**Machine Learning Engineer Nanodegree**

Capstone Project
Payal Raj Chaudhary
May 23rd, 2018

## Definition :

### Project Overview

Time series modeling research has grabbed an attention of many researchers over past few decades. The time series is useful in collecting the required information and studying the past observations which can help in learning the structure of time series. The model should be capable of learning how past events affect current predictions i.e. To make forecasts. In other words time series termed as an act of predicting the future from the past behavior. Since it has a capability of learning from the past time series data it is useful in many practical fields such as business, economics, finance, science and engineering biological data, weather forecasting. Different approaches have been made over the past few years  in order to solve the time-series problem for predictions the three mostly used approaches are, viz. The stochastic, neural networks and SVM approach. We will be using Time Series for predicting the web traffic for Wikipedia articles.

### Problem Statement

This problem statement was part of kaggle competition.

https://www.kaggle.com/c/web-traffic-time-series-forecasting

Web traffic generally refers to the amount of data send and receive from the visitors, it is usually determined by a number of users visited and number of pages they have opened. It can tell you many more things what is the preference of most of the users which is mostly searched page moreover it can be distinguished based on country also. The preference of the page of the user of a particular country. There are a lot of things which we can do if we prediction using the web traffic.

This project focuses on the problem of forecasting the future values of multiple time Series, as it has always been one of the most challenging problems in the field. The aim is to forecast the future of web traffic for Wikipedia articles.

For solving the above mentioned problem statement I will use RNN model LSTM and GRU in order to make predictions.

**Metrics**

Earlier we have proposed to use SMAPE for evaluation but after implementation SMAPE was not fit for our model. It's value is always zero when predicted value is equal to true value. Moreover for the value which are above true value the function is not convex which often leads to local minima. In order to solve this issue I switched to MAE

**MAE Mean Absolute Error:**

**Mean Absolute Error (MAE):** MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j|$$

All the problem which are stated above where solved we tried to implement using MAE. Moreover it simplify the computation and gives an intuitive interpretation for evaluation of time series prediction.So, we decided to move ahead with MAE as our evaluation metrics.

# Analysis :

**Data Exploration**

The training data set contains approximately 145k time series. Every time series represents the number of views of a Wikipedia article on daily basis, starting from 1st, July, 2015  till 1st, September, 2017

Splitting of the dataset

The time series model has a problem with a look ahead bias. In order to solve it, I will use walk forward split between the data sets. Validation helps in the tuning of the model and gives an estimate of model skill. The data from 1st-July-2015 till 31st-Dec-2016 will be used for model tuning, will be splitting the data into training and validation using walk forward split. Though we have termed it as splitting, we are actually using full data set for training as well as validation but in different time frames. Once satisfied with the model tuning, the final model will be trained without validation.

Training dataset - The data from 1st-July-2015 till 31st-Dec-2016 will be used for training.

Testing dataset- The data from 1st-Jan-2017 till 1st-Sep-2017 will be used for testing our predictions.

For each time series, the name of the article along with the traffic type, the time series represent (all, mobile, desktop) is provided. The data set does not distinguish between traffic values of zero and missing values. A missing value can be taken as zero traffic or unavailability of data for that particular day.

File descriptions

*Final_Traffic_Data*.csv* - Information about the traffic data is there in this CSV file, whereas each row represents a detail of the particular article and each column contains the entry of a particular date Some entries are not present or blank. The page names have data on Wikipedia project, and access type, and the agent type. In other words, each article name has the following format; name_project_access_agent; (e.g. 'AKB48_zh.wikipedia.org_all-access_spider').

*Final_Google_Trends_Tain*.csv* - It contains training data of the titles which are there in Final_Traffic_Data*.csv, which is extracted from google trend using pytrend.

Google_trends_Test_Final - It contains testing data of the titles which are there in Final_Traffic_Data*.csv, which is extracted from google trend using pytrend library.

Feature Distribution :

*Google Trend* - Most of the searches are usually done through google so, our main feature is Google Trend which we have extracted using pytrend.

*Train_hits* - It refers to number of views on that particular page in training time period.

*Test_hits* - It refers to number of views on that particular page in testing time period

*Page popularity* - This feature is the median of the page views.

