stands for term frequency–inverse document frequency, which is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. It is used as a weighting factor in searches of information retrieval, text mining, and user modeling. The tf–idf value increases proportionally to the number of times a word appears in the document and is offset by the number of documents in the corpus that contain the word, which helps to adjust for the fact that some words appear more frequently in general. The approach is to rescale the frequency of words by how often they appear in all documents, so that the scores for frequent words like “the” that are also frequent across all documents are penalized. We read the json file containing the tags, greeting and responses data, then perform various processor functions on it to check for emojis, special characters, URLs, whitespaces. Once this step is done, we use the TfidfVectorizer() method to create the “tfv” vector.

```
def preprocess_message(self, message):
    processed_message = []
    # Convert to lower case
    message = message.lower()
    # Replaces URLs with the word URL
    message = re.sub(r'((www\.[\S]+)|(https?://[\S]+))', ' URL ', message)
    # Replace 2+ dots with space
    message = re.sub(r'\.{2,}', ' ', message)
    # Strip space, " and ' from message
    message = message.strip(' "\'')
    # Replace emojis with either EMO_POS or EMO_NEG
    message = self.handle_emojis(message)
    # Replace multiple spaces with a single space
    message = re.sub(r'\s+', ' ', message)
    words = message.split()

    for word in words:
        word = self.preprocess_word(word)
        if self.is_valid_word(word):
            processed_message.append(word)
    return ' '.join(processed_message)

def transformCleanedMessage(self, message):
    tfv=TfidfVectorizer(min_df=0, max_features=None, strip_accents='unicode', lowercase =True,
    analyzer='word', token_pattern=r'\w{3,}', ngram_range=(1,1), sublinear_tf=True, stop_words = "engl
ish")
    transformedMessage=tfv.fit_transform(message)
    return transformedMessage
```

### **4.3.3. NLTK**

NLTK stands for Natural Language Toolkit. The NLTK module is a massive tool kit, aimed at helping you with the entire Natural Language Processing (NLP) methodology. NLTK will aid with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, and then even with helping your machine to understand what the text is all about. In this series, we're going to tackle the field of opinion mining, or sentiment analysis. It is one of the most powerful NLP libraries which contains packages to make machines understand human language and reply to it with an appropriate response. Tokenization, Stemming, Lemmatization, Punctuation, Character count and word count are included in this library. We use NLP to make predictions and follows the below steps in sequence Data preparation, feature extraction, training and prediction. Steps included in NLTK are importing nltk, loading nltk data, tokenizing sentences, tagging sentences, counting tags and running the nlp scripts.

### **4.3.4. Lancaster Stemmer**

Stemming is a method for collapsing distinct word forms. This could help reduce the vocabulary size, thereby sharpening one's results, especially for small data sets. Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

There are 3 main types of stemmers namely Porter, Snowball and Lancaster. Lancaster is considered to be most aggressive than others. It is the fastest algorithm and reduces the working set of words hugely. With porter and snowball, the stemmed

representations are usually intuitive to a reader, not so with Lancaster, as many shorter words will become totally obfuscated.

