web_traffic_prediction_capstone_project_2-Copy1

June 23, 2019

1 Machine Learning Engineer Nanodegree

1.1 Capstone Project

Lalit Yadav June 23, 2019

1.2 I. Definition

1.2.1 Project Overview

This is a competition hosted on Kaggle, the dataset has been taken from Kaggle.

This competition 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.

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.

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

(approx. 2-4 pages)

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

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 III. Methodology

(approx. 3-5 pages)

1.4.1 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.

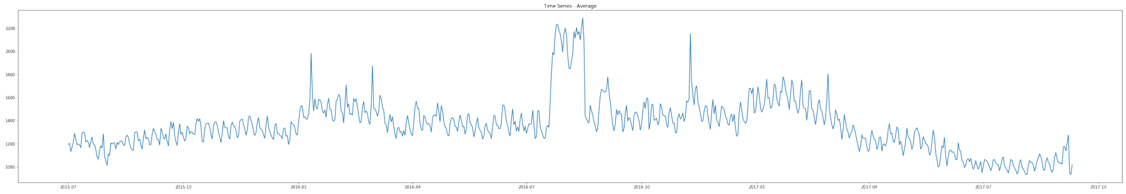
To train our model we will divide the data in training and validation set using below logic:

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

1.5 Initial Data Exploration

- Let's make each page and date into it's individual columns
- Also we will check if the given day is a weekend or weekday

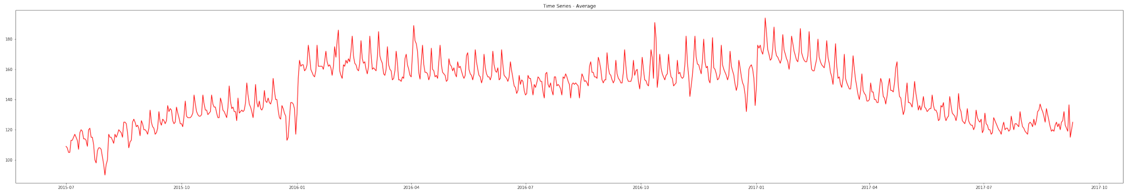
```
In [11]:
```



Mean

The above graph shows the mean of the page visits

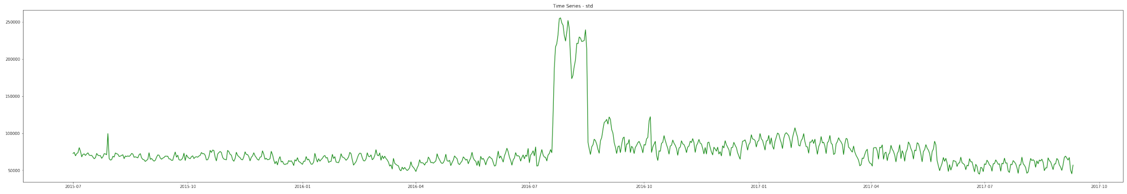
```
In [12]:
```



Median

The above graph shows the median of the page visits

```
In [13]:
```

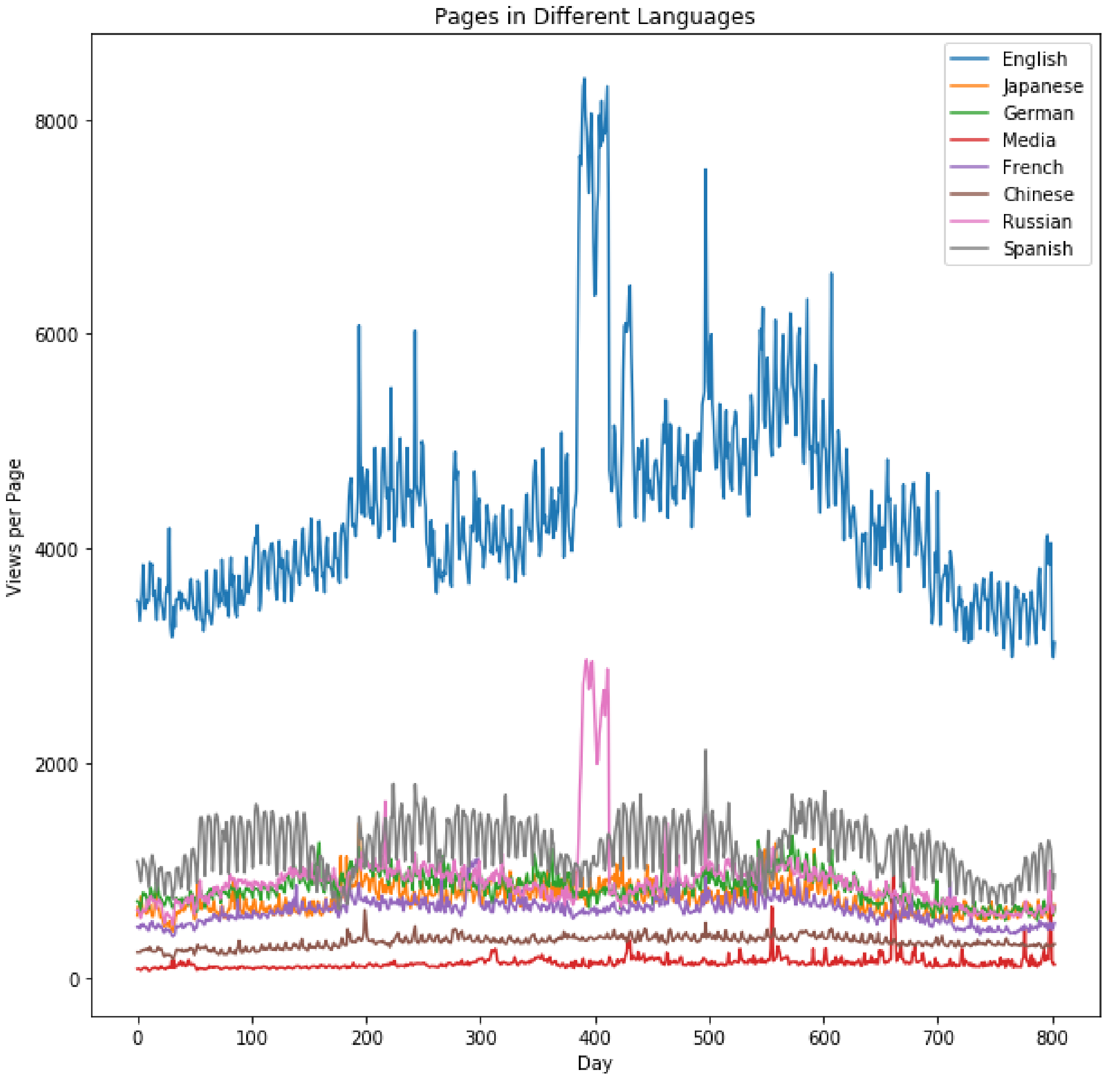


Standard Deviation

The above graph shows the standard deviation of the page visits

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

In [12]:



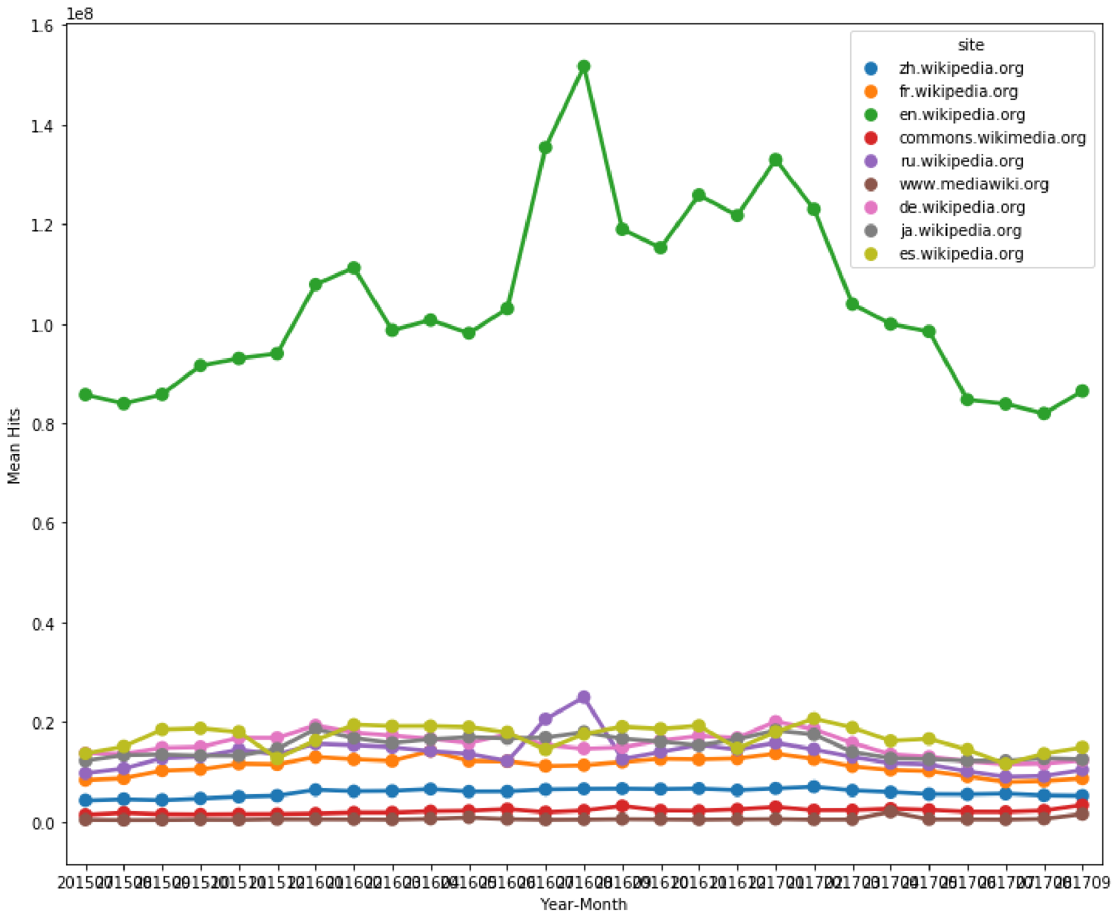
In [14]:

Out[14]:

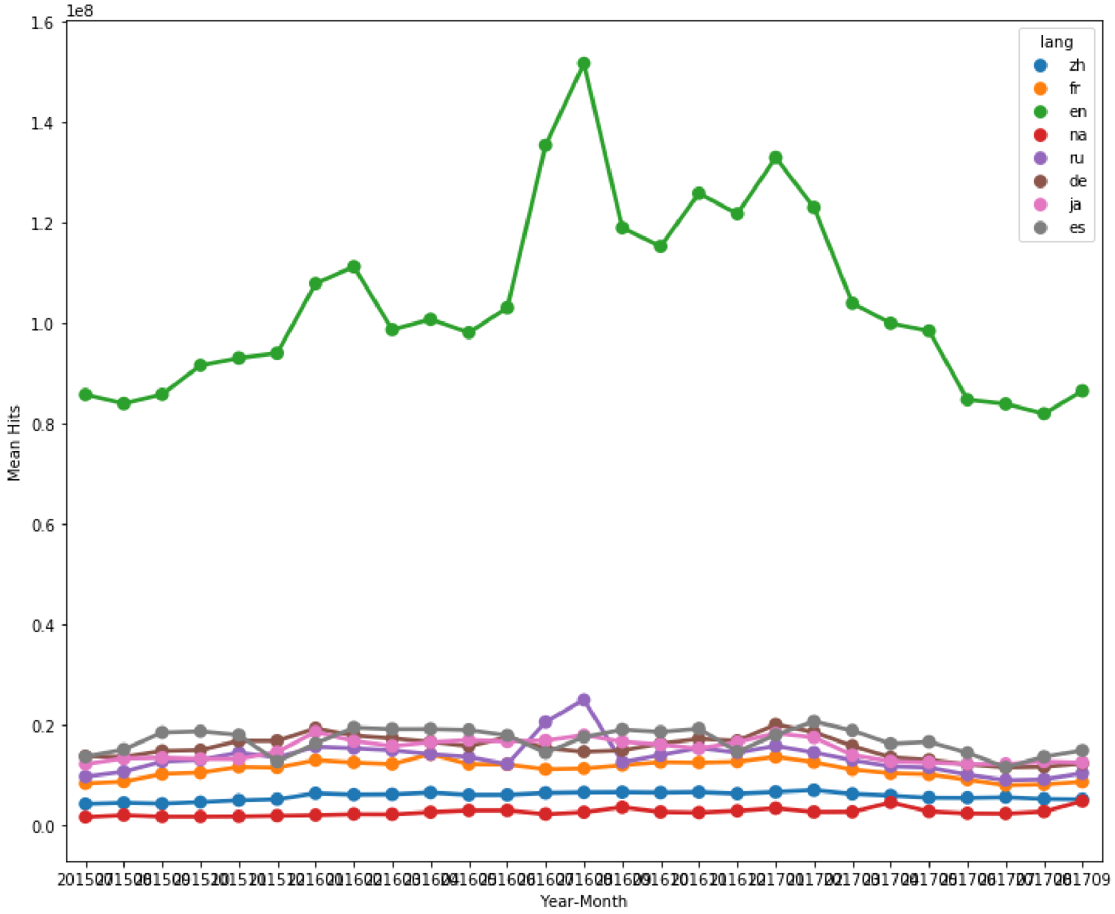
```
                    site      access       agent
count             145063      145063      145063
unique                 9           3           2
top     en.wikipedia.org  all-access  all-agents
freq               24108       74315      110150
```

```
In [ ]:
```

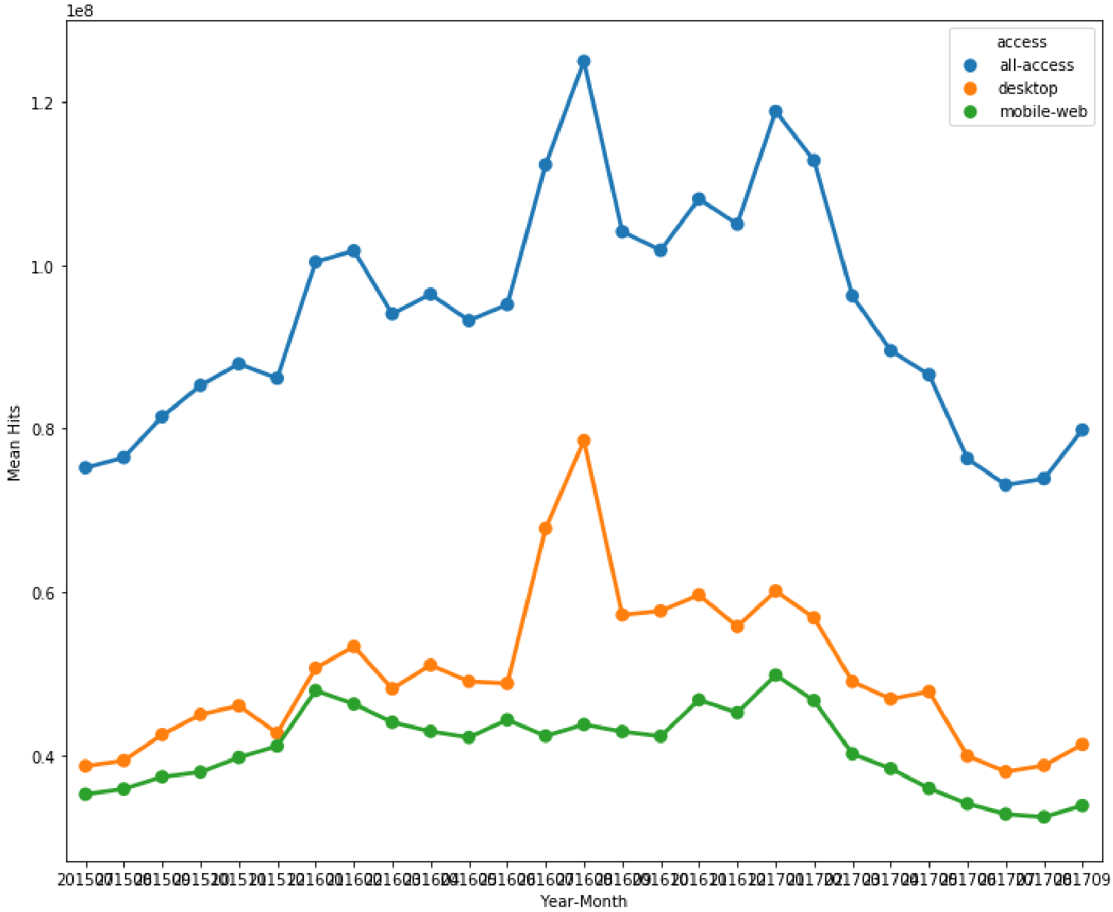
```
In [25]: plot_by_feature(site_columns, "site")
```



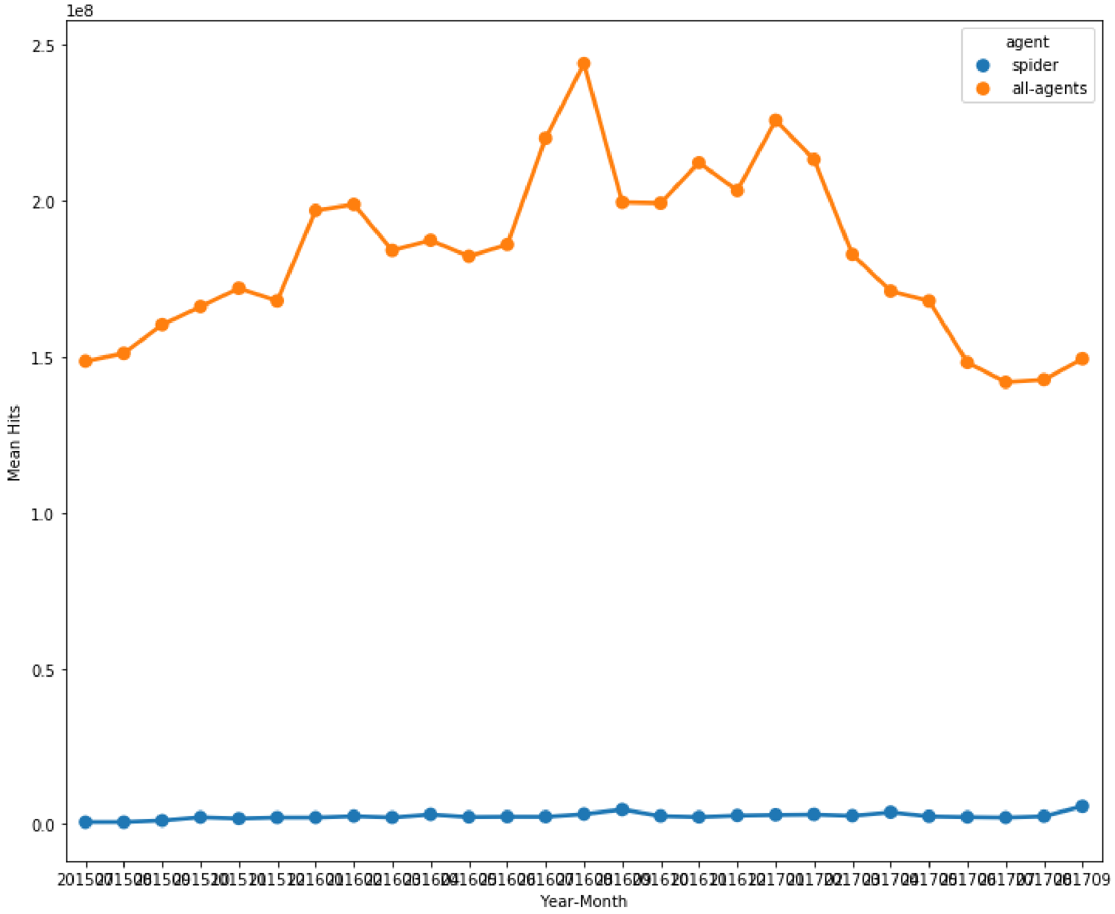
```
In [26]: plot_by_feature(lang_columns, "lang")
```



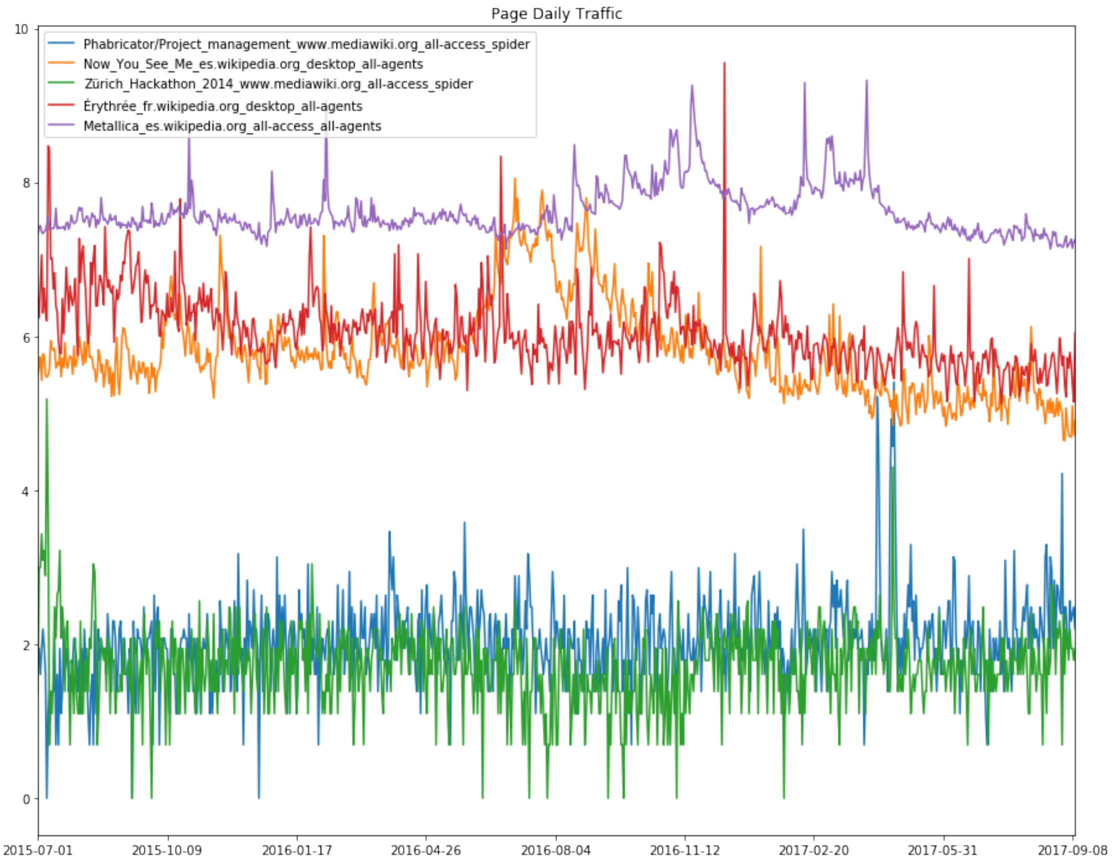
```
In [27]: plot_by_feature(access_columns, "access")
```



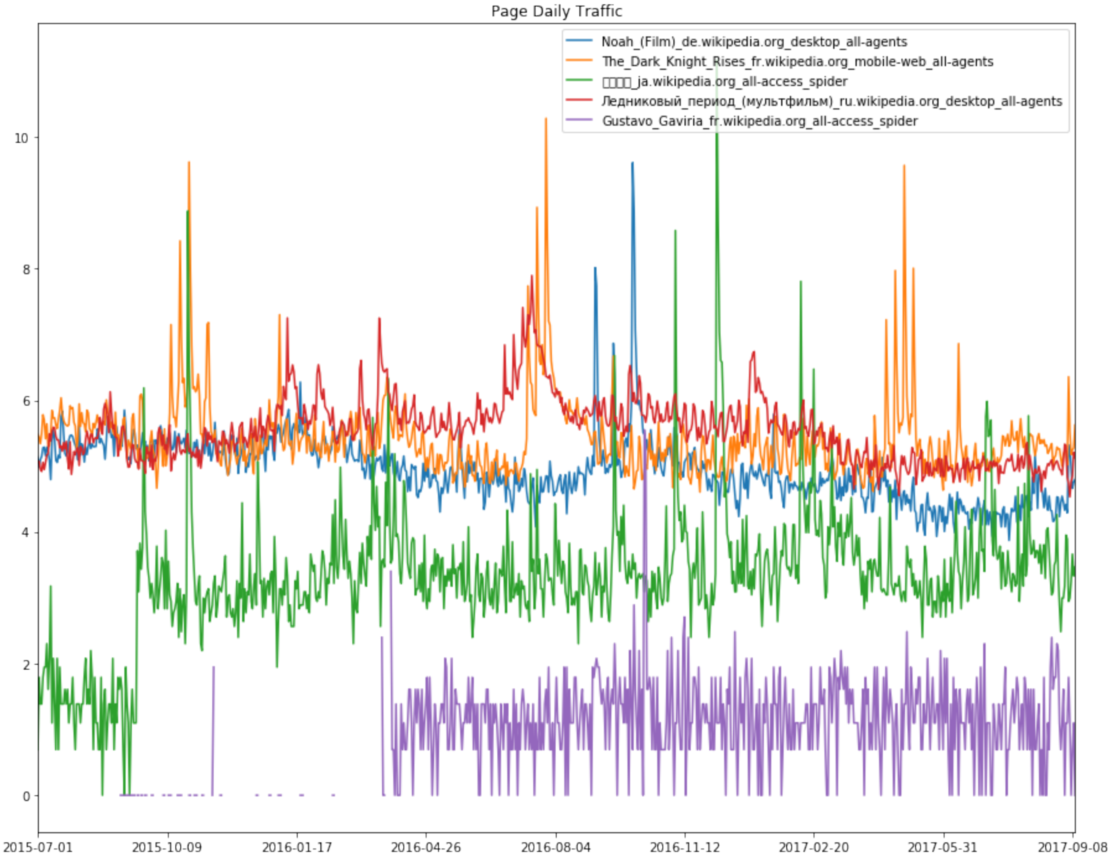
```
In [28]: plot_by_feature(agents_columns, "agent")
```



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



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



1.5.1 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?*

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.

1.7 Losses and Regularization

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:

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

1.7.1 Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section: - *Has some result or value been provided that acts as a benchmark for measuring performance? - Is it clear how this result or value was obtained (whether by data or by hypothesis)?*

1.8 Building The Baseline Model

1.9 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.
- The benchmark for the new model will be the score on Kaggle leader board

1.9.1 3. Building the Model - Architecture

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

```
In [ ]:
```

1.10 IV. Results

1.10.1 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'

1.10.2 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.

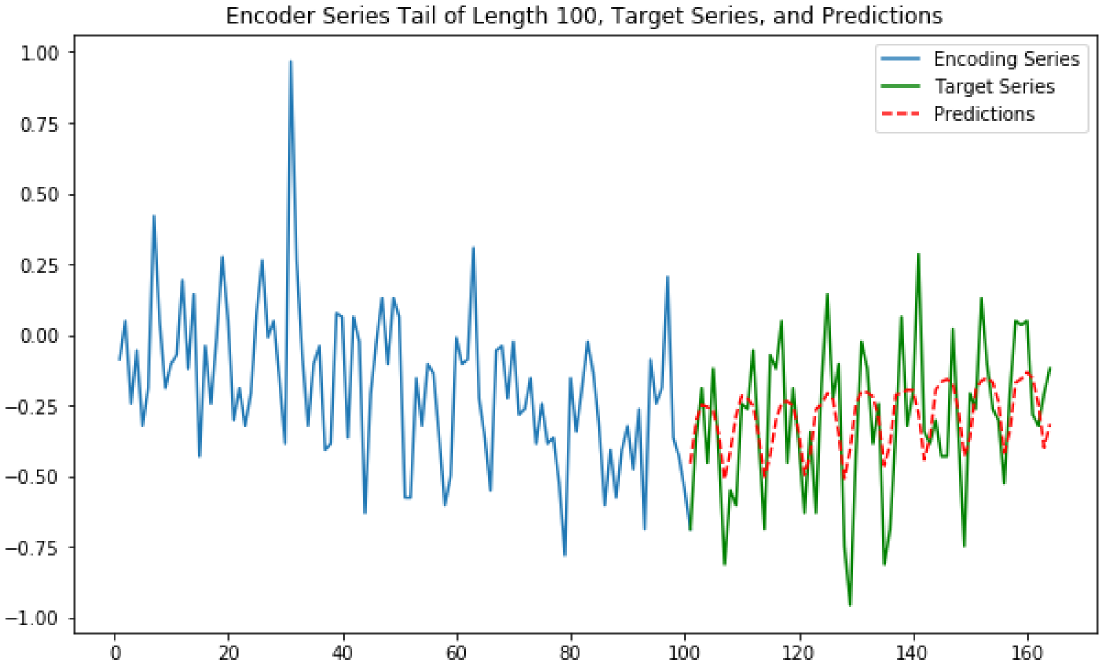
```
In [ ]:
```

1.11 V. Conclusion

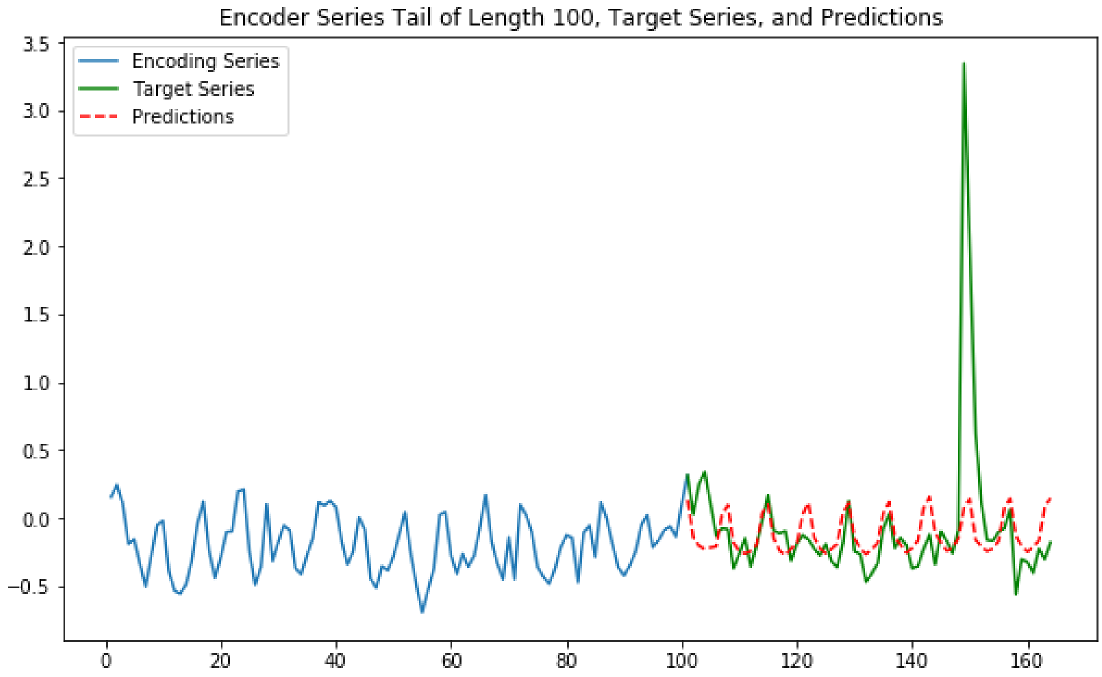
1.11.1 Free-Form Visualization

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.

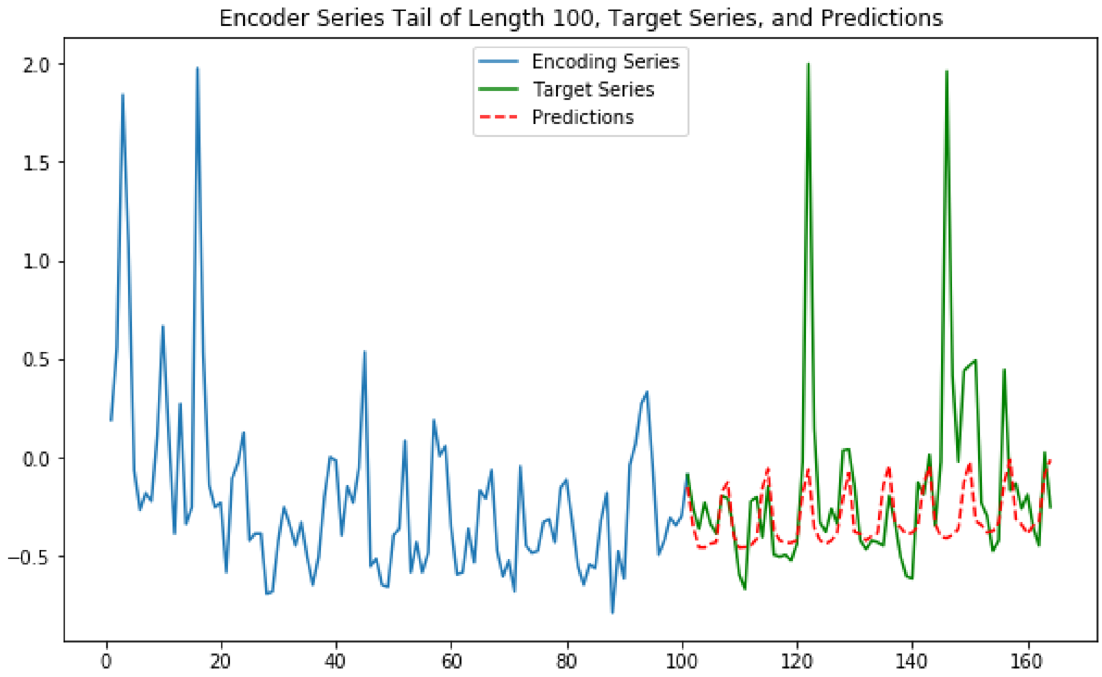
```
In [69]: plot_prediction(encoder_input_data, decoder_target_data, sample_index=10000)
```



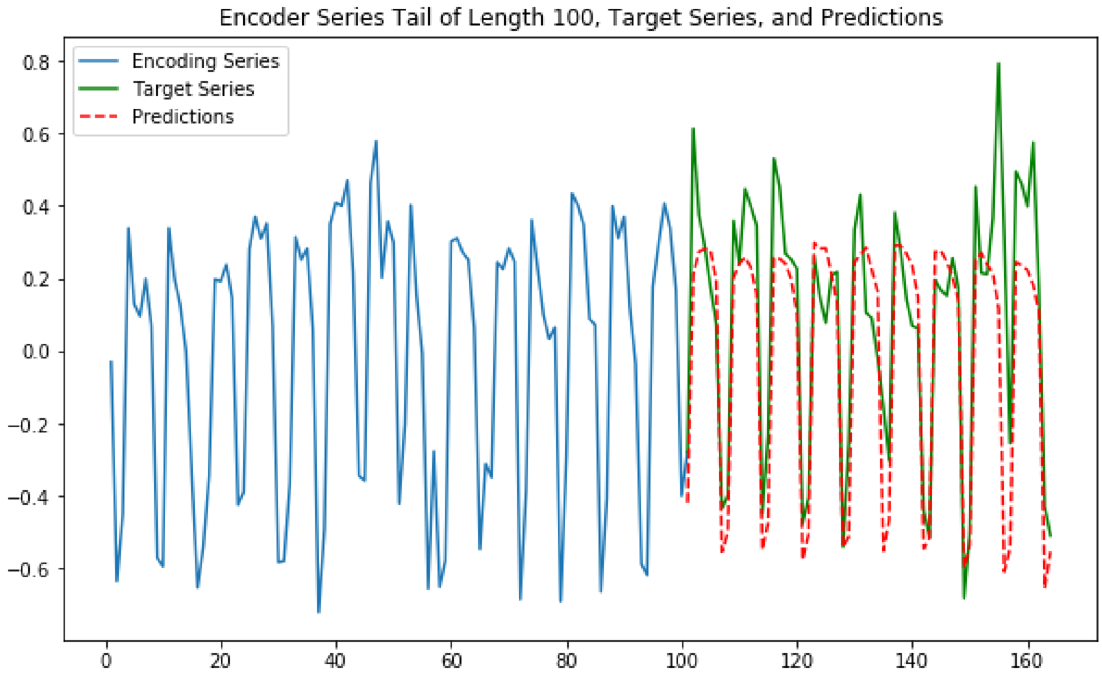
```
In [70]: plot_prediction(encoder_input_data, decoder_target_data, sample_index=16555)
```



```
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)
```



1.11.2 Reflection

- Getting good result si really challenging on this dataset.
- Initialy had tried model like RandomForest and XGBoost, the validation score for those were in the range of 45-50.
- This dataset has around 145k records, so training the model on such large data is very challenging.
- Initally I was using tensorflow cpu to train the model on google cloud instance. Each epoch was taking aroung 25 mins to complete. When I changed to tensorflow-gpu the time significantly drooped to 2min per batch. This was very unexpected for me.
- The biggest challenge that I faced while doing this project was in getting started.

1.12 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.

1.12.1 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 [ ]:
```