*Agent,site,titles* - These features are extracted from the url and encoded. All these feature are extracted by using regular expression.

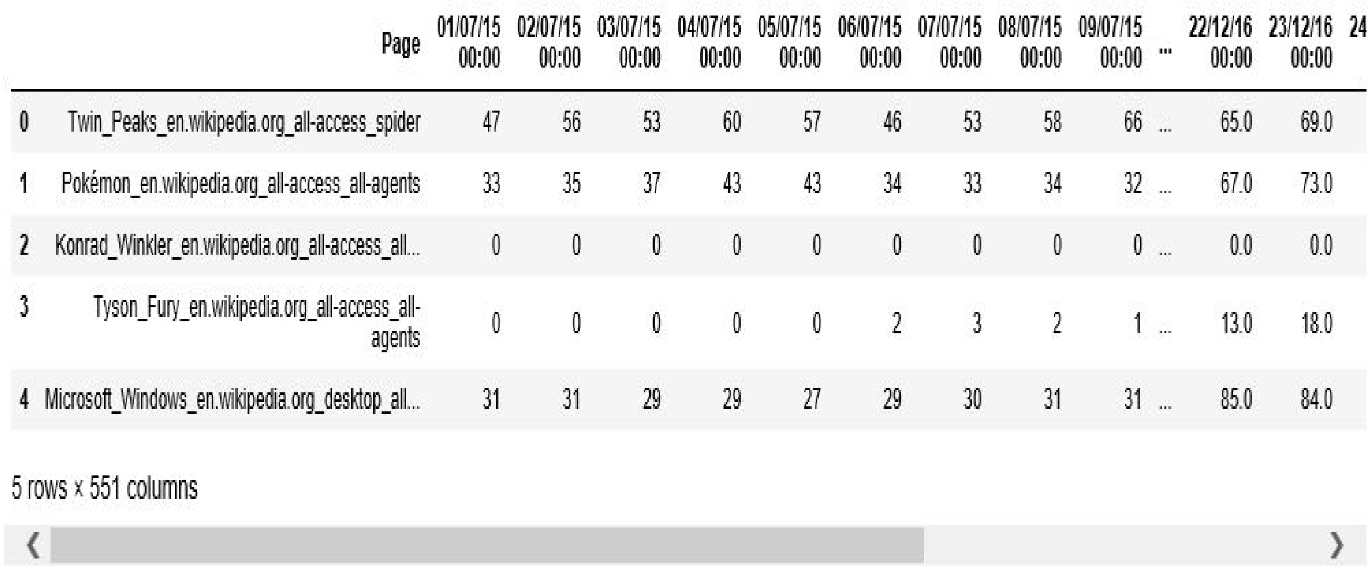
**Exploratory Visualization**

One of the most important and difficult task was to get data from google trend for all the articles. Google Trend API is not available publically we can’t extract bulk data. It was very challenging to extract trend data.

We have around 145k wikipedia article from those articles we need to check for the articles for which the google trend data is available. Since we have considered google trend to be our main feature. We have selected the articles only in English language.

In some cases the google trend data was not available so we have not included that article as well.

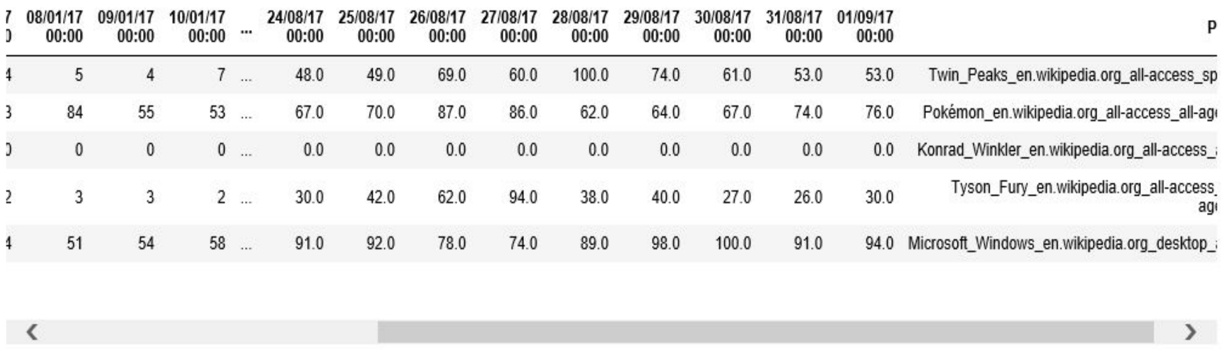
Below we have included some graphs to the visualization of the data. Further we have added the graph to compare the visualization of data provided by Kaggle and data provided by google trend.



| | Page | 01/07/15 00:00 | 02/07/15 00:00 | 03/07/15 00:00 | 04/07/15 00:00 | 05/07/15 00:00 | 06/07/15 00:00 | 07/07/15 00:00 | 08/07/15 00:00 | 09/07/15 00:00 | ... | 22/12/16 00:00 | 23/12/16 00:00 | 24 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Twin_Peaks_en.wikipedia.org_all-access_spider | 47 | 56 | 53 | 60 | 57 | 46 | 53 | 58 | 66 | ... | 65.0 | 69.0 | |
| 1 | Pokémon_en.wikipedia.org_all-access_all-agents | 33 | 35 | 37 | 43 | 43 | 34 | 33 | 34 | 32 | ... | 67.0 | 73.0 | |
| 2 | Konrad_Winkler_en.wikipedia.org_all-access_all... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0.0 | 0.0 | |
| 3 | Tyson_Fury_en.wikipedia.org_all-access_all-agents | 0 | 0 | 0 | 0 | 0 | 2 | 3 | 2 | 1 | ... | 13.0 | 18.0 | |
| 4 | Microsoft_Windows_en.wikipedia.org_desktop_all... | 31 | 31 | 29 | 29 | 27 | 29 | 30 | 31 | 31 | ... | 85.0 | 84.0 | |

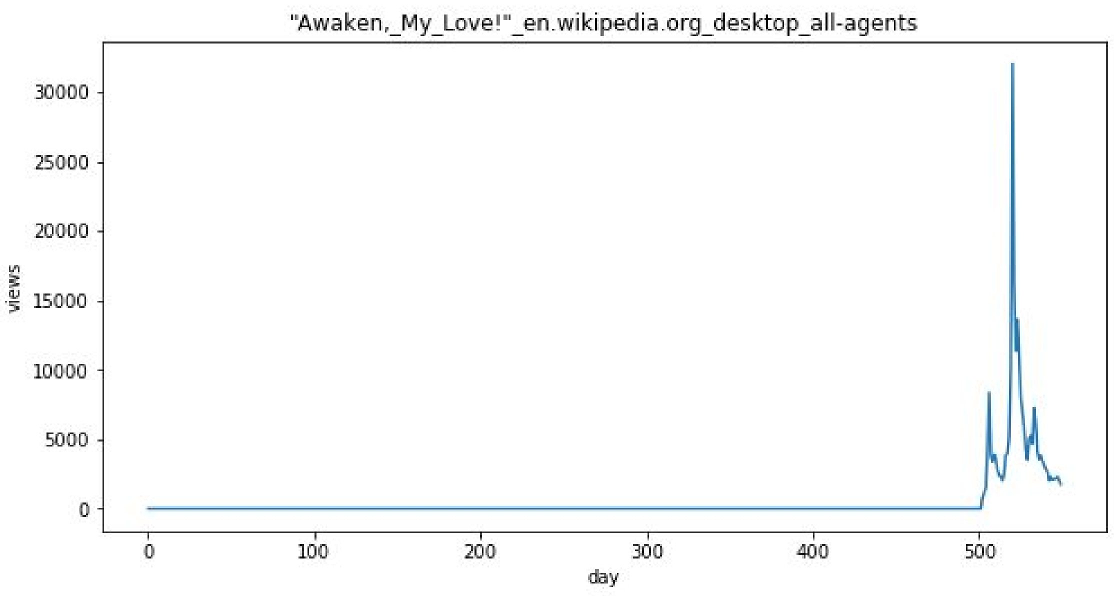
5 rows × 551 columns

Google trend data for training of some pages.

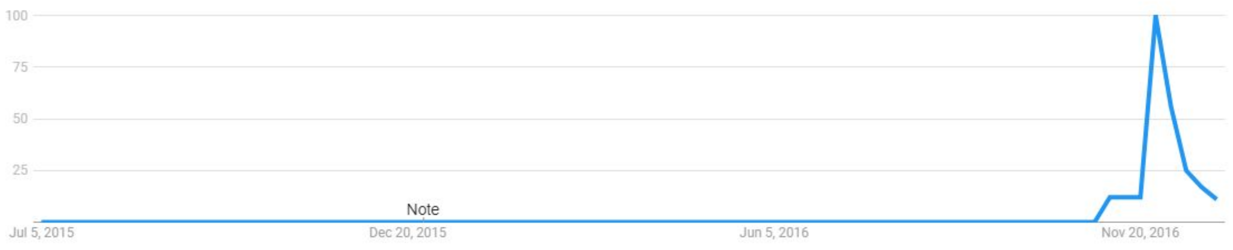


| 7 0 | 08/01/17 00:00 | 09/01/17 00:00 | 10/01/17 00:00 | ... | 24/08/17 00:00 | 25/08/17 00:00 | 26/08/17 00:00 | 27/08/17 00:00 | 28/08/17 00:00 | 29/08/17 00:00 | 30/08/17 00:00 | 31/08/17 00:00 | 01/09/17 00:00 | P |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 4 | 5 | 4 | 7 | ... | 48.0 | 49.0 | 69.0 | 60.0 | 100.0 | 74.0 | 61.0 | 53.0 | 53.0 | Twin_Peaks_en.wikipedia.org_all-access_sp |
| 3 | 84 | 55 | 53 | ... | 67.0 | 70.0 | 87.0 | 86.0 | 62.0 | 64.0 | 67.0 | 74.0 | 76.0 | Pokémon_en.wikipedia.org_all-access_all-ag |
| 0 | 0 | 0 | 0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | Konrad_Winkler_en.wikipedia.org_all-access_ |
| 2 | 3 | 3 | 2 | ... | 30.0 | 42.0 | 62.0 | 94.0 | 38.0 | 40.0 | 27.0 | 26.0 | 30.0 | Tyson_Fury_en.wikipedia.org_all-access_ ag |
| 4 | 51 | 54 | 58 | ... | 91.0 | 92.0 | 78.0 | 74.0 | 89.0 | 98.0 | 100.0 | 91.0 | 94.0 | Microsoft_Windows_en.wikipedia.org_desktop_ |

Google trend data for testing for some pages



Analysis from kaggle dataset



Analysis from Google Trend dataset

In the above graph we can see that the page hit was maximum at end same was the case in the google trend graph. So, considering the google trend data to be the main feature is completely justified in the graph. Further it also shows the duration when the page was most popular.

## Algorithms and Techniques

For the above-mentioned problem statement, we will be using RNN (Recurrent Neural Network) for prediction. RNN can be considered as a more advanced version of ARIMA Models, but in RNN any alteration or modification can be done easily. RNN is non-parametric its structure is completely determined by the data and simplifies learning. Unlike ARIMA it is easy to do prediction with RNN for parameters for 145K time series. Any feature like (numerical or categorical, time or series-dependent) can be easily passed into the model. seq2seq seems good for this task: it has a tendency to predict next values, conditioning on joint probability of previous values, including our past predictions.

Long Short-Term Memory (LSTM) has been used for the prediction of future traffic. LSTMs and conventional RNNs are most commonly used for the sequence prediction and time series predictions as they give the better result than any other model. The algorithm works with LSTM (Long Short-Term Memory) neuron cells. LSTM neurons are little-complicated cells than standard neurons, as they can be seen as a sequence of neurons, containing 3 gates namely (input, output and forget gate). Memorization is achieved through the memory cell, which helps to remember the required information and doesn’t allow the full information from the hidden cell to pass on. There is an internal recursion loop, but also the whole cell can have external recursion as well.

Whereas in GRU there are only 2 gates (reset and update gate). The GRU memory unit does not use it pass on the fully hidden content without any control or modification. It can be better than LSTM.

## Benchmark

The model which we have considered as a benchmark was also implemented using RNN and GRU, it got first prize in the kaggle competition on this problem statement. The features used in this model are pageviews; age, country, site; day of the week; year to year autocorrelation; quarter to quarter autocorrelation; page popularity; lagged pageviews. Google trends data were not used in the predictions in the benchmark model.

The metrics for this project are evaluated on SMAPE between forecasts and actual values. Symmetric mean absolute percentage error (SMAPE) is an accuracy measure based on percentage (or relative) errors. SMAPE for the benchmark model was 35.48.

## Methodology

### Data Preprocessing

Data Preprocessing refers to preparing your data which can be used for training and testing. We need to clean our data.

1. We need to fill all the null value by assigning zero for that fillna() function was used.
2. Now we need to extract google trend data for the given articles as the data set contains 145k article with multiple language Considering the fact that google trend is one of the main feature, to extract google trend data for language other than English is difficult so we have used only English language.
3. We need to check for the availability of google trend data for the remaining articles.
4. Pytrend library is available in python, which can be used to extract data from google trend.

Now that we have clean our data, we can split dataset into training validation and testing.
The testing dataset should not be same as training data set.

### Implementation

We have implemented our model using Keras.It is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

Now that our data set is ready we need to extract feature from our data which can be used for training and testing. Our result solely depends upon the feature and the model we use. Hence feature extraction is one of the important task.

*Google Trend* - Most of the searches are usually done through  google so, our main feature is Google Trend which we have extracted using pytrend.

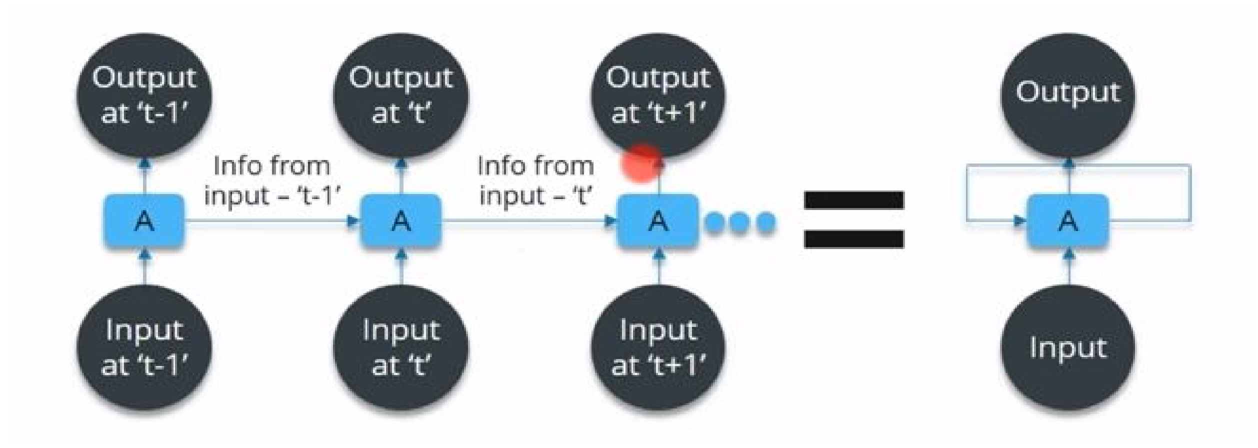
*Train_hits* - It refers to number of views on that particular page in training time period.

*Test_hits* - It refers to number of views on that particular page in testing time period

*Page popularity* - This feature is the median of the page views.

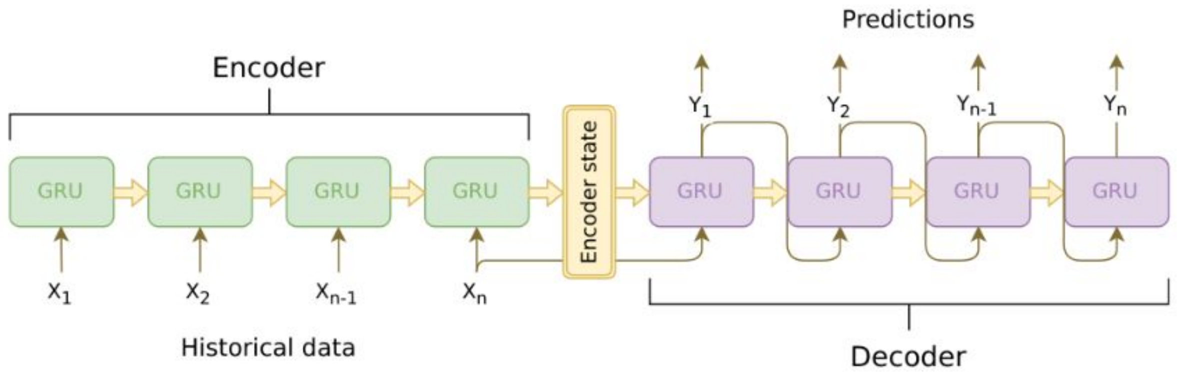
*Agent,site,titles* - These features are extracted from the url and encoded. All these feature are extracted by using regular expression.

After feature extraction we need to pass that feature into our model in order to train. We have used sequence to sequence which consist of two main block a) an Encoder and b) a decoder. These block can be implemented using LSTM and GRU. LSTM has has capability of learning from past and make a prediction.



RNN (LSTM)

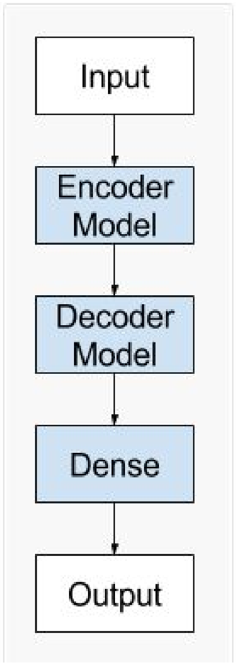
Encoder and decoder :



Model Core

Encoder - The main function of encoder is to take input in our case it refers to the feature and encodes it into the fixed representation that our decoder can use that as a context. We have used two layer of encode in our model.

Decoder - The main function of decoder is to decode the information passed by the encoder and generate the output sequences from that. The decoder should generate output of same length as that of encoder.We have used 2 layer of decoder in our model.



Encoder-Decoder Layer in GRU

Keras has predefined layer for GRU. And for validation 20% of the dataset is used.

Some challenges during coding even for 1 error it was taking lot of time to correct and check as every time it was reading the whole data since we have around 12k of data it was very time time consuming in order to resolve this issue and speed up the process we have cached our data. Then fetching google trend day it took around 6 days to fetch data from google for 12k article as it has daily limit problem. So we have to change IP every time in order to fetch data.

## Refinements

As mentioned in the Evaluation Metrics SMAPE was supposed to be the metric for evaluating the model but later I found that SMAPE was not the good fit for our model.

It was very difficult to optimize SMAPE for following reasons:

1. It give error as 0 when the predictive value is equal to true value, ideally that should not be the case.

2. Whenever the value is above true value the function is not convex because of which instead of giving the global minima it ends up giving local minima.

3. Further it is scale dependent SMAPE doesn’t give good result for the small scale data.

Because of the above mentioned issue we decided to go with MAE where these kind of issue can be solved which can increase our model accuracy as well.

Hyperparameter Tuning :

Initially we tried with 1 layer of encoder and decoder since we have large amount of data the 1 layer was enough for learning data and all input feature. Later we have implemented using 2 layer of encoder and decoder.

Large unit were taken in GRU the model was giving good accuracy while training but same was not the case while testing. It was overfitting so 512 unit was considered as accuracy was good while testing. And further we have changed our dropout also the final value 0.2 was taken for dropout.

## Results

### Model Evaluation and Validation

The final architecture and the hyperparameter used in the model is listed below :

1. The shape of the encoder input is 186, 5
2. There are 2 layers of encoder and both layers are encoded using GRU
3. In both the encoder layer the GRU unit is 512, unit refers to dimensionality of output space
4. In both encoder layer the activation function used is selu.
5. After encoder there are two layers of decoder again for that GRU is used.
6. In both decoder layer the GRU unit is 512
7. Activation function used for the decoder layer is selu.

8. Dense layer unit is 67
9. Number of epoch for our model is 10.

To verify the accuracy of the model I have done validation of data the validation is set to 0.2 i.e 20 % of the data.

Talking about the robustness our model is good in that as well, any minor fluctuation won't result in drastic change in the model because the RNN internal structure has capability of learning from past. Let's take an example one page has view average view of 10 per day but suddenly the view is increased to 100 for certain period of time, our model will do the prediction considering the data of 10 view per and 100 view per day it won't change drastically when there is 100 views. The current result is not based on the current input instead is based on the current input and learning from the past input.

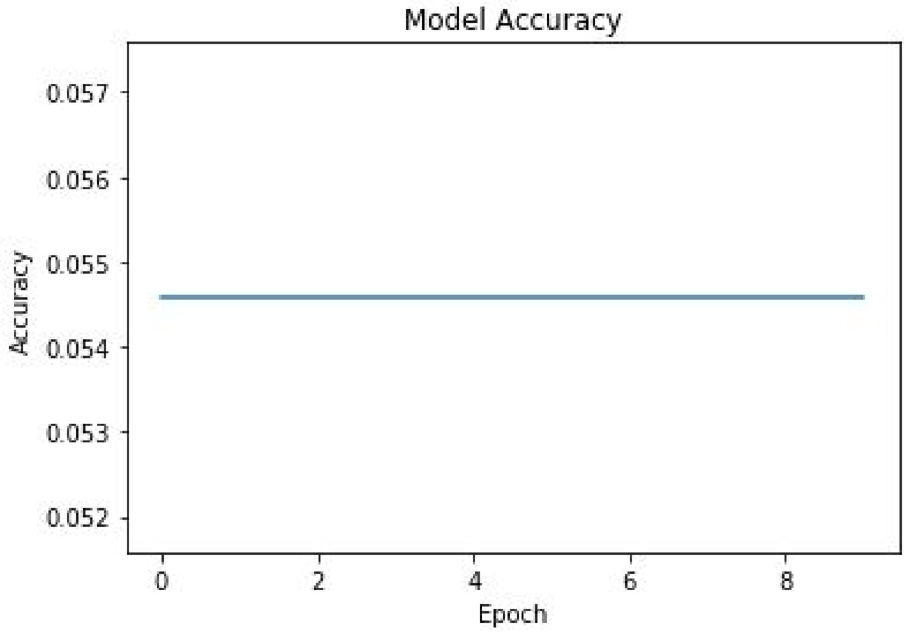
And regarding testing model on unseen data, it was difficult as we have already used 12k articles. Getting google trend data for that itself took 6 days. So getting more data and collecting google trend data for that would consume lot of time.

**Justification**

The accuracy of the model is 57 % and the error is 43. Our benchmark model error was SMAPE 35.

## Conclusion

We are able to achieve accuracy of 55% . The MAE (mean absolute error ) is 43. We have added google trend data as one of our main feature.



Graphical Visualization

In the above graph y label represents the accuracy and x label represents the number of epoch.The graph shows the accuracy of one dataset which is run for 10 epoch and the every epoch the accuracy of 54.7 % was achieved.

**Reflection**

The overall view of the project can be given using the following steps:

1. The problem statement and the public dataset which are relevant were found.
2. The Google Trend data was extracted using pytrend
3. The Data was preprocessed from that the final dataset was created.
4. The important features were extracted from the final dataset.
5. Data validation was done before passing into the model.
6. Model was made usings Keras highly interactive API on the top of tensorflow.
7. Two layers Encoder and decoder unit were implemented using GRU.
8. All the extracted features along with google trend data was passed into the model and training was initiated.
9. Validation is 20% of the entire dataset
10. Multiple training was done in order to get accurate prediction.

I found that step 2 and 4 was very challenging as google Trend API is not public and extracting data from Google Trend in bulk was very difficult task as they have set daily limit. We need to change IP every time to extract overcome daily limit problem. Selection of feature was equally challenging.

**Improvement:**

Our model is giving accuracy of 57% we will try to improve the accuracy of model. We can add more number of encoder and decoder layer and more randomization of data while training. Since we are large amount of dataset it takes lot of time for computation that too when are are training using CPU. The model is capable of predicting the future for web traffic for the articles which are in English language. Further we will try to improve our model and do the prediction for the article in different language.

## References:

1. https://arxiv.org/ftp/arxiv/papers/1302/1302.6613.pdf

2. https://www.kaggle.com/c/web-traffic-time-series-forecasting

3. https://trends.google.com/trends/

4. https://en.wikipedia.org/wiki/Mean_absolute_error

5. https://arxiv.org/pdf/1705.05690.pdf

6. https://github.com/GeneralMills/pytrends

7. https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

8.https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-netwo
rks/