## Neural Nets

```
with open(os.getcwd()+ '/../Data/convertcsv.json') as file:
    data = json.load(file)
words = []
labels = []
docs_x = []
docs_y = []

for intent in data['intents']:
    for pattern in intent['patterns']:
        wrds = nltk.word_tokenize(pattern)
        words.extend(wrds)
        docs_x.append(wrds)
        docs_y.append(intent["tag"])

    if intent['tag'] not in labels:
        labels.append(intent['tag'])

stemmer = LancasterStemmer()
words = [stemmer.stem(w.lower()) for w in words if w != "?"]
words = sorted(list(set(words)))

labels = sorted(labels)

training = []
output = []

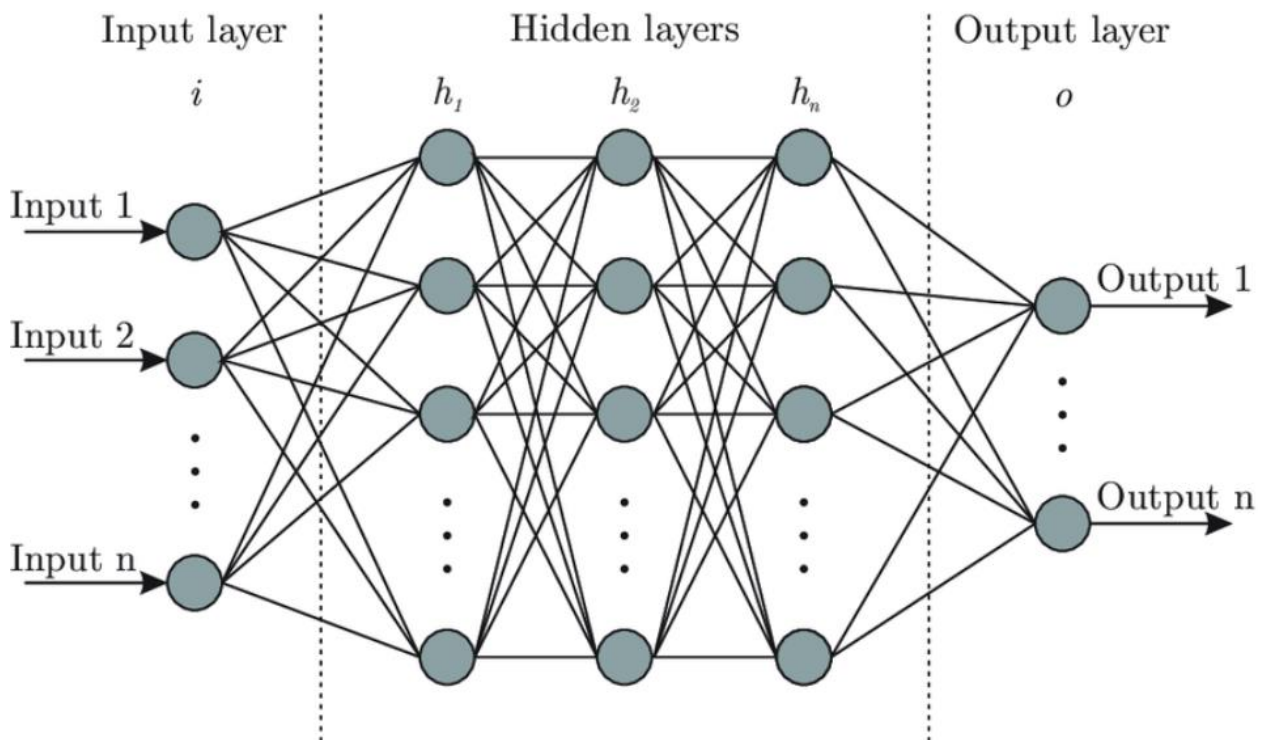
out_empty = [0 for _ in range(len(labels))]
```

### 4.3.5. Sequential Least Squares Programming

SciPy optimize provides functions for minimizing (or maximizing) objective functions, possibly subject to constraints. It includes solvers for nonlinear problems (with support for both local and global optimization algorithms), linear programming, constrained and nonlinear least-squares, root finding and curve fitting. It solves a linear least-squares problem with bounds on the variables.

### 4.3.6. Artificial Neural Network

Neural Networks are a class of models within the general machine learning literature. Neural networks are a specific set of algorithms that has revolutionized the field of machine learning. They are inspired by biological neural networks and the current so called deep neural networks. A neural network is made up of neurons connected to each other; at the same time, each connection of our neural network is associated with a weight. Each neuron has an activation function that defines the output of the neuron. The activation function is used to introduce non-linearity in the modeling capabilities of the network. Artificial Neural Network algorithms are inspired by the human brain. The artificial neurons are interconnected and communicate with each other. Each connection is weighted by previous learning events and with each new input of data more learning takes place.



## 4.4. Processes

### 4.4.1. Training

#### 1. Importing libraries

```
: import pandas as pd
import numpy as np
import os
import json
import re
import numpy as np
import pickle
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
import tensorflow as tf
import tflearn
import nltk
from nltk.stem.lancaster import LancasterStemmer
```

#### 2. Reading the data

```
dfload = pd.read_json(os.getcwd()+ '/../Data/convertcsv.json')
df = pd.DataFrame(columns=['TAGS', 'PATTERNS', 'RESPONSES'])
i = 0
for _, r in dfload.iterrows():
    for resp in r[0]['response']:
        for p in r[0]['patterns']:
            df.set_value(i, 'PATTERNS', p)
            df.set_value(i, 'TAGS', r[0]['tag'])
            df.set_value(i, 'RESPONSES', resp)
            i += 1
```

