Zishen Wan, Feiyu Chen, Queena Zhou
Prof. Ba
Decision Theory
10 May 2019

Web Traffic Time Series Forecasting

INTRODUCTION

Time series data are gathered everywhere in our daily life, from online webpage traffic to airline passenger volume. Time series data are not only prevalent but also priceless; correct usage of them could make the future foreseeable and hence lead to big success. Big technology companies like Facebook invest a large amount of time and money into developing tools, such as Prophet, to forecast time series data (Taylor and Letham). Each member of our group is intrigued by the value of forecasting time series data and wants to master the techniques to perform it. We, therefore, select a dataset that contains web traffic time series data from Wikipedia and use part of it to predict the rest. We use symmetric mean absolute percentage error (SMAPE) to measure our forecasting accuracy. We investigate different models and techniques of forecasting time series data and try to learn why they work or not work.

DATA

We gather our data from a Kaggle competition. The dataset contains web traffic views of approximately 145 thousand Wikipedia pages from July 1, 2015, until September 1, 2017. After an exploratory analysis on our data, we find that the dataset counts web traffic for Wikipedia articles in seven different languages. It also differentiates web traffic based on devices (desktop and mobile-web) and accessing types (spider and all agents).

We wonder whether the language that each article is in will affect the view counts of that article. From Figure 1, we confirm that the total numbers of views per day do depend on languages. In particular, articles in English has higher views than other languages. We also see that Spanish articles have strong weekly patterns and that English and Russian plots show large spikes around day 400.

Figure 1. The total number of views per day for different languages

Since Figure 1 shows some periodic structures, we use Fast Fourier Transform (FFT) to further explore weekly pattern. The peaks in FFT show us the strengths of periodicity. We see that Spanish articles, shown in Figure 2, share the strongest periodic features. All the languages have periodic features at 1 and 1/2 week. This is not surprising since browsing habits may differ on weekdays compared to weekends.

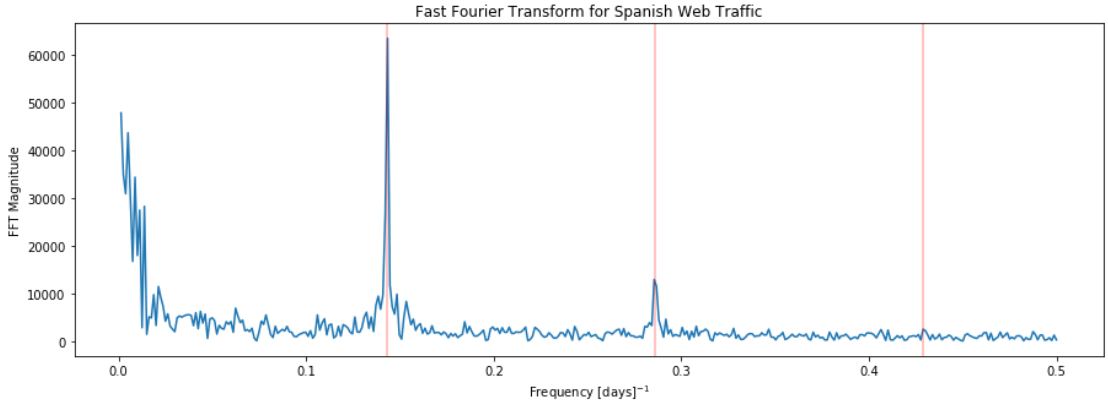

Figure 2. Fast Fourier Transform for Spanish Web Traffic

Yearly patterns in these view counts can also be discovered by autocorrelation. After random shuffling the data and splitting them into five batches, we graph the average autocorrelation of each batch with respect to the lags, shown in Figure 3. All five batches exhibit large autocorrelations at lags around 365 and 730, which are multiples of the number of days in one year. Therefore, the data also have yearly seasonal effects.

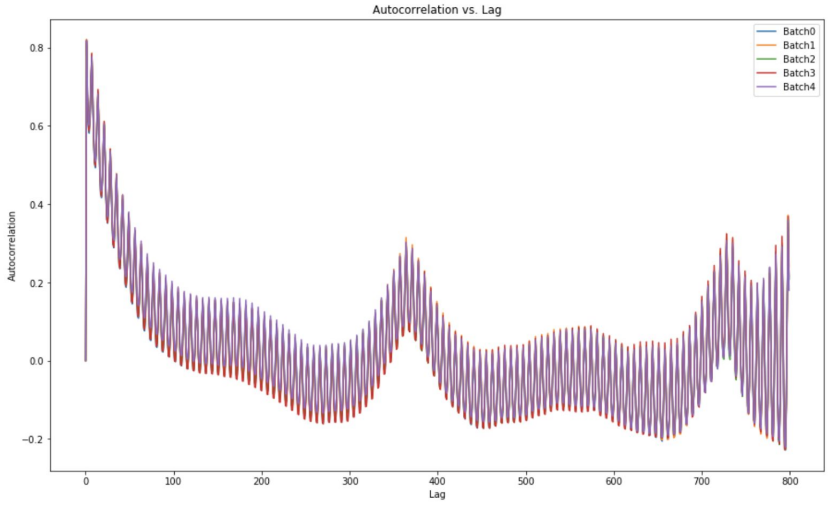

Figure 3. The average autocorrelation of each batch with respect to the lags

Stationarity is an important feature of time series data. Using Dickie-Fuller test, we find 87.54% data to be stationary, meaning that they each have a constant mean, variance and also autocorrelation overtime. This contradicts with Figure 3, which shows strong time-dependent autocorrelation for all data. After inspecting both our code and data, we figure out the cause of

Figure 3 showing changing autocorrelation. It is due to the shuffling of our data before we take the average of the autocorrelation. The approximate 13% data which are not stationary probably have strong trend and seasonality, and they are spread out in the five batches affecting the average autocorrelation.

METHODOLOGY

The setup of the modeling problem is: given web traffic of each Wikipedia article from July 1, 2015 until July 31, 2017, we forecast the web traffic of each article from August 1, 2017 until September 1, 2017. We then compare our forecasts with the true data in these two months using SMAPE, which is defined as:

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

SMAPE is favored here instead of Mean Squared Error (MSE) or Mean Absolute Error (MAE) because MSE and MAE will penalize base on the absolute error which will effectively ignore the prediction error of articles with little views. This is not what we want since most of the articles in our dataset don't have large numbers of views.

1. Baseline

   We start with setting up a baseline model so that we have a sense of what minimum prediction performance we should strive for. Our baseline model is that we use the median of an article's past two months' traffic to forecast its traffic in the next two months. Here we choose the median rather than the mean because we have many random spikes in our dataset which are essentially outliers. These spikes can distort the mean traffic and result in positive bias in our forecasting while the median is less affected by these spikes. We choose the last 2 month instead of the last 2 years as the window to calculate the median because the traffic last 2 months are more indicative of the current popularity of an article. As an example, some articles of trending events may have had high numbers of views in the past but have lost their popularity. For these articles, using a smaller window for the median calculation results in more accurate future traffic estimation. The following table shows the accuracy of each of the baseline models.

Table 1. Comparing the baseline models

Baseline Models	Mean of the past 2 years	Median of the past 2 year	Median of the past 2 month
SMAPE	69.5	52.8	39.7

2. ARIMA

If our objective is to predict the change in the web traffic so that we can make preparation for its future increase/decrease, then we need a model that can capture the trend and seasonality in our dataset rather than predicting a single value all the time. We choose Auto-Regressive Integrated Moving Average (ARIMA) as the first model to explore. Auto-Regressive Moving Average (ARMA) assumes the time series has the following property:

$$X_t = c + \varepsilon_t + \sum_{i=1}^{p} \varphi_i X_{t-i} + \sum_{i=1}^{q} \theta_i \varepsilon_{t-i}.$$

where X_t is our variable of interest at time t in the time series and ε_t is the noise at time t. X_t is assumed to be a linear combination of its p previous historical values and its q previous noise. ARIMA is like ARMA applied to the difference (X_t - X_{t-1}) of a time series rather than the time series itself (X_t). In our model, we choose the hyperparameter p and q to be 7 because we want the model to focus on the traffic of the past week. Estimating the parameters for ARIMA is time-consuming because we have 145 thousand articles and we have to fit an ARIMA model for each article. Therefore, we only tested ARIMA on 1% of the articles that have the highest number of views (in terms of the median views from the past 2 years). Unfortunately, the prediction accuracy of ARIMA is less accurate than our baseline model on the same subset of articles.

Table 2. Comparing ARIMA and the baseline model

Models	Median of the past 2 month	ARIMA
SMAPE (on a subset)	38.8	55.4

ARIMA-based methods do not outperform our baseline model. We think the reason is that the most-viewed time series in our dataset are nonstationary so that the assumptions of ARIMA are not met.

3. Encoder-Decoder GRU

We notice in our exploratory analysis that most of the web traffic time series in our dataset do not exhibit a strong trend or seasonality. Another thing we notice is that for those web traffic time series that do have obvious patterns, their patterns are very different from each other. In other words, they are generated by different processes. Therefore, we need to take into account this observation, which means that either (1) we need to use some predefined criteria to categorize the time series in our dataset, or (2) we need to use a model that is flexible enough to learn different patterns for different data. We go for the second path and the model of our choice is encoder-decoder RNN. We end up using an encoder-decoder GRU with 200-dimensional latent space. We use the data of

the past 200 days as the input and try to predict the traffic of the next 60 days. The reason for choosing GRU instead of vanilla RNN or LSTM is that: (1) compared to vanilla RNN, GRU can better capture the long term dependency in the time series (2) compared to LSTM, GRU is computationally cheaper (Keras even has a GRU optimized for GPU which is even faster).

For the input, we use the following features:

- the web traffic of this particular article on this particular day
- the median web traffic of this particular article
- the web traffic of this particular article a week ago
- the web traffic of this particular article a quarter ago
- the web traffic of this particular article a year ago
- all categorical features (accessing device and type, language) one-hot encoded.

We use the median traffic as one of our features because articles with very high and very low views show different patterns. For example, articles with very high views are more likely to show a strong weekly pattern. We also add all categorical one-hot encoded values as features because time series of different accessing devices, accessing types and languages behave differently as well. We use web traffic a week/quarter/year ago as our features because we want the neural network to notice the seasonality in our data. Theoretically, GRU should be able to detect those patterns by itself, but in practice, it works better if those features are provided to GRU.

Before feeding our data to the neural network, we first log-transform the daily views so that the long-tailed distribution of the daily views become more 'normal'. We then normalize the log-daily-views to make the optimization faster and reduce the chance of being stuck in local minima. We choose to use MAE as our loss function to approximate SMAPE.

Our encoder-decoder GRU model reaches a SMAPE of 38.2.

4. Prophet

Prophet is a model for predicting time-series data released by Facebook. It is based on a self-additive model that is used to fit nonlinear trends such as year, week, season and vocation. After shuffling our data and randomly choose 1000 article, we implement Prophet model on them and find the SMAPE is 97.43, which is larger than our baseline and aforementioned model. We consider that the reason for high SMAPE of Prophet is most of our data have low traffic and no strong seasonality, however, Prophet is more suitable for data with strong periodicity.

CONCLUSION AND FUTURE WORK

Our project aims to forecast the web traffic of Wikipedia. We start with a baseline that uses the median of the previous 2 months as the prediction. We then explore ARIMA and

encoder-decoder GRU to capture the seasonality and trend in our data. ARIMA performs worse than our baseline in this dataset because most of our series don't show a strong simple pattern. Encoder-decoder GRU reaches a SMAPE of 38.2, which is higher than our baseline but the improvement is limited (from 39.7 to 38.2). The first place solution in the Kaggle competition where we get our dataset reached a SMAPE of 35.5 with a model similar to the encoder-decoder GRU we use (but with more complex feature engineering and hyperparameter tuning). This implies that forecasting web traffic is not an easy problem. We think part of the reason of why this is so difficult is that many of the changes in web traffic are by nature unpredictable. For example, we cannot predict what time in the future will someone web-scrape Wikipedia. Another part of the reason is that some of the predictions require prior knowledge of the mechanism which generates the views. For example, if we know the next season of some popular TV show will premiere next month, then the the traffic to its related articles is sure to rise. Therefore, we think one improvement that can be made in the future is to bring external data source into the models. Even though there are things to be improved upon, we think we have learned a lot from this project. During the process of applying statistical tools and building neural networks, we achieve our goal of forecasting time series data.

Works Cited

Taylor, Sean J., and Benjamin Letham. "Forecasting at scale." The American Statistician 72.1 (2018): 37-45.

Kaggle Competition. https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/43795#latest-528073

Appendix I

Our code for this project is available on GitHub: https://github.com/feiyu-chen96/am231-project

Description of files:

1. EDA.ipynb: exploratory analysis of our web traffic data
2. Autocorrelation.ipynb: Implement autocorrelation to analyze weekly and yearