capstone_report_update1

June 24, 2019

1 Machine Learning Engineer Nanodegree

1.1 Web Traffic Time Series Forecasting

Lalit Yadav June 24, 2019

1.2 I. Definition

1.2.1 Project Overview

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. More specifically, we aim to forecast future web traffic for approximately 145,000 Wikipedia articles.

Time Series forecasting is an important area of Machine Learning. It is important because there are so many prediction problems that involve a time component. However, while the time component adds additional information, it also makes time series problems more difficult to handle compared to many other prediction tasks. Deep Learning has plenty of applications in the world of statistical analysis. One of the areas with an environment for its applicability is the time series.

Deep Learning methods offers a lot of promise for Time Series forecasting, such as the automatic learning of temporal dependence and the automatic handling of temporal structures like trends and seasonality. With their quality of extracting patterns from the input data for long durations, they have the perfect applicability in forecasting. They can, therefore, deal with large amounts of data, multiple, complex variables and multi-step actions, which is what time series forecasting demands.

Related Research:-

https://arxiv.org/pdf/1703.07015.pdf

1.2.2 Problem Statement

This project is about predicting the future behaviour of time series' that describe the web traffic for Wikipedia articles. The data contains about 145k time series and comes in two separate files: train_2.csv holds the traffic data, where each column is a date and each row is an article, and key_2.csv contains a mapping between page names and a unique ID column (to be used in the submission file).

Each of these time series represent a number of daily views of a different Wikipedia article, starting from July, 1st, 2015 up until December 31st, 2016. The leaderboard during the training stage is based on traffic from January, 1st, 2017 up until September 10th, 2017. The goal is to forecast the daily views between September 13th, 2017 and November 13th, 2017 (64 data points) for each article in the dataset.

Following are needed task for development of algorithm: - First step would be to import the training data, then analyzing and cleanning the data. - We added some features that helped the model in predictions. Features like language of text, country, agent and access were extracted from the provided page information. - We will create a encoder decoder model to fit the data. One conern that I can see is that on longer sequences LSTM/GRU works, but can gradually forget information from the oldest items, it is also slower to train for larger dataset. We will try using Wavenet models. - As the total datasize is around 145k, we took some subsample(20-40k) of the original data and train the model. Once the model seems to generalize well on small subset of data we tried to use all data for prediction.

1.2.3 Metrics

SMAPE (target loss for competition) can't be used directly, because of unstable behavior near zero values (loss is a step function if truth value is zero, and not defined, if predicted value is also zero).

I used MAE loss on log1p(data), it's smooth almost everywhere and close enough to SMAPE for training purposes.

mean absolute error (MAE) is a measure of difference between two continuous variables. Assume X and Y are variables of paired observations that express the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement. Consider a scatter plot of n points, where point i has coordinates (xi, yi)... Mean Absolute Error (MAE) is the average vertical distance between each point and the identity line. MAE is also the average horizontal distance between each point and the identity line.

The Mean Absolute Error is given by:

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

1.3 II. Analysis

1.3.1 Data Exploration

- This challenge is about predicting the future behaviour of time series' that describe the web traffic for Wikipedia articles. The data contains about 145k time series
- The train_1.csv file has data from `2015-07-01` to `2016-12-31`
- The train_2.csv file has data from `2015-07-01` to `2017-09-10`
- We will be using data from train_2.csv for our training. Which had all data from train_1.csv and additional record. It has records from `July, 1st, 2015` to `September 1st, 2017`
- We have to predict daily page views between `September 13th, 2017` to `November 13th, 2017`.
- key_*.csv gives the mapping between the page names and the shortened Id column used for prediction > Download input files from https://www.kaggle.com/c/web-traffic-time-series-forecasting/data

- For this project we will be using data train_2.csv file which has page vistis for each page from `1st July, 2015` to `10th September, 2017`

```
In [3]: train_all.head()
```

```
Out[3]:                                                Page  2015-07-01  2015-07-02  \
        0            2NE1_zh.wikipedia.org_all-access_spider        18.0        11.0   
        1             2PM_zh.wikipedia.org_all-access_spider        11.0        14.0   
        2              3C_zh.wikipedia.org_all-access_spider         1.0         0.0   
        3         4minute_zh.wikipedia.org_all-access_spider        35.0        13.0   
        4  52_Hz_I_Love_You_zh.wikipedia.org_all-access_s...         NaN         NaN   

           2015-07-03  2015-07-04  2015-07-05  2015-07-06  2015-07-07  2015-07-08  \
        0         5.0        13.0        14.0         9.0         9.0        22.0   
        1        15.0        18.0        11.0        13.0        22.0        11.0   
        2         1.0         1.0         0.0         4.0         0.0         3.0   
        3        10.0        94.0         4.0        26.0        14.0         9.0   
        4         NaN         NaN         NaN         NaN         NaN         NaN   

           2015-07-09  ...  2017-09-01  2017-09-02  2017-09-03  2017-09-04  \
        0        26.0  ...        19.0        33.0        33.0        18.0   
        1        10.0  ...        32.0        30.0        11.0        19.0   
        2         4.0  ...         6.0         6.0         7.0         2.0   
        3        11.0  ...         7.0        19.0        19.0         9.0   
        4         NaN  ...        16.0        16.0        19.0         9.0   

           2017-09-05  2017-09-06  2017-09-07  2017-09-08  2017-09-09  2017-09-10  
        0        16.0        27.0        29.0        23.0        54.0        38.0  
        1        54.0        25.0        26.0        23.0        13.0        81.0  
        2         4.0         7.0         3.0         4.0         7.0         6.0  
        3         6.0        16.0        19.0        30.0        38.0         4.0  
        4        20.0        23.0        28.0        14.0         8.0         7.0  

        [5 rows x 804 columns]
```

- For each time series, we are provided with the name of the article as well as the type of traffic that this time series represent (all, mobile, desktop, spider). The data source for this dataset does not distinguish between traffic values of zero and missing values. A missing value may mean the traffic was zero or that the data is not available for that day.
- For our training we will replace the missing value with 0.
- We will transform the daily page visits using log1p.
- We will bea extracting the access, ageant and language information of the page from Page field in the dataset.
- We will have to transform the added access, ageant and language details to numerical value before feeding it to the model.
- The prediction we need to make is the number of visitis of the wikipdia page from Sep 13, 2017 - Nov 13, 2017

1.3.2 Exploratory Visualization

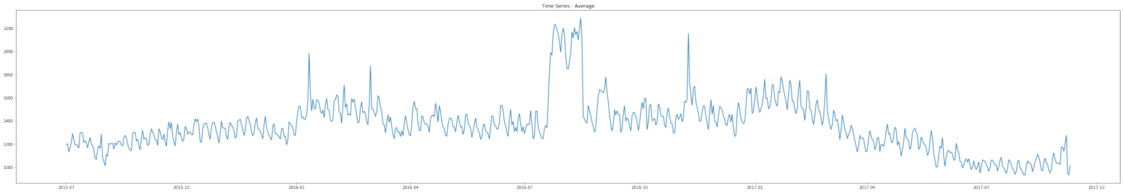
In this section, you will need to provide some form of visualization that summarizes or extracts a relevant characteristic or feature about the data. The visualization should adequately support the data being used. Discuss why this visualization was chosen and how it is relevant. Questions to ask yourself when writing this section: - *Have you visualized a relevant characteristic or feature about the dataset or input data? - Is the visualization thoroughly analyzed and discussed? - If a plot is provided, are the axes, title, and datum clearly defined?*

The training data contains prediction of page views for 803 days.

1.4 Initial Data Exploration

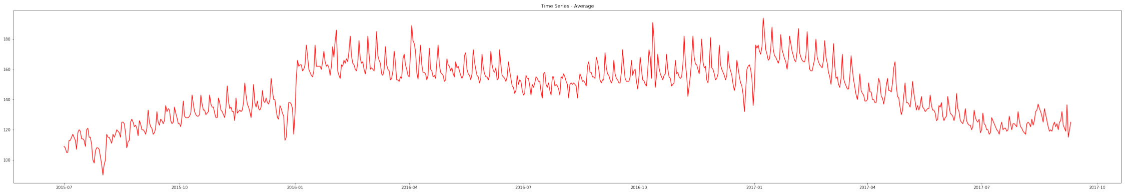
- Let's take each page and date into it's individual columns
- We will plot the statistical analysis of the page and daily visits during the training period to see if there is any intetesing analysis of visit.

In [11]:



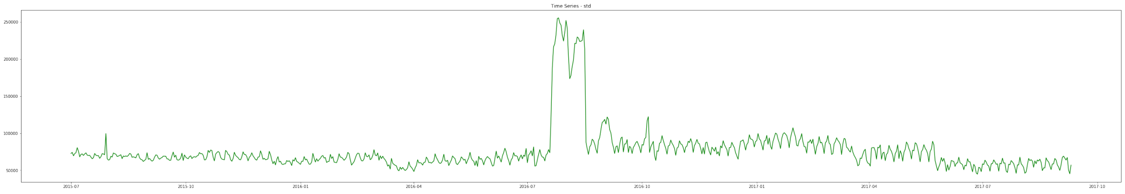
Mean
The above graph shows the mean of the page visits for given date

In [12]:



Median
The above graph shows the median of the page visits for given date

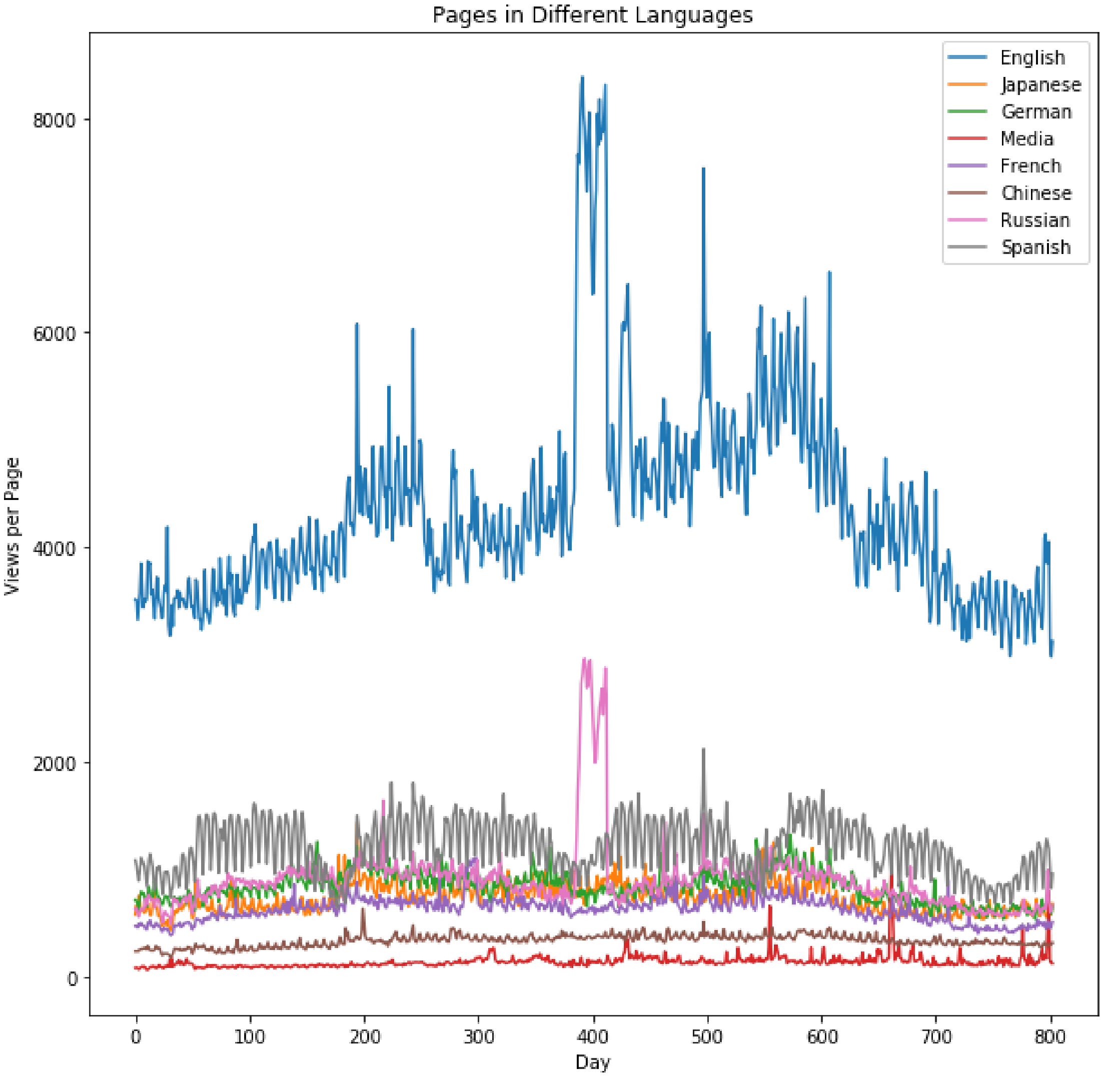
In [13]:



Standard Deviation
The above graph shows the standard deviation of the page visits for given date

Let's check if Language has any Impact on Page View

In [12]:



English shows a much higher number of views per page, as might be expected since Wikipedia is a US-based site. There is a lot more structure here than I would have expected. The English and Russian plots show very large spikes around day 400 (around August 2016), with several more spikes in the English data later in 2016. My guess is that this is the effect of both the Summer Olympics in August and the election in the US.

The Spanish data is very interesting too. There is a clear periodic structure there, with a ~1 week fast period and what looks like a significant dip around every 6 months or so.

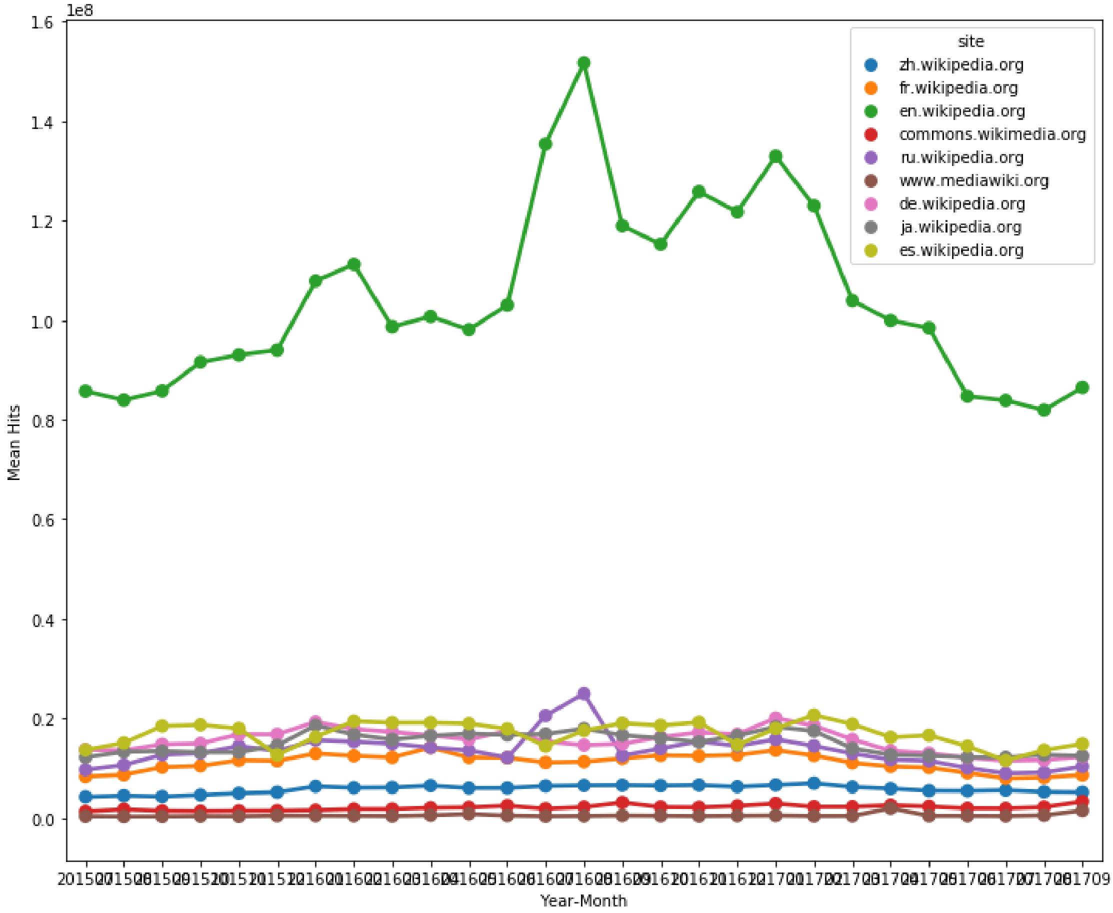
```
In [14]:
```

```
Out[14]:                    site      access       agent
         count            145063      145063      145063
         unique                9           3           2
         top     en.wikipedia.org  all-access  all-agents
         freq              24108       74315      110150
```

We can see above details the different feature for each page.

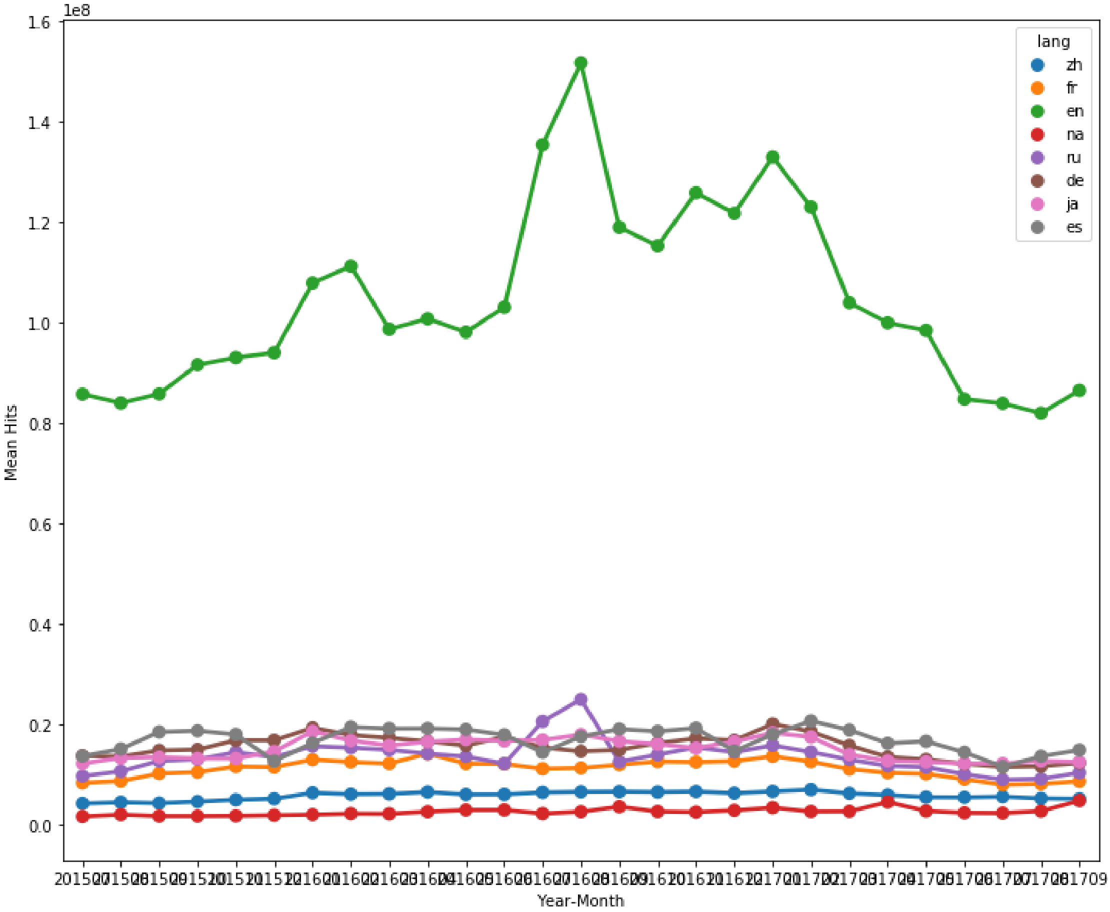
- The site 'zh.wikipedia.org', 'fr.wikipedia.org', 'en.wikipedia.org', 'commons.wikimedia.org', 'ru.wikipedia.org', 'www.mediawiki.org', 'de.wikipedia.org', 'ja.wikipedia.org', 'es.wikipedia.org'
- Access 'all-access', 'desktop', 'mobile-web'
- Agent 'spider', 'all-agents'

```
In [25]: plot_by_feature(site_columns, "site")
```