#### 3. Remove blank values

```
df.dropna(inplace=True)
df.reset_index(drop=True, inplace=True)
df.head(10)
```

#### 4. Preprocessing message: Explained in the section 4.4.2

#### 5. Tokenize

```

for word in words:
    word = self.preprocess_word(word)
    if self.is_valid_word(word):
        processed_message.append(word)
return ' '.join(processed_message)

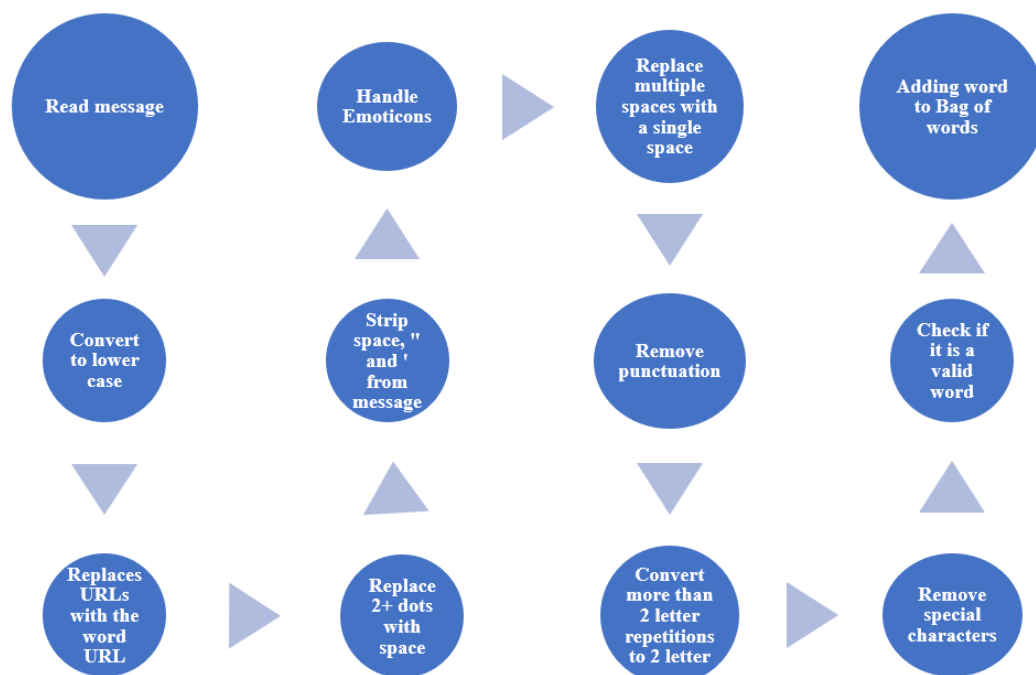
def transformCleanedMessage(self, message):
    tfv=TfidfVectorizer(min_df=0, max_features=None, strip_accents='unicode', lowercase=True,
    analyzer='word', token_pattern=r'\w{3,}', ngram_range=(1,1), sublinear_tf=True, stop_words = "english")
    transformedMessage=tfv.fit_transform(message)
    return transformedMessage

```

## 4.4.2. Preprocessor

Pre-processing refers to the transformations applied to our data before feeding it to the algorithm. Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. For achieving better results from the applied model in Machine Learning projects the format of the data has to be in a proper manner. Some specified Machine Learning model needs information in a specified format.

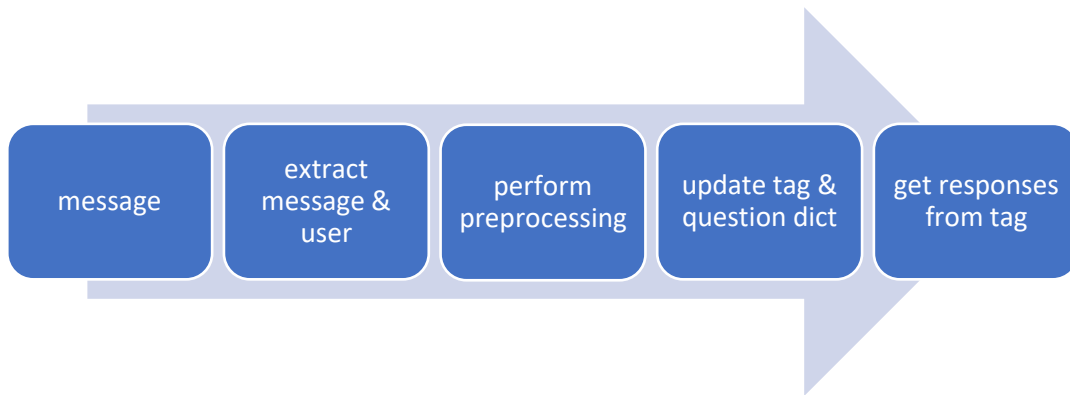
Flow created for preprocessor:





### 4.4.3. Logic Adaptors

Flow chart:



### 4.4.4. Stock Prediction

Flow 1:



```
def getLiveStockRate(self, stockCode):  
    df = self.ts.get_intraday(stockCode, interval='1min', outputsize='compact')  
    df = df[0]['4. close']  
    return df[len(df)-1]
```

