Web Traffic Prediction Report

Jiayi Gao, Xiaoyu Xue, Ben Zhang

1. Introduction

Nowadays internet surfing is almost a daily routine for everyone in the society. When people visited a webpage, they left digital footprints and produced "Web traffic", which is determined by the number of visitors and the number of pages they visit. Since the mid-1990s, web traffic has been the largest portion of Internet traffic.[1] Web traffic data is also a resourceful mine from which we can find out the trends of different domains such as commercial, politics, entertainment, to implement a more accurate marketing and advertising. By forecasting the traffic distribution and tendency, we can avoid web traffic overload and improve server efficiency, thus provide better internet surfing experience. In order to achieve those objectives and dig out gold from web traffic data, we will forecast web traffic in a given time period.

We use the dataset from Kaggle which contains the number of visits for a collection of wiki pages contains visits starting from July 1st, 2015 up until September 10th, 2017, our goal is to predict the last 60 days traffic of these pages. The name of the article as well as the type of traffic (all, mobile, desktop, spider) is given for each article.The wiki page feature contains interesting information like article id, article name, language, way of access, agent and etc. which can be extracted and to be applied in our model. In addition, we scrape the wiki page references counts as our supplementary feature.

Since this is a time series dataset, we apply some popular time series models including Arima, Prophet in our baseline phase. We also build Ridge Regression as reference model. We find that machine learning models can not produce best result even after being tuned. So we set our rudder towards deep learning and attempted seq2seq CNN, LSTM. Combined with heuristic knowledge, we deploy our model with selecting different sets of features and tuning the parameters, and find that seq2seq CNN returns the best results among all the tested models.

Our project solution and founding are not only beneficial and applicable to web traffic time series forecasting, but also can be widely implemented on many other temporal and sequential data.

2. Preliminary Data Analysis

In this project, two datasets has been used. The main data set is Wikipedia traffic data starting from July, 1st, 2015 up until September 10th, 2017, which is provided by a research prediction competition held on Kaggle. The other data set is wikipedia page content data, which is downloaded by crawler app we developed.

2.1 Traffic Data:

The dataset consists of approximately 145k time series. Each of these time series contains meta information and a number of daily views, including article id, article name, language, access, date and daily hit counts. The main issue in this data set is that about 8% values are missing and it doesn't distinguish between traffic values of zero and missing values.

Observations on this Data Set:

- Non-stationary vs. Stationery:

A stationary time series is one whose statistical properties such as mean, variance, autocorrelation, etc. are all constant over time. Some entries in our dataset is apparently non-dictionary and some is dictionary. We decide to use different approaches to predict on these different time series.

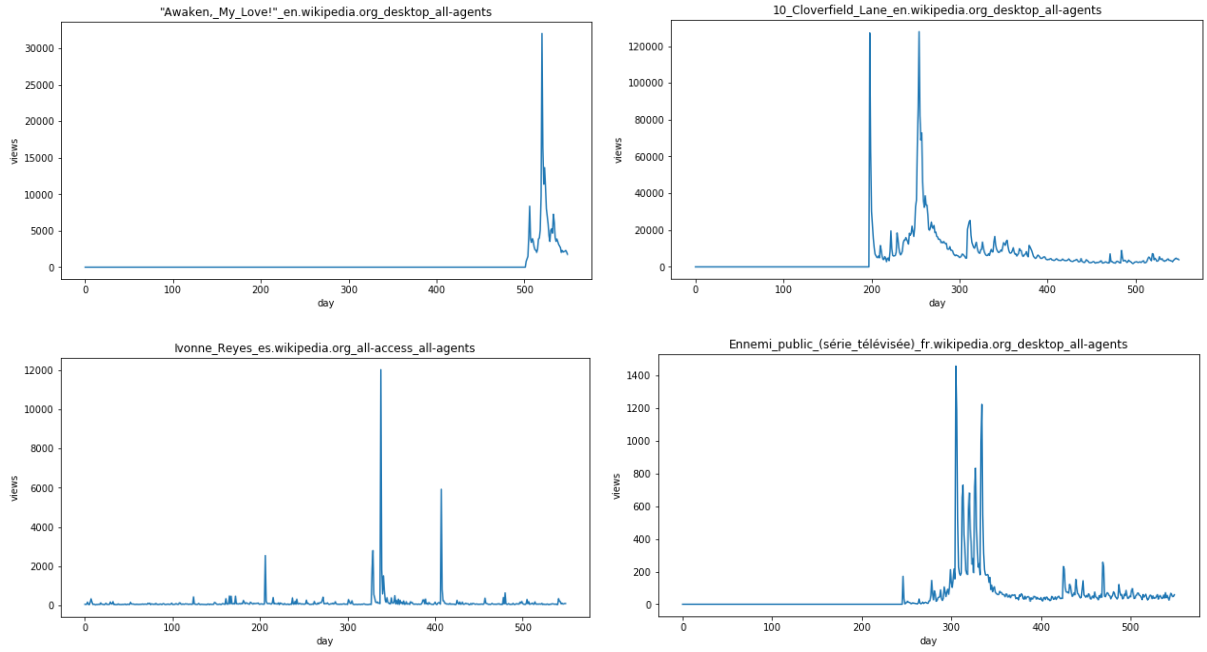

Figure 1: sample wiki pages - Non-stationary

- Periodical:

Many entries have periodical structures. We see that the periodic features are mainly by week and month, which is reasonable that browsing behaviors may differ on weekdays compared to weekends.

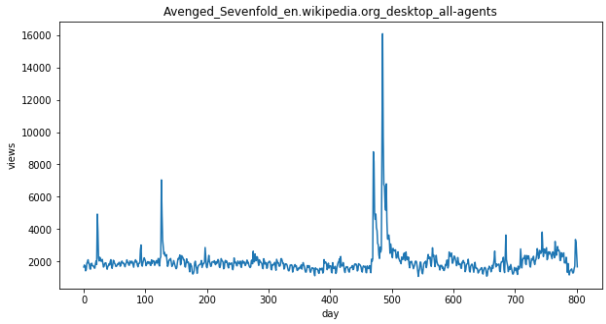

Figure 2: sample wiki page - Periodical

- Outlier:

The other observation we discovered in this dataset is that many time series has some outliers in a very short time. Apparently, these high visits value might be triggered by a specific event. This feature is more complicated to be applied on our training model because it needs massive real-time events data happening across the world.

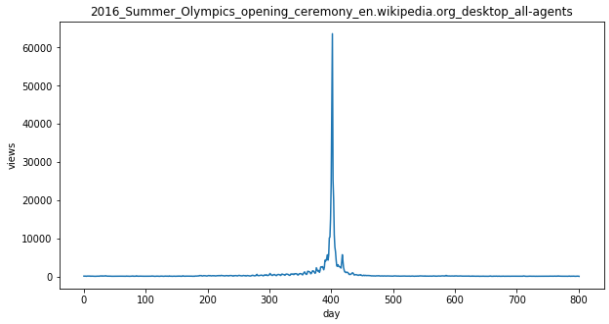

Figure 3: sample wiki page - Outlier

2.2 Additional Page Data

The other data set is the page content data grabbed by our spider program. This data set has about 69k entries, which covers about 50% on the origin dataset. It contains page key and references count on each page. Since the missing data is quite high, it leaves us a point to be improved.

3. Evaluation Metric

In general, there are two groups of evaluation metrics: scale-dependent metric and scale-independent metric. Scale-dependent metric family has mean square error (MSE), root mean square error (RMSE), mean absolute error (MAE) etc., which are useful when evaluate different forecasting methods that are applied to data with the same scale, but should not be used when comparing forecasts for series that are on different scales (Chatfield, 1988; Fildes &

Makridakis, 1988). Scale-independent metric family has MAPE, sMAPE, MASE, and the MAE/Mean ratio etc.

In our situation, we predict web-traffic for approximately 145k Wikipedia articles which have a huge scale difference, i.e. the traffic mean range of train data in 803 days is [0, 2.141502e+07]. One of the lowest traffic is about “Artificial insemination”, while the highest is Wikipedia “Main Page - En”. It is apparently that scale-independent evaluation metrics are more appropriate for our project.

Among the evaluation metrics we choose the symmetric mean absolute percentage error (SMAPE), proposed by Makridakis (1993), which is a modified MAPE in which the divisor is half of the sum of the actual and forecast values. SMAPE has advantages such as scale-independency and interpretability. It also overcome the limit of MAPE that it puts a heavier penalty on negative errors than on positive errors. However, SMAPE has a similar limitation: the symmetric absolute percentage error is always equal to two for a zero actual value, regardless of the forecast that is used.

Since the prediction or true values are can be zero, we define SMAPE = 0 when the actual and predicted values are both 0:

$$\text{SMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

The boundary of SMAPE is between 0 and 200%. The smaller the SMAPE is, the better our model performs. For the training purposes, we may use MAE loss on log1p(data) which is smooth almost everywhere and close enough to SMAPE.

4. Data Cleaning and Feature Engineering

In order to achieve better performance of models and find some distribution pattern, we implement data cleaning and feature engineering, to explore the insight of time series data. Although the dataset is not messy but it has a lot missing values. We implement two methods to deal with missing values. Besides, we generated 4 new columns from “Page” column: name, language, access, agent. Afterwards, we created an new feature from additional wiki web pages.

- **Missing Value:**

Among the 145063 records, 27786 records contain missing values. We deal them by the following two methods according the needs of models:

Fill nan with zero: The majority missing values appeared at early stage of 803 days, which indicates those pages were not created at the beginning of 803 days, so it maybe not reasonable to fill them with mean or median. We fill them with zero and applied in all of our models.
Fill nan with the data of yesterday: For the test need of seq2seq cnn model, we fill the missing value of the date among the last 3 days with the data of the day just before that date.

- **Generating Features:**

The "Page" column has data format like this: "Milénico_es.wikipedia.org_all-access_spider ". "Milénico" is the name or id of this page, and in the whole dataset there are 145063 different names. "es.wikipedia.org" means this page is written by Spanish, and we have 7 different languages in our dataset. They are English, German, Spanish, French, Japanese, Russia, Chinese. "all-access" stands for the access, there are total 3 access ways, namely "all-access", "desktop", "mobile-web". "spider" is one of two agents which are "spider" and "all-agents".

We splitted the "Page" by "_" and extract 4 features: name, language, access, agent. We plotted the distribution of language, access, agent which can be clustered.

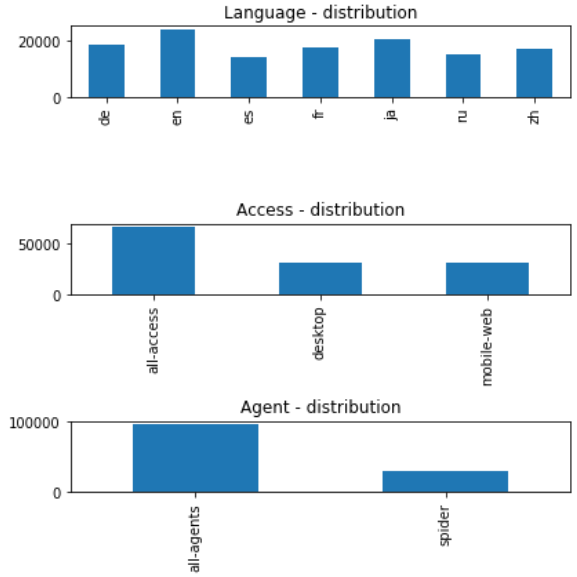

Figure 4: Distribution of 3 new features

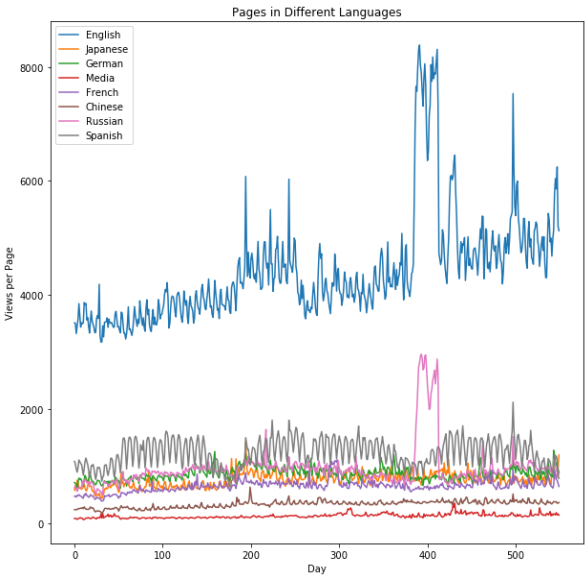

Figure 5: Detailed distribution of Languages

Some interesting pattern in the languages can be detected: English shows a much higher number of views per page because Wikipedia is a US-based site. The English and Russian plots presented a very large spikes around day 400 (around August 2016), with several more spikes in the English data later in 2016. The reason might be the effect of both the Summer Olympics in August and the election in the US. This structure indicates we may take the trends or hot topics into consideration. However this is very challenging and tricky: we have to find additional

information about popular events closing to the prediction periods, and we don't know how long this influence of these events can last. Besides, it maybe works on well clustered pages, maybe language or topic based clusters, but maybe not useful on a particular page.

- **Additional Page Content data：**

During building our baseline models on this problem, we come up with an idea that adding some more page information into these models might be able to make some improvement on predictions. Hence, we decide to grab more wiki page data from the website by a simple crawler program which is developed on our own.

5 Baseline Models

We tried 3 baseline algorithms: median, ARIMA, Ridge Regression. Because of the long running time, we downsampled our 140k dataset by random to 10k for some basic models. Ridge Regression is easier to be tuned so we apply the original dataset for it.

5.1 Simple Median Model

Median model is simply calculating the median of the past dates' hits value and make a constant prediction on the last 60 days. It mainly services as a warm-up model and helps us understand the dataset deeper. The average SMAPE error on the downsample dataset is 72% which is not that bad.

5.2 Arima

ARIMA is the synonym of an autoregressive integrated moving average model. ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.[2]

As mentioned above, part of dataset is non-stationary and suitable for ARIMA model. We used our downsampled dataset, splitted it, and took the first 573 days visits of each wiki page as train data, the following 180 days visits as validation data and the final 60 days visits as test data. After we trained and tuned the parameter**,** our average SMAPE of 10k wiki pages test data is 56.03%, which is not bad but still has room to be improved.

5.3 Ridge Regression

In addition to predict using the traditional time series model like ARIMA, our data can also be fit to a regression model. As a baseline, we chose ridge regression. Different from the ARIMA setting, we applied one model to the whole dataset and we split the data by samples. Using the cleaned data with additional features as described above, we split the data into train, validation, and test set with a ratio of 7:2:1. The target of the model are the last 60 days of page visits from

2017-07-13 to 2017-09-10, and the input are all previous 743 days of visits with added features. After scaling the features to range from 0 to 1, we fit the data into the model. We then tuned the L2 regularization parameter with the validation set and found the optimal value to be 0.01. The plot shows the validation performance under different alphas. By choosing the optimal hyperparameter, we re-trained the model with the combined train and validation data. The average test SMAPE is around 95%.

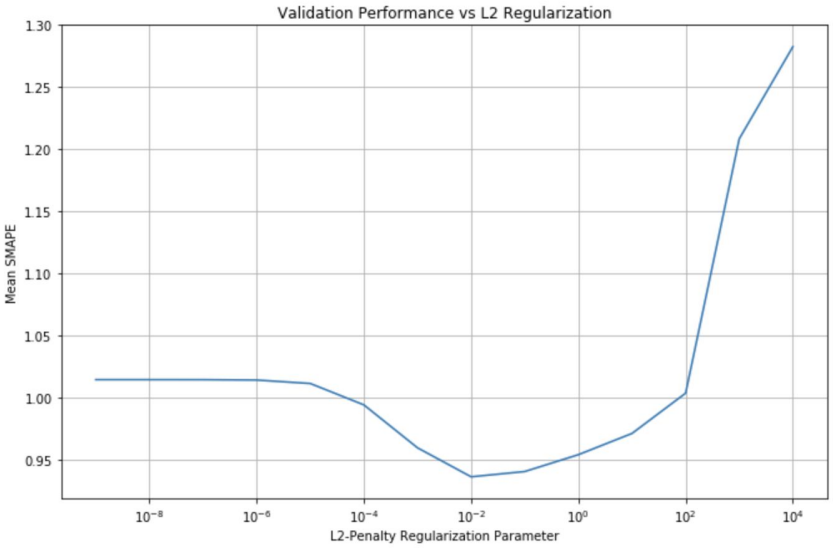

Figure 6: Validation Performance vs L2 Regularization

6 Baseline Improvements and Advanced Models

6.1 Arima Improvements

The **SMAPE** of ARIMA is not so bad, but according the sample plot, the ARIMA is way under expectation, especially for stationary data. This is because in ARIMA we forecast future traffic by the moving average, so the predicted web traffic will not deviate the median of real value a lot. And in the first one week there is some fluctuation in prediction but lately it becomes a line. **So ARIMA is suitable for short time (3-7 days) prediction but not 60 days.**

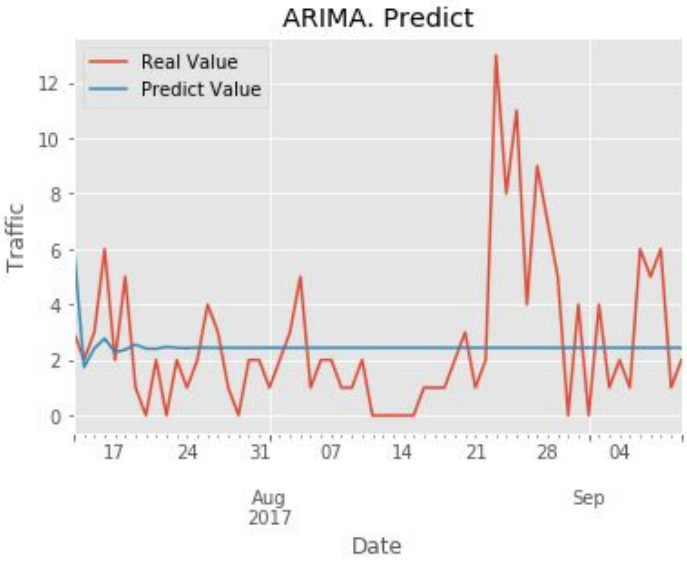

Figure 7: Sample Forecast of wiki page by ARIMA(60 days-test)

We also find the seasonal feature in our dataset so we tried an adjusted version of ARIMA algorithm, namely seasonal ARIMA (SARIMA). A seasonal ARIMA model is formed by including additional seasonal terms in the ARIMA models we have seen so far. It is written as follows:

$$\text{ARIMA}\ \underbrace{(p,d,q)}_{\substack{\uparrow \\ \text{Non-seasonal part} \\ \text{of the model}}}\ \underbrace{(P,D,Q)_m}_{\substack{\uparrow \\ \text{Seasonal part} \\ \text{of the model}}}$$

We used the same dataset for ARIMA, and tuned the parameters (p, d, q, P, D, Q), our average SMAPE of 10k wiki pages test data is 52.8%, which is slightly better than ARIMA model. In the sample plot of a same wiki page by ARIMA and SARIMA, the SARIMA version seems more reasonable.

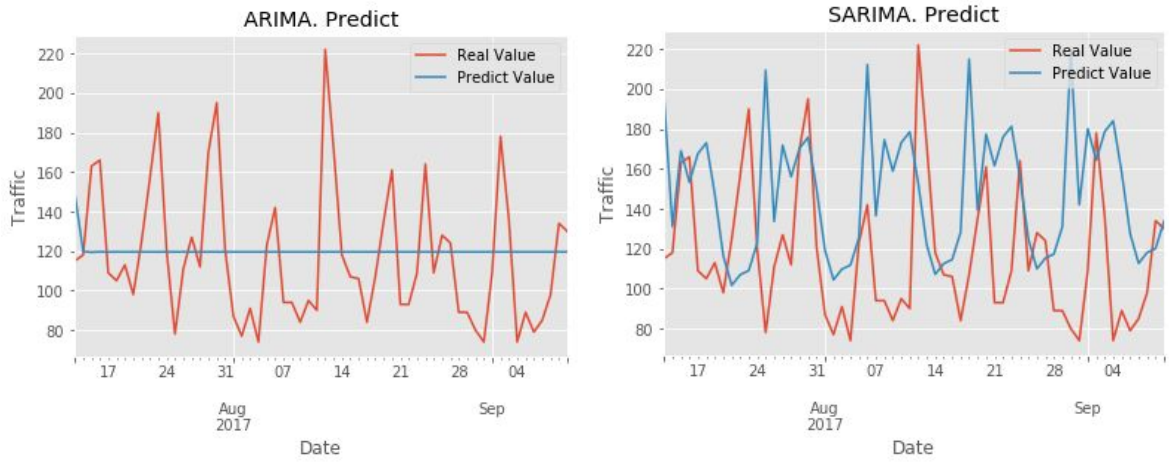

Figure 8: Sample Forecast of the same wiki page by ARIMA and SARIMA（60 days-test）

Except for adding seasonal factors into ARIMA, there are some other approach which is deserved to be tried :

1, While ARIMA can not take exogenous variables, we can not combined other features, for instance the article name, language, way of access etc. However, there are various extensions of ARIMA models that overcome this drawback including ARIMAX models, transfer function models, dynamic regression models, etc.

2, Classify the dataset into stationary and non stationary clusters. Apply SARIMA or ARIMAX models on the non stationary cluster, while applied other algorithm(e.g Ridge regression) on the stationary cluster.

6.2 Prophet

Prophet is an open-source time series forecasting tool developed by Facebook. [3] It is implemented in an R library, and also a Python package .Prophet works as an additive regression

model which decomposes a time series into (i) a linear/logistic trend, (ii) a yearly seasonal component, (iii) a weekly seasonal component, and (iv) an optional list of important days (such as holidays, special events, …). It claims to be "robust to missing data, shifts in the trend, and large outliers", which would make it well suited for this particular task.

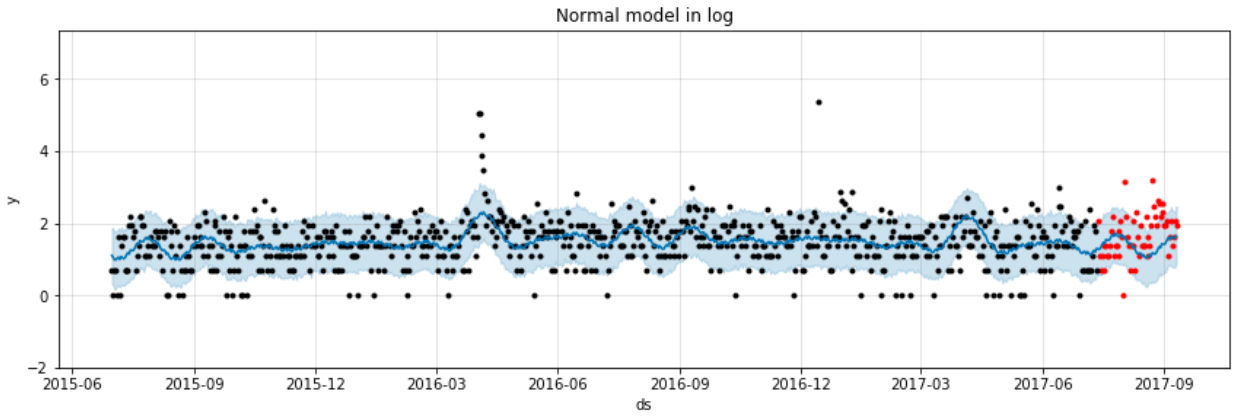

Figure 9: Sample Predictions by Prophet

We tried both linear input and logistic input with Prophet on downsample dataset, it separately gives us 68% and 63% mean SMAPE error. Since the logistic model has a slightly better result, we think it reduces the impact caused by outliers and missing data.

One of the advantage of Prophet is that it needs little mathematical and professional experience, since it doesn't need tuning parameter. However, it becomes a disadvantage when it gives unsatisfiable prediction results - there is little things the developers can do to improve the performance.

6.3 LSTM

Inspired by natural language translation models and how time series data resembles a sentence that is sequential and time-dependent, we built a simple RNN with one layer of LSTM followed by a dense layer. The idea of using the long short-term memory model is because it can adjust how much to remember data from long time ago. The model takes in a scaled data of the difference of number of visits with a lag of 1 and predicts the last 60 days of visits, which in fact, are the differences between consecutive two days. We then inverted the transformation and scaling to perform the evaluation. The train, validation, test split is the same as we did for the ridge regression. At first, our model performed badly on the validation data. The mean SMAPE was around 199% which is basically not predicting anything.

After debugging, we found that there were at least two reasons that resulted in our model not learning anything:

1) Since the raw output of the model, which is the difference, can be negative, the reverted prediction, which is the actual number of visits, is also negative for some sequences. However, the true value of visits will never be negative, thus increases the error between prediction and true values.
2) There might be exploding gradients in the network when backpropagating. Due to the LSTM nature that it handles vanishing gradients, we might need to manually control the gradients to go the other direction.

By setting all negative prediction to 0 and applying a clipping-norm to 1, the test SMAPE got down to 168%. Though this was not performing well as we expected, we believe there exist many details that could affect the overall performance of a deep neural network. Due to time constraint, we would try out smaller batch size as 1 instead of 10 and a different loss function for improvement.

6.4 Seq2seqCNN

Seq2seq is natural for time series and achieved good results according previous works. In traditional CNN model, computations over all elements can be fully parallelized but can not be sequenced. Thanks to the work by Jonas Gehring(2017)[4], we can apply an entirely convolutional sequence to sequence modeling in our project. A single neural network was used to model all 145k time series. The model architecture is similar to WaveNet, consisting of a stack of dilated causal convolutions, as demonstrated in the diagram below.[5]

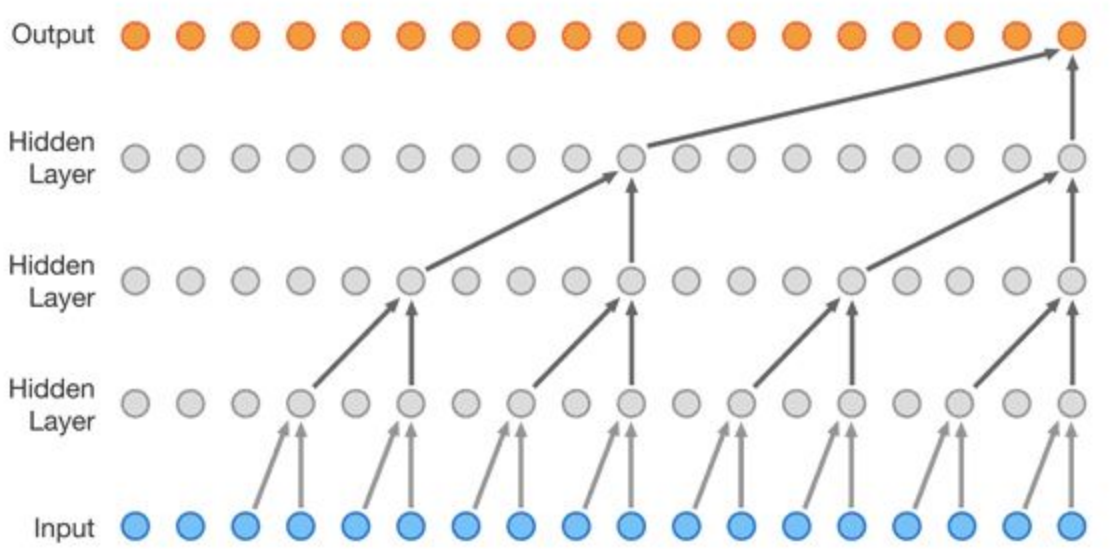

Figure 10: WaveNet diagram

We took advantage of the 6th position model by sjvasquez[6] from Kaggle test, and made a few modifications to add our additional feature inside and adapted the model to generate coherent predictions for the entire forecast horizon (60 days). We adopted a sequence to sequence approach where the encoder and decoder do not share parameters. This allowed the decoder to handle the accumulating noise when generating long sequences.

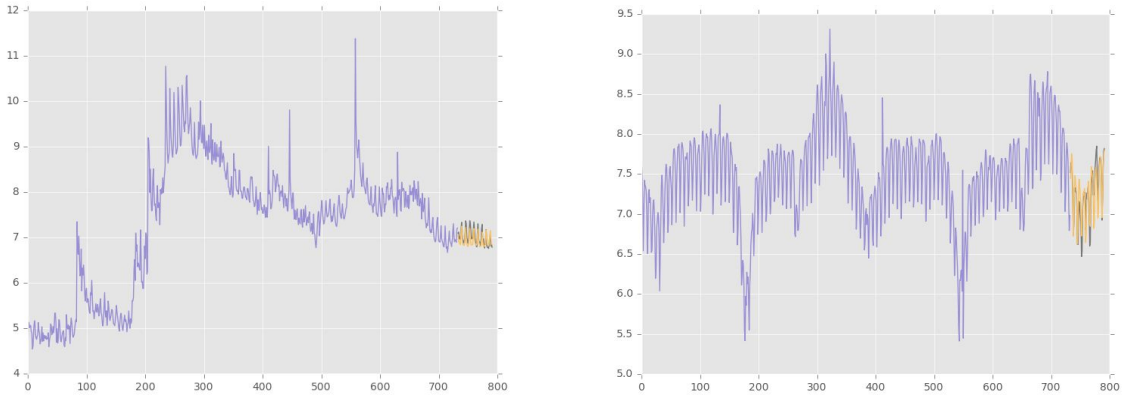

Figure 11: Sample predictions(Forecasted values - yellow; truth values - grey. The y-axis is log transformed.)

Ours didn't overtake the original version by sivasquez. The validation SMAPE of our model was 37.59% which was lower than the original version which got 35%.And the test SMAPE was 41.3%. We guess two reasons are responsible for the result: 1, half of missing values in the added feature; 2, the references will not increase the wiki page visit, but the increasing of wiki page visit will lead users to visit these reference.

7. Conclusion

In the following table we can found the Seq2seq CNN achieved the best SMAPE of Test Data.

Models	Dataset	SMAPE of Test Data
Simple Median Model	Downsampled Dataset (10k rows)	72%
Arima	Downsampled Dataset (10k rows)	56.03%
Ridge Regression	Downsampled Dataset (10k rows)	95%
SArima	Downsampled Dataset (10k rows)	52.8%
Prophet	Whole Dataset (140k rows)	63%(logistic input)

LSTM	Whole Dataset (140k rows)	168%
Seq2seq CNN	Whole Dataset (140k rows)	41.3%

Except some reasons we mentioned in the individual model above, we draw out a couple of further analysis and conclusions:

- Traditional regression models are not suitable for time series data even though we used downsampled dataset. In time series data, the current time period will be influenced by its value in the previous period, even postier period. As a result, autocorrelation in the residuals will occur when fitting a regression model to time series data, In this case, the estimated model violates the assumption of no autocorrelation in the errors, and our forecasts may be inefficient. More often than not, time series data are "non-stationary" which can lead to spurious regressions. Cases of spurious regression might appear to give reasonable short-term forecasts, but they will generally not continue to work into the future.[7]
- Arima family works for a couple sequential of time series data, i.e. several wiki pages in shorter time periods, but is not a potential candidate for huge sequential of time series data like 140k row. Because the model is based on the same sequential input. And tuning the parameters for one sequential takes a lot time (100-100s), which is not efficient for our dataset.
- Seq2seq NN is natural choice for time series data. The LSTMs is supposed to be a good model for our forecasting, because through its ability to process and predict time series sequences without forgetting unimportant information, that is to avoid "Long-Term Dependencies".[8] General RNNs or Arima failed to learn to connect the inputs. However, our model didn't make it because we didn't find the best fitting parameters due to the time constraint.
- Seq2seq CNN combines advantages both encode-decode and CNN. Convolutional networks do not depend on the computations of the previous time step and therefore allow parallelization over every element in a sequence. This contrasts with RNNs which maintain a hidden state of the entire past that prevents parallel computation within a sequence. Seq2seq CNN can be fully parallelized and achieve good performance.

7. Challenges

This project provides us a precious opportunity learning basic time series prediction techniques and applying machine learning knowledges on real world problem. During the process, improving baseline models performance, debugging and tuning complicated NN models and crawling page data from wikipedia websites become most challengeable part through the whole project. For the baseline models and NN models, we figured out some solutions in the end, but,

for the spider program, we failed to bring data coverage degree to a higher level which, to some extent, influences our NN models' results. It's definitely a good part to be worked on in the future.

8. Future Work

- Digging into NN models and Tensorflow APIs to enhance our NN models' performance, which is proved that they can come out fairly good results.
- Crawling specific page content data, more than just references count.
- Cluster data and try ensemble methods.
- Test our NN models on other time series data forecasting.

9. Reference

[1] *https://en.wikipedia.org/wiki/Web_traffic*
[2] *https://en.wikipedia.org/wiki/Autoregressive_integrated_moving_average*
[3]*Prophet: forecasting at scale. Sean J. Taylor, Ben Letham*
[4] *Convolutional Sequence to Sequence Learning. Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, Yann N. Dauphin*
[5][6] *6th placesolution, https://github.com/sjvasquez/web-traffic-forecasting*
[7] *Regression with time series data https://www.otexts.org/fpp/4/8*
[8]*Sequence to Sequence Weather Forecasting with Long Short-Term Memory Recurrent Neural Networks*