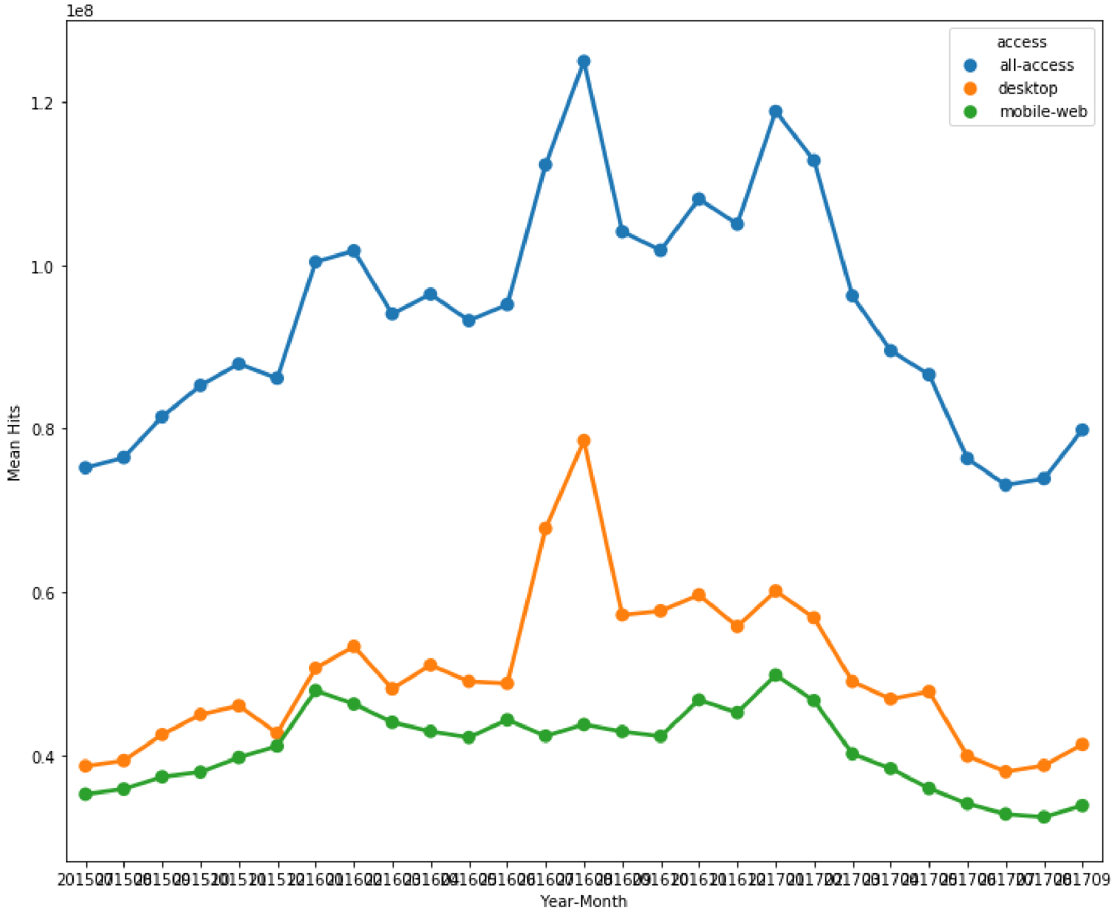
The site information for `zh.wikipedia.org` seems to be much higher than any other page. Adding this feature will better represent the underlying problem to the predictive models, resulting in improved model accuracy on unseen data.

```
In [26]: plot_by_feature(lang_columns, "lang")
```



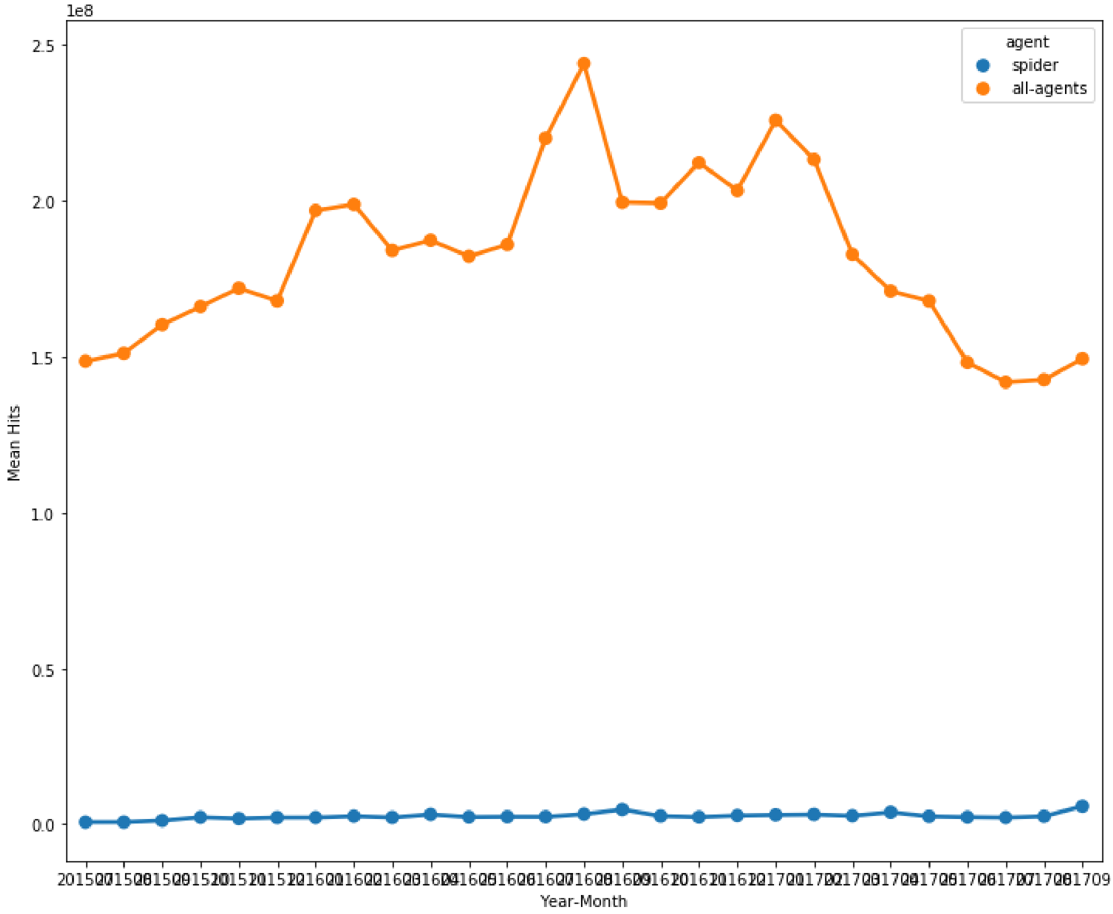
The language information english page seems to be much higher than any other page. We have already explored this relationship between data in one of the above graph. Aslo this data is quite similar to the site information we have shown above. Adding this feature won't help our model much.

```
In [27]: plot_by_feature(access_columns, "access")
```



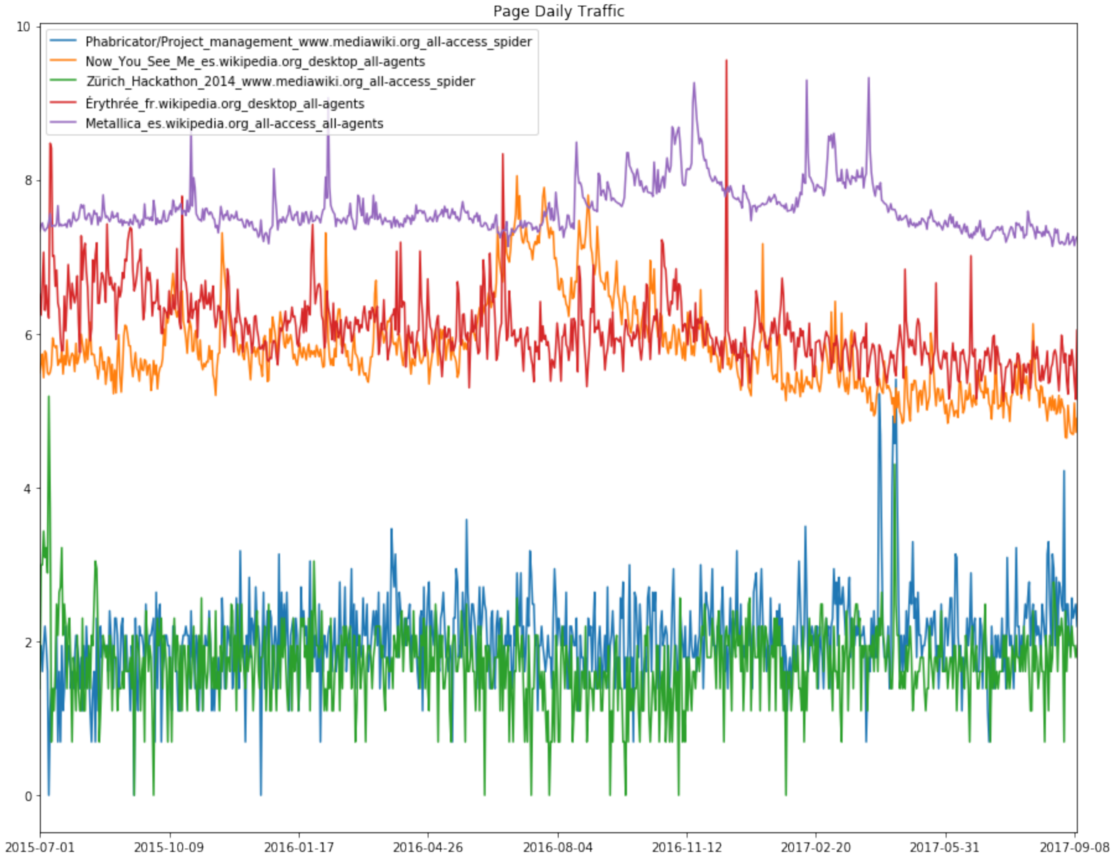
- We can see the difference in the page vists for above tree different access type, the access for `all-access` is far higher than the other two access type.
- The access for `desktop` and `mobile-web` seenms to be quite similar to one another, except between `March, 2016` and `September, 2016`, where the page visits for desktop is much higher than mobile visits.

```
In [28]: plot_by_feature(agents_columns, "agent")
```



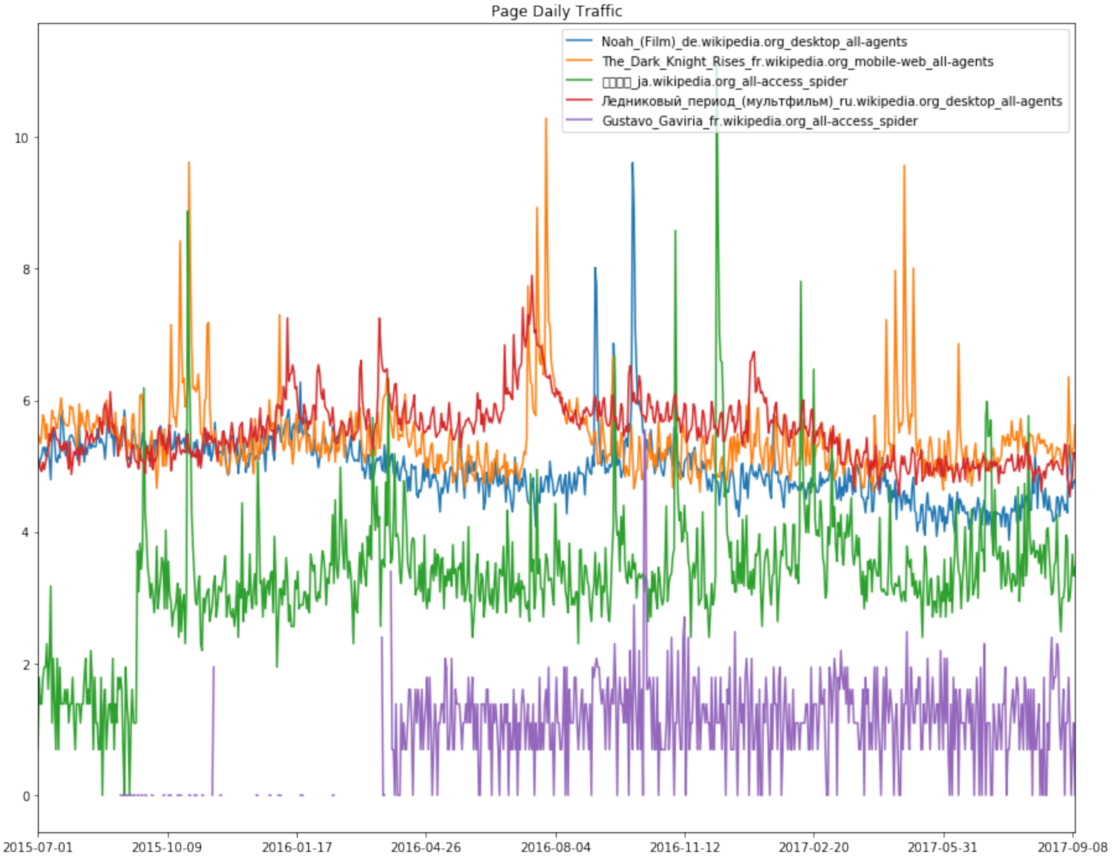
The page visists for agent `all-agent` seems to be much higher than that of `spider`. The page visits for `spider` is almost flat except for very small spikes in-between and towrads the end.

```
In [37]: plot_page_traffic(df_train, 5)
```



The above and below graph shows page visits for few random pages

```
In [38]: plot_page_traffic(df_train, 5, 2)
```



1.4.1 Algorithms and Techniques

This convolutional architecture is a full-fledged version of the WaveNet model, designed as a generative model for audio (in particular, for text-to-speech applications). The wavenet model can be abstracted beyond audio to apply to any time series forecasting problem, providing a nice structure for capturing long-term dependencies without an excessive number of learned weights.

The core building block of the wavenet model is the dilated causal convolution layer, This style of convolution properly handles temporal flow and allows the receptive field of outputs to increase exponentially as a function of the number of layers. This structure is nicely visualized by the below diagram from the wavenet paper.

The model also utilizes some other key techniques: gated activations, residual connections, and skip connections.

Gated Activations One of the potential advantages that wavenet has is that it contains multiplicative units (in the form of the LSTM gates), which may help it to model more complex interactions. To amend this, DeepMind replaced the rectified linear units between the masked convolutions in the original pixelCNN with the following gated activation unit: = tanh (,) (,) where denotes a convolution operator, denotes an elementwise multiplication operator, (ů) is a sigmoid function, is the layer index, and denote filter and gate, respectively, and W is a learnable convolution filter.

Residual and Skip Connections Both residual and parameterized skip connections are used throughout the network, to speed up convergence and enable training of much deeper models.

Our Architecture

- 16 dilated causal convolutional blocks
- Preprocessing and postprocessing (time distributed) fully connected layers (convolutions with filter width 1):
- 18 output units
- 32 filters of width 2 per block
- Exponentially increasing dilation rate with a reset (1, 2, 4, 8, 16, 32, 64, 128, 256, 1, 2, 4, 8, 16, 32, 64, 128, 256)
- Gated activations
- Residual and skip connections
- 2 (time distributed) fully connected layers to map sum of skip outputs to final output
- *Are the algorithms you will use, including any default variables/parameters in the project clearly defined?*
- *Are the techniques to be used thoroughly discussed and justified?*
- *Is it made clear how the input data or datasets will be handled by the algorithms and techniques chosen?*

  Increasing popularity of Wavenets and my interest to leverage neural nets for forecasting made us choose Wavenets over the rest. 'Google Deep mind' being the mastermind behind Wavenets (currently used in Google's AI service, Cloud-text-to-speech) made me futher insterested in exploring this model.

  Few research has also shown that Wavenets tend to perform at par or better than RNNs. Here is a link to an article which talks about how Wavenets can capture long term dependencies like LSTMs but are faster and more light weight for training.

1.4.2 Benchmark

The benchmark is the Kaggle Leaderboard score of sjv's model, which I have taken as a benchmark. His model got a score of 37.420002 on Kaggle leaderboard

1.4.3 Building a Baseline Model

1.5 Model has two main parts: encoder and decoder.

The baseline model that we have build gets below score:

```
loss: 0.2412 - val_loss: 0.3174
```

- The new model should perform better than this.

1.6 III. Methodology

1.6.1 Data Preprocessing

There are two ways to split timeseries into training and validation datasets:

Walk-forward split. This is not actually a split: we train on full dataset and validate on full dataset, using different timeframes. Timeframe for validation is shifted forward by one prediction interval relative to timeframe for training. Side-by-side split. This is traditional split model for mainstream machine learning. Dataset splits into independent parts, one part used strictly for training and another part used strictly for validation.

Walk-forward is preferable, because it directly relates to the competition goal: predict future values using historical values. But this split consumes datapoints at the end of timeseries, thus making hard to train model to precisely predict the future.

A great explanation has been provided in Arthur's web traffic solution.

To train our model we will divide the data in training and validation set using below logic:

1. Train encoding period
2. Train decoding period (train targets, 64 days)
3. Validation encoding period
4. Validation decoding period (validation targets, 64 days)

```
In [41]: print('Train encoding:', train_start_encoded, '-', train_end_encoded)
         print('Train prediction:', train_start_pred, '-', train_end_pred, '\n')
         print('Val encoding:', val_start_encoded, '-', val_end_encoded)
         print('Val prediction:', val_start_pred, '-', val_end_pred)

         print('Encoding interval:', encoded_length.days)
         print('Prediction interval:', prediction_length.days)
```

```
Train encoding: 2015-07-01 00:00:00 - 2017-05-05 00:00:00
Train prediction: 2017-05-06 00:00:00 - 2017-07-08 00:00:00

Val encoding: 2015-09-03 00:00:00 - 2017-07-08 00:00:00
Val prediction: 2017-07-09 00:00:00 - 2017-09-10 00:00:00
Encoding interval: 675
Prediction interval: 64
```

Feature Engineering We have added few extra features in our model.

- We have added site information for the pages.

```
   'zh.wikipedia.org', 'fr.wikipedia.org', 'en.wikipedia.org',
'commons.wikimedia.org', 'ru.wikipedia.org', 'www.mediawiki.org',
'de.wikipedia.org', 'ja.wikipedia.org', 'es.wikipedia.org'
```

- We have added access information for the pages.

```
   'all-access', 'desktop', 'mobile-web'
```

- We have added agent information for the pages.

```
'spider', 'all-agents'
```

- We have added language information for the pages.

```
'zh', 'fr', 'en', 'na', 'ru', 'de', 'ja', 'es'
```

Data Formatting

- Pull the time series into an array, save a date to index mapping as a utility for referencing into the array
- Create function to extract specified time interval from all the series
- Create functions to transform all the series.
- Here we smooth out the scale by taking log1p and de-meaning each series using the encoder series mean, then reshape to the (n_series, n_timesteps, n_features) tensor format that keras will expect.
- If we want to generate true predictions instead of log scale ones, we can easily apply a reverse transformation at prediction time.

1.6.2 Implementation

The implementation covers the fllowing flow:-

Load Data File: Load the training data from provided csv file into data frame

Explore Data: We will perform some exploratory data analysis to understand the provided data. We will also analyze page visit based on some features. We will plot these data to get better understanding of the effect of these data on the page views.

Feature Engineering: After analysis of data we will create few new features - Language of the page - Access type of the page - Agent of the page - Site detail of the page

Data Processing: For better training our model, we will transform the daily page vists using log1p transform. The new feature added in the above steps will be converted to numerial value code using pandas.

Preparing Training and split Data: To train our model we will divide the data in training and validation set using below logic:

```
1. Train encoding period
2. Train decoding period (train targets, 64 days)
3. Validation encoding period
4. Validation decoding period (validation targets, 64 days)
```

We'll do this by finding the appropriate start and end dates for each segment. Starting from the end of the data we've loaded, we'll work backwards to get validation and training prediction intervals. Then we'll work forward from the start to get training and validation encoding intervals.

Preparing Data for our Model: We have prepared time segment dates, we'll build data to provide to our model.

- Pull the time series into an array, save a date_to_index mapping as a utility for referencing into the array
- Create function to extract specified time interval from all the series

- Create functions to transform all the series.
- We smooth out the scale by taking log1p and de-meaning each series using the encoder series mean, then reshape to the (n_series, n_timesteps, n_features) tensor format that keras will expect.
- While generating true predictions instead of log scale ones, we can easily apply a reverse transformation at prediction time.

Prepare our Model: We will be using Wavenet model for predicting data. The architecture of the project has been explained above in the notebook.

- 16 dilated causal convolutional blocks
- Preprocessing and postprocessing (time distributed) fully connected layers (convolutions with filter width 1):
- 18 output units
- 32 filters of width 2 per block
- Exponentially increasing dilation rate with a reset (1, 2, 4, 8, 16, 32, 64, 128, 256, 1, 2, 4, 8, 16, 32, 64, 128, 256)
- Gated activations
- Residual and skip connections
- 2 (time distributed) fully connected layers to map sum of skip outputs to final output
- We'll extract the last 64 steps from the output sequence as our predicted output for training.

Model Training: - We will train our model with data prepared above. For training our model we will be using a method used in RNN called teacher forcing, teacher forcing is a strategy for training recurrent neural networks that uses model output from a prior time step as an input.

- We will train our model on GPU(we will need tensorflow-gpu and keras). Training the above model will need lot of computational hardward resources. Training this model on 12 GB CPU takes around 7-8 hours for 50 epochs. We have used GPU for training. The training time was reduced to 1/10 of the CPU time.

Inference Loop: - We'll generate predictions by running our model in a loop, using each iteration to extract the prediction for the time step one beyond our current history then append it to our history sequence. With 64 iterations, this lets us generate predictions for the full interval we've chosen.

Predict and Plot: - We'll pull out our set of validation encoder/target series shifted forward in time. Then using a plotting utility function, we can look at the end of the encoder series, the true target series, and the predicted target series. This gives us a feel for how our predictions are doing.

1.6.3 Refinement

- Initially I tried to train the model with learning rate of 1e-1, at this learning rate the model was overfitting very badly. The training error was decreasing with each epoch but validation error seemed to plateau around ~0.3242.
- I tried to increase the dilation layer to 20, but it didn't seem to improv the score. Even training that model was around 1.5x slower than our current model. It seem the default listed in the Wavenet paper seems to generalize well.
- The other things I tried for refinement were.
  - Reducing the learning rate by half when the validation loss is same for two iteration. This made the validation loss decrease further.
  - I saw training the model for more number of epoch made the model overfit badly, the training loss was decreaing but the validation loss had plateaued. So, the added the Early stopping criteria for our model. If the validation loss is same for aroung 5 baches we will stop training further.
  - We were saving the best model weight after each iteration.

1.7 IV. Results

1.7.1 Model Evaluation and Validation

We'll generate predictions by running our model in a loop, using each iteration to extract the prediction for the time step one beyond our current history then append it to our history sequence. With 64 iterations, this lets us generate predictions for the full interval we've chosen.

We can generate predictions for history and target data. We will define plotting function which will plot the true target series, and the predicted target series. This gives us a feel for how our predictions are doing.

- The final model gets the below score:-

  ```
  loss: 0.2496 - val_loss: 0.3032
  ```

- The test set generates an score of 37.61846 on kaggle submission'

1.7.2 Justification

The final model trained on total data has loss of 0.3032 on the validation data, on testing set I was able to get an score of 37.61846 on Kaggle Submission. This means that we are able to reach the benchmark set for the model.

In the boundaries of the competition, I can say these results are encouraging and show that the approach taken is the right direction, although there is few places where the model can still improve.

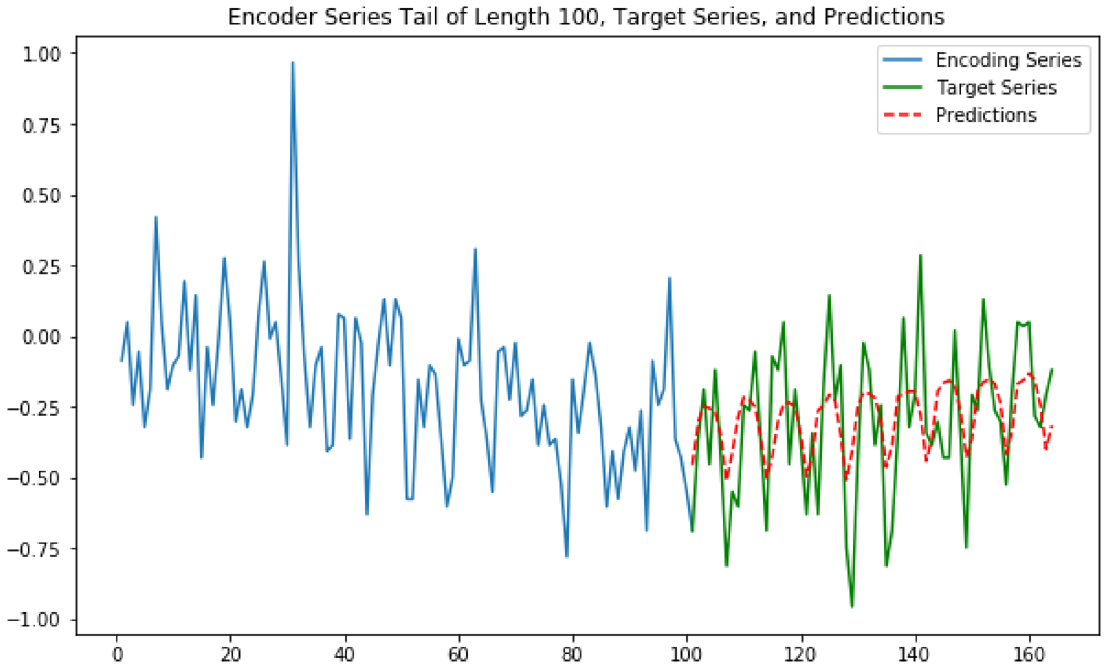
```
In [ ]:
```

1.8 V. Conclusion

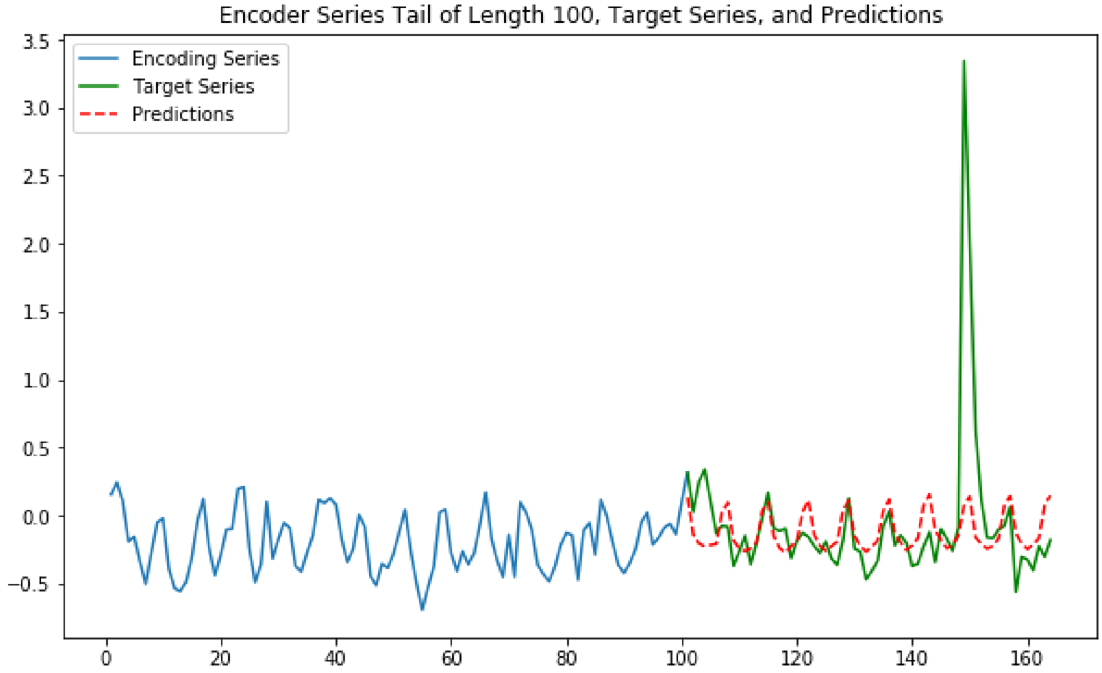
1.8.1 Free-Form Visualization

We predicted the data for last 64 days in the dataset through our model. The following are the trends for 6 random Wikipedia articles of our choice. As you can see the trends for the log of page views are being captured quiet well. The peaks are still hard to capture as is the case for all time series forecasting.

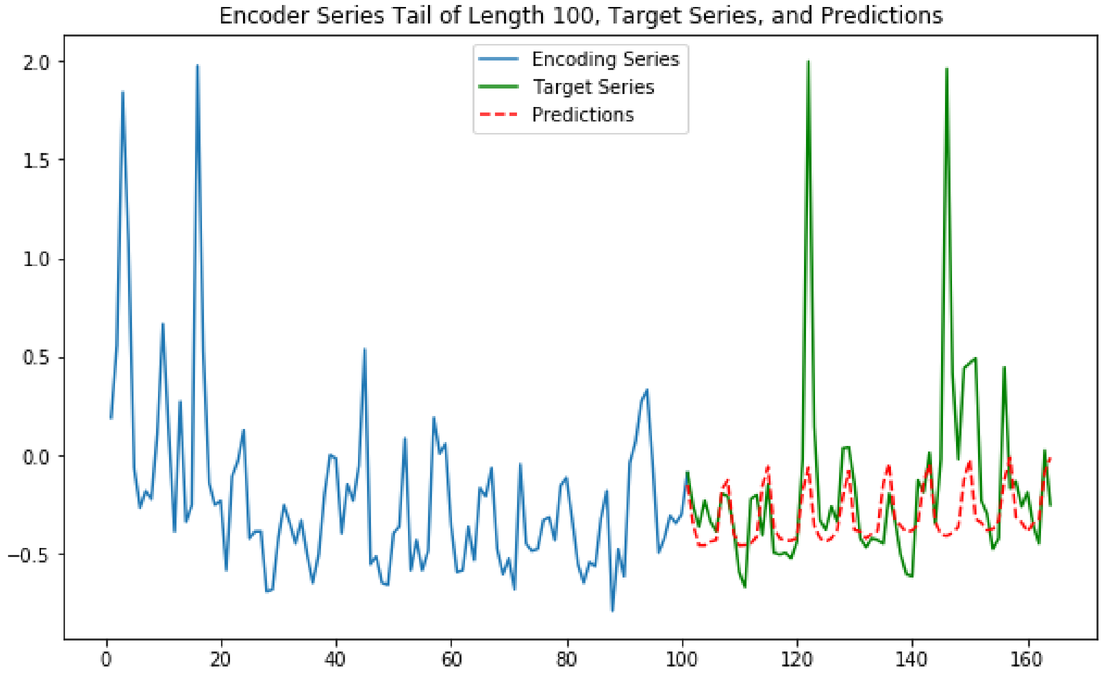
```
In [69]: plot_prediction(encoder_input_data, decoder_target_data, sample_index=10000)
```



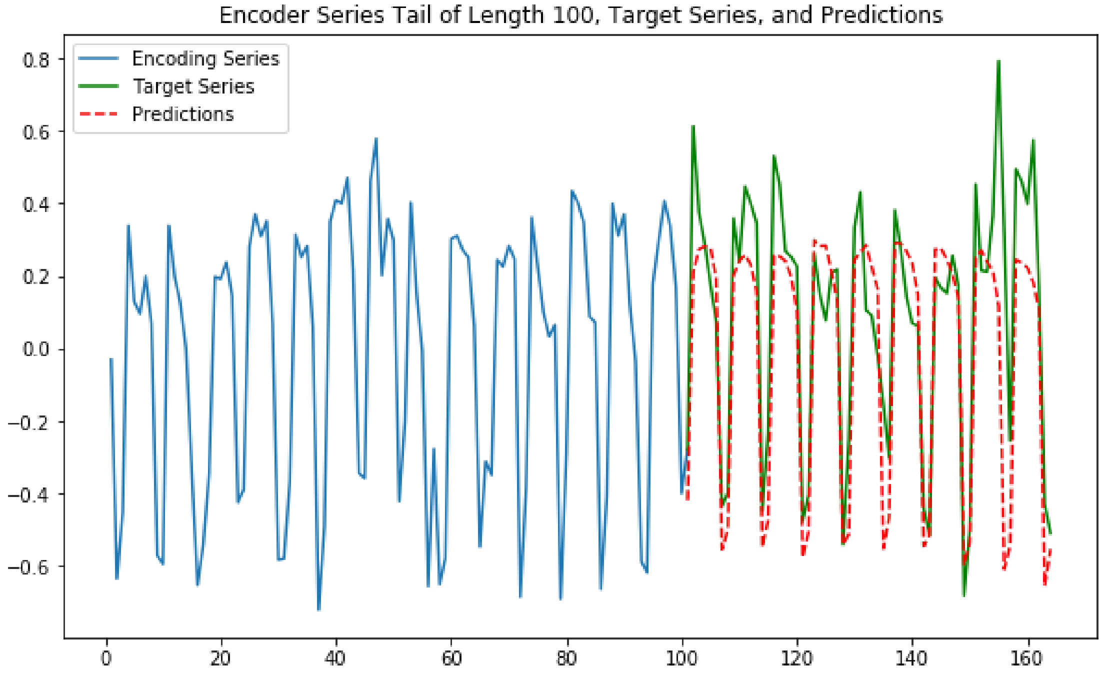
```
In [70]: plot_prediction(encoder_input_data, decoder_target_data, sample_index=16555)
```



```
In [71]: plot_prediction(encoder_input_data, decoder_target_data, sample_index=18000)
```



```
In [72]: plot_prediction(encoder_input_data, decoder_target_data, sample_index=68000)
```



1.8.2 Reflection

The whole project experience can be summarized as follows: - The dataset is available on Kaggle, Web Traffic Analysis. - Load the data in dataframe. - Visualize and preprocess the data. - Build a good training and validation set, a good training and valiation set is very critical for good time series prediction. - Tuning learning rate helped a lot while training. Learning rate is very critical in training a deep learning Model. - We used Keras funtionality like ReduceLearningRate and EarlyStopping and ModelCheckpoint while training our model. - Comparing the result with sjv's model, which I have taken as a benchmark. His model got a score of 37.420002 on Kaggle leaderboard

- The trained model can be used to predict future page visits.

  Regarding the challenges faced: - This dataset has around 145k records, so training the model on such large data was very challenging. I had to take sample of 20-40k and train and model to see if it's working well before training on the whole dataset. - Initally I started traing this model on my laptop, it was very very slow. It took around 2-3 hours to complete around 5 epochs. Then I moved to gcs instance, where it took me around 7-8 hours to train around 30 epochs. The most interesting thing happened during the last phase of the project, I unintalled tensorflow from my gcs instance and installed tensorflow-gpu, I was very impressed with the reduction in taining time, Initally it took around 25-30 min for training one epoch. When I changed to tensorflow-gpu the time significantly drooped to 2min per epoch. - The gcs instance I am using is a spot instance which somtime automatically get's shutdown. It happened twice while training the model. I had to again restart the training. It took lot of time for restaring again from begining. Would have been better if I would have started training the model from last saved model weight. - The most difficult part was choosing the loss function, initially i started with smape loss but it was giving a validation loss of `nan` when training for long duration. While going through Kaggle discussion forum I read the wierdness with smape and that we can use mae loss on log data.

1.9 Futher Improvements

- I would like to try an ensemble model to see if that improves the score that we have got above. I wanted to train an xgboost regressor on the output from our current model, will try it later.
- I am planning to train the model using the one cycle policy and see how much it reduces the training time.
- In the current model we are not using any weekday/holiday feature, would be interesting to see if that improves the score.

1.9.1 References

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

- https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/43795#latest-525730
- https://www.kaggle.com/headsortails/wiki-traffic-forecast-exploration-wtf-eda
- https://www.kaggle.com/muonneutrino/wikipedia-traffic-data-exploration
- SMAPE
- https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error
- MAE
- https://en.wikipedia.org/wiki/Mean_absolute_error
- Other Resources used for this project
- https://github.com/JEddy92/TimeSeries_Seq2Seq
- https://github.com/Arturus/kaggle-web-traffic
- https://towardsdatascience.com/web-traffic-forecasting-f6152ca240cb

```
In [ ]:
```