 Sunil 10:27 PM

@InvestoBot tell me about google and apple



InvestoBot APP 10:28 PM

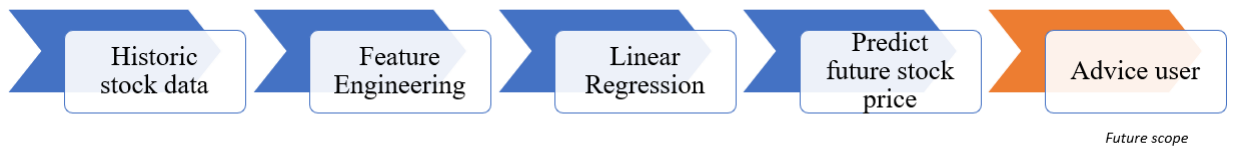
Awesome! Below is some information you maybe interested in. The live rate for Google's stock is \$1306.755. The live rate for Apple's stock is \$267.5. I hope this helps you and you create a portfolio with us. If you need to create a portfolio or know more about stock investment, kindly contact our office. Kindly type quit to exit or type Hello to continue chatting with me.

### Loading historic stock data

```
key = 'E4M6BVAOLPUG20L7'  
ts = TimeSeries(key, output_format='pandas')
```

```
data, googl_meta_data = ts.get_daily('GOOGL', outputsize='full')  
data = data.rename_axis(['Date'])  
data
```

## Flow 2:



### 1. Feature Engineering

```
def read_train():
    prices = pd.read_csv("trainData.csv")
    prices = clean(prices)
    return prices

def clean(prices):
    (m, n) = prices.shape

    prices = prices.loc[:, ['Date', 'Open', 'Close']]

    # Add new column "Close Price at the Previous Day" and "Open Price at the Previous Day"
    prevColumn = prices.loc[:, ['Open', 'Close']].assign(index = lambda x: x.index-1)
    prevColumn = prevColumn.set_index('index')
    prices = prices.join(prevColumn, rsuffix='Prev')

    # Add new column "Close Price at the Day Before" and "Open Price at the Day Before"
    prevColumn = prices.loc[:, ['Open', 'Close']].assign(index = lambda x: x.index-2)
    prevColumn = prevColumn.set_index('index')
    prices = prices.join(prevColumn, rsuffix='Prev2')

    prevColumn = prices.loc[:, ['Open', 'Close']].assign(index = lambda x: x.index-3)
    prevColumn = prevColumn.set_index('index')
    prices = prices.join(prevColumn, rsuffix='Prev3')

    prevColumn = prices.loc[:, ['Open', 'Close']].assign(index = lambda x: x.index-4)
    prevColumn = prevColumn.set_index('index')
    prices = prices.join(prevColumn, rsuffix='Prev4')

    prevColumn = prices.loc[:, ['Open', 'Close']].assign(index = lambda x: x.index-5)
    prevColumn = prevColumn.set_index('index')
    prices = prices.join(prevColumn, rsuffix='Prev5')
```

## 2. Linear Regression

```
def predict_and_plot(X, date):
    m, n = X.shape
    X_test = X.loc[np.floor(m*0.6)+1:]
    X_train = X.loc[:np.floor(m*0.6)]

    y_test = y.loc[np.floor(m*0.6)+1:]
    y_train = y.loc[:np.floor(m*0.6)]

    date_test = date.loc[np.floor(m*0.6)+1:]
    date_train = date.loc[:np.floor(m*0.6)]

    lr = LinearRegression()

    lr.fit(X_train, y_train)

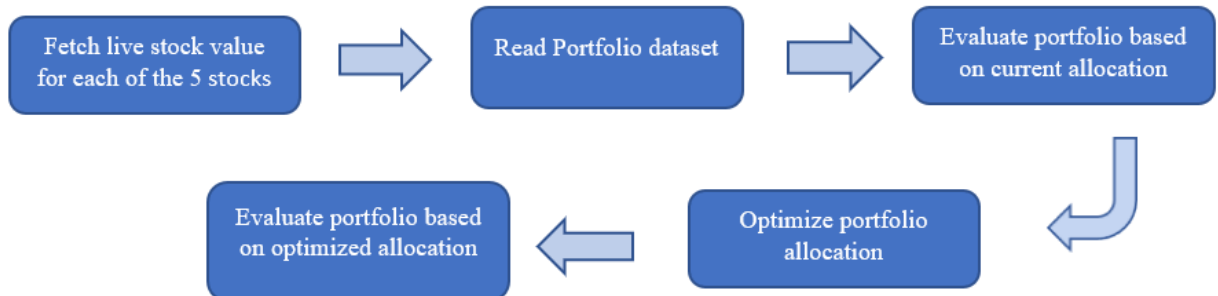
    print('Coefficients: \n', lr.coef_)
    print('Intercept: \n', lr.intercept_)
    pred = lr.predict(X_test)
    return lr, pred
```

## 3. Load the stock prices for the above dates and predict

```
prevOpen = data.loc[:,['Open']]
prevOpen = prevOpen.iloc[len(prevOpen)-1]
prevClose = close_val.item(len(close_val)-1)
prevOpen = prevOpen.values.item(0)
newX3 = X3
try:
    newX3.drop('Date2num', axis=1, inplace=True)
except:
    print('already dropped')
dummy = pd.DataFrame([[prevClose, prevOpen, prevClose]], columns=['Open', 'OpenPrev', 'ClosePrev'])
newX3 = newX3.append(dummy, ignore_index=True)
date = date.append(newDate, ignore_index=True)
for x in range(0, len(newDate)-1):
    print(len(newX3))
    print(len(date))
    lr, close_val = predict_and_plot(newX3, date)
    prevClose = close_val.item(len(close_val)-1)
    prevOpen = newX3.loc[:,['Open']]
    prevOpen = prevOpen.iloc[len(prevOpen)-1]
    prevOpen = prevOpen.values.item(0)
    dummy = pd.DataFrame([[prevClose, prevOpen, prevClose]], columns=['Open', 'OpenPrev', 'ClosePrev'])
    newX3 = newX3.append(dummy, ignore_index=True)

newX3
```

## 4.4.5. Portfolio Optimization



For fetching the current stock values, we have used the - Alpha Vantage APIs (Stock Time Series Data). The suite of APIs provides Realtime and historical global equity data in 4 different temporal resolutions: (1) daily, (2) weekly, (3) monthly, and (4) intraday. Daily, weekly, and monthly time series contain 20+ years of historical data.

- Importing libraries & fetching live stock values for each of the 5 stocks:

```
import pandas as pd
import numpy as np
from alpha_vantage.timeseries import TimeSeries
```

```
# Your key here
#key = 'enter_your_key_here'
ts = TimeSeries(key, output_format='pandas')
```

```
googl, googl_meta_data = ts.get_daily('GOOGL', outputsize='full')
fb, fb_meta_data = ts.get_daily('FB', outputsize='full')
msft, msft_meta_data = ts.get_daily('MSFT', outputsize='full')
aapl, aapl_meta_data = ts.get_daily('AAPL', outputsize='full')
twtr, twtr_meta_data = ts.get_daily('TWTR', outputsize='full')
```

- Function for getting stock allocation for the respective users:

```
def getStockAllocation(dfObj, account_number):
    stockAllocations = []
    currentValue = 0
    # Check if the element exists in dataframe values
    if account_number in dfObj.values:
        df_accounts = df.copy(deep=True)
        df_accounts = df_accounts[df_accounts['ACCOUNT_NO']==account_number]
        currentValue = df_accounts['TOTAL'].item()
        df_accounts.drop(['ACCOUNT_NO', 'NAME', 'TOTAL'], axis=1, inplace=True)
        for i, j in df_accounts.iterrows():
            stockAllocations = list(j.items())
    else:
        stockAllocations = [], currentValue
    # Returns a list of stocks and its current allocation
    return stockAllocations, currentValue
```

- Function to print the stock valuations:

```
def printStats(evaluation):
    cumulativeReturn = (evaluation[-1]/evaluation[0])-1

    dailyReturn = evaluation.copy()
    dailyReturn[1:] = (evaluation[1:]/evaluation[:-1].values)-1
    dailyReturn = dailyReturn[1:]

    dailyReturnAverage = dailyReturn.mean()

    dailyReturnStd = dailyReturn.std()

    sharpeRatio = dailyReturnAverage/dailyReturnStd
    sharpeRatio = np.sqrt(len(evaluation))*sharpeRatio

    portValue = evaluation.iloc[len(evaluation)-1]

    print("Cumulative Return is ", cumulativeReturn)
    print("Daily Return mean is ", dailyReturnAverage)
    print("Volatility or Standard Deviation is ", dailyReturnStd)
    print("Sharpe Ratio is ", sharpeRatio)
    print("Portfolio evaluation at the end of the term is ", portValue)

    return cumulativeReturn, dailyReturn, dailyReturnAverage, dailyReturnStd, sharpeRatio, portValue
cumulativeReturn, dailyReturn, dailyReturnAverage, dailyReturnStd, sharpeRatio, portValue = printStats(evaluation)
```

## 4.5. Web Services

### 4.5.1. Flask

Flask is a micro web framework written in Python. It provides us with tools, libraries and technologies that allows us to build a web application. This web application can be some web pages, a blog, a wiki or go as big as a web-based calendar application or a commercial website. Flask is part of the categories of the micro-framework. Micro-framework is normally framework with little to no dependencies to external libraries. This has pros and cons. Pros would be that the framework is light, there are little dependency to update and watch for security bugs, cons is that some time you will have to do more work by yourself or increase yourself the list of dependencies by adding plugins.

#### 1. Installing flask

Flask is microframework for python. The “micro” in microframework means Flask aims to keep the core simple but extensible. Installing flask with the following command: `$ pip install Flask`

#### 2. Create REST API in flask

```
from flask import Flask
```

This line asks the application to import Flask module from flask package. Flask used to create instances of web application.

```
app = Flask(__name__)
```

This line creates an instance of your web application. `__name__` is a special variable in python, it will equal to “`__main__`” if the module (python file) being executed as the main program.

```
@app.route("/")
```

This line defines the routes. For example, if we set route to “/” like above, the code will be executed if we access localhost:5000/. This line define function that will be executed if we access route.

```
if __name__ == '__main__':  
    app.run(debug=True)
```

This line mean that your flask app will being run if we run from app.py.

## **4.5.2. Slack**

A bot is a type of Slack App designed to interact with users via conversation. A bot is the same as a regular app: it can access the same range of APIs and do all of the magical things that a Slack App can do. But when you build a bot for your Slack App, you're giving that app a face, a name, and a personality, and encouraging users to talk to it. Your bot can send DMs, it can be mentioned by users, it can post messages or upload files, and it can be invited to channels - or kicked out.

### **1. Creating a bot user**

Heading to your app's settings page and click the Bot Users feature in the navigation menu. System presents with a button marked Add a Bot User, and when you click on it, you'll see a screen where you can configure your app's bot user with name, username and status.

### **2. Setting up the Events API**

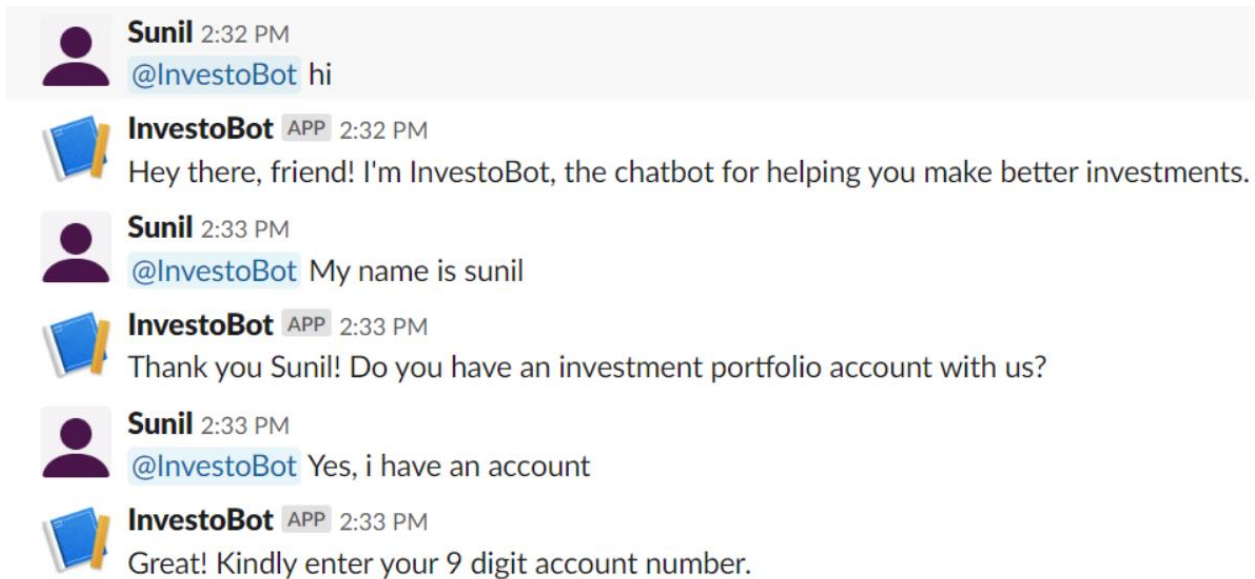
The Events API is a bot's equivalent of eyes and ears. It gives a bot a way to react to posted messages, changes to channels, and other activities that happen in Slack. When these events happen, a data payload will be sent to your bot, and it can use that data to form a useful response.

### **3. Installing the bot to a workspace**

A bot user is added to a workspace by installing the app that the bot is associated with. Once you do, you'll get a bot token that is imbued with the bot scope. This token can be used with a subset of Web API methods that we'll discuss later. If you had already installed your app in the past, you'll need to reinstall to grant the additional bot scope.

### **4. Handling events**

There are 4 events triggered in this conversation: the first is an 'app\_mention' event from the first message that mentions the bot; the next three are message events for each of the messages posted by Johnny. The bot will need to be able to interpret each event and respond accordingly.



## 5. Receiving events

Each event will trigger a request, containing a JSON payload, sent to your configured Request URL. The Events API docs contain a full description of the shape of this JSON, and the reference for `app_mention` and the `message.channels` contain any details specific to each event type.

## 6. Responding to mentions using the Web API

For a bot, being mentioned is usually the triggering event for a conversation, just as a human will respond when they hear their name. The app code should use the `type` field inside the event payload to spot these `app_mention` events, and differentiate them from any other events it might receive.



## 4.6. Results

### 4.6.1. Model

We have implemented different algorithms and compared their results. While evaluating and comparing the across different parameters, we chose Artificial Neural Network for implementation.

The comparison results are below:

#### 1. Compare Algorithms

Sr. No	Algorithms	Accuracy (%)
1	Logistic Regression	76.87
2	XGBoost	85.03
3	Random Forest	94.56
4	Artificial Neural Network	99.89

#### 2. Model Evaluation and Matrix Parameters

Sr. No	Measures	Formula
1	Sensitivity (TPR)	$TP/(TP+FN)$
2	Specificity (TNR)	$TN/(TN+FP)$
3	Precision (PPV)	$TP/(TP+FP)$
4	Negative Predictive Value (NPV)	$TN/(TN+FN)$
5	Fall out / False positive Rate (FPR)	$FP/(FP+TN)$
6	False negative rate (FPR)	$FN/(TP+FN)$
7	False discovery rate (FDR)	$FP/(TP+FP)$
8	Overall Accuracy	$(TP+TN)/(TP+FP+FN+TN)$

### 3. Evaluation of Neural Network Results

	accountNumberEntered	bondsInvestment	charityInvestment	educationInvestment	exit	foreignExchangeInvestment
Sensitivity/Recall/TPR	1	1	1	1	1	1
Specificity/TNR	1	1	1	1	1	1
Precision/PPV	1	1	1	1	1	1
NegativePredictiveValue	1	1	1	1	1	1
FallOut/FPR	0	0	0	0	0	0
FalseNegativeRate	0	0	0	0	0	0
FalseDiscoveryRate	0	0	0	0	0	0
Accuracy	1	1	1	1	1	1

Similar results are obtained for other algorithms, of which, Neural Network proved to be the best.

## 4.6.2. Chatbot

Below are the snippets which shows the interaction of InvestoBot with the User communicating over Slack.

**Sunil** 6:22 PM  
@InvestoBot Hi

**InvestoBot** APP 6:22 PM  
Hey there, friend! I'm InvestoBot, the chatbot for helping you make better investments. Before we proceed ahead, may I know your name?

**Sunil** 6:23 PM  
@InvestoBot my name is sunil

**InvestoBot** APP 6:23 PM  
Thank you Sunil! Do you have an investment portfolio account with us?

**Sunil** 6:23 PM  
@InvestoBot yes, i have an account

**InvestoBot** APP 6:23 PM  
Great! Kindly enter your 9 digit account number.

**Sunil** 6:26 PM  
@InvestoBot my account number is 100000001

**InvestoBot** APP 6:26 PM  
Thank you! The current portfolio allocations are as follows: 30.0% stocks of Microsoft, 30.0% stocks of Twitter, 40.0% stocks of Apple. Would you like to optimize it for higher gains?

The current portfolio allocations are as follows: 30.0% stocks of Microsoft, 30.0% stocks of Twitter, 40.0% stocks of Apple. Based on these allocations, the current portfolio returns are as follows. Cumulative Return is \$0.447902400834006. Daily Return mean is \$0.0017242381616811347. Volatility or Standard Deviation is 0.015089540976086656. Sharpe Ratio is 1.736709636592353. Portfolio evaluation at the end of the term is \$1478.988865379912.. However, on optimizing it, the new portfolio returns are as follows. Cumulative Return is \$0.5780973734146857. Daily Return mean is \$0.002071668363435927. Volatility or Standard Deviation is 0.013128489666495225. Sharpe Ratio is 2.3983439014438397. Portfolio evaluation at the end of the term is \$1611.9791240218997.. These new returns can be achieved if portfolio allocations change to the following: 50.824289452125335% stocks of Microsoft, 7.2590337623876735% stocks of Facebook, 41.91667678548702% stocks of Apple. The optimized portfolio is \$132.99025864198757 higher than original portfolio. Kindly drop by our office, if you want to update your portfolio allocations. Kindly type quit to exit or type Hello to continue chatting with me.

**Sunil** 6:30 PM  
@InvestoBot i need your help with investments

**InvestoBot** APP 6:30 PM  
Awesome! Here are some types of investments. I can help you make investments in 4 categories - Personal, Education, Retirement and Charity. Please enter the type of investment to get more information about it.

**Sunil** 6:31 PM  
@InvestoBot personal

**InvestoBot** APP 6:31 PM  
I can help you make personal investments in 4 categories - Stocks, Mutual Funds, Bonds and Foreign Exchange. Please enter the type of personal investment to get more information about it.



**Sunil** 7:27 PM

@InvestoBot Sure, tell me something about Apple



**InvestoBot** APP 7:27 PM

Awesome! Below is some information you maybe interested in. The live rate for Apple's stock is \$267.5. I hope this helps you and you create a portfolio with us. If you need to create a portfolio or know more about stock investment, kindly contact our office. Kindly type quit to exit or type Hello to continue chatting with me.



**Sunil** 7:28 PM

@InvestoBot Sure, tell me something about google



**InvestoBot** APP 7:28 PM

Awesome! Below is some information you maybe interested in. The live rate for Google's stock is \$1307.45. I hope this helps you and you create a portfolio with us. If you need to create a portfolio or know more about stock investment, kindly contact our office. Kindly type quit to exit or type Hello to continue chatting with me.



**Sunil** 7:29 PM

@InvestoBot tell me more about TWTR



**InvestoBot** APP 7:29 PM

Awesome! Below is some information you maybe interested in. The live rate for Twitter's stock is \$30.92. I hope this helps you and you create a portfolio with us. If you need to create a portfolio or know more about stock investment, kindly contact our office. Kindly type quit to exit or type Hello to continue chatting with me.

## **5. Future Scope**

Chatbots are hot software in the enterprise, and to maintain longevity and relevance, we will need to look at the barriers to entry, interface options and NLP issues so that we can better assist customer service experiences and response time. We can focus on developing our bot to become smarter at aiding in wide varieties of things and target large group of users by focusing on the below things.

1. Multi-language support
2. Support for more stocks
3. Improving Natural Language Understanding (NLU) framework
4. Live assistance with other investment services i.e. Retirement, Education, Charity, Personal, etc. investments
5. Integrating news APIs to improve future stock price predictions and make appropriate suggestions

## **6. Conclusion**

From my perspective, chatbots or smart assistants with artificial intelligence are dramatically changing businesses all over the world. Currently there are a wide range of chatbot building platforms that are available for various enterprises, such as e-commerce, retail, banking, leisure, travel, healthcare, and so on. Chatbots can reach out to a large audience on messaging apps and be more effective than humans. They may develop into a capable information-gathering tool in the near future. The job for businesses and brands after a certain point is to take the next leap and move forward. For AI and chatbot, the future is coming one way or another, which can't be avoided.

## 7. References

- <https://www.quandl.com/tools/python>
- <http://docs.tweepy.org/en/latest/api.html>
- <https://dialogflow.com/>
- <https://pythonprogramming.net/chatbot-deep-learning-python-tensorflow/>
- <https://hackernoon.com/deep-learning-chatbot-everything-you-need-to-know-r11jm30bc>
- [https://www.youtube.com/watch?v=jBXAi-Vm\\_-g&list=PLzMcbGfZo4-ndH9FoC4YWHGXG5RZekt-Q&index=5](https://www.youtube.com/watch?v=jBXAi-Vm_-g&list=PLzMcbGfZo4-ndH9FoC4YWHGXG5RZekt-Q&index=5)
- <https://medium.com/analytics-vidhya/building-a-simple-chatbot-in-python-using-nltk-7c8c8215ac6e>
- <https://chatbotslife.com/how-to-create-an-intelligent-chatbot-in-python-c655eb39d6b1>
- <https://towardsdatascience.com/natural-language-processing-feature-engineering-using-tf-idf-e8b9d00e7e76>
- <https://www.datacamp.com/community/tutorials/stemming-lemmatization-python>
- <https://www.digitaltrends.com/cool-tech/what-is-an-artificial-neural-network/>
- <https://www.geeksforgeeks.org/nlp-how-tokenizing-text-sentence-words-works/>
- [https://en.wikipedia.org/wiki/Bag-of-words\\_model](https://en.wikipedia.org/wiki/Bag-of-words_model)
- <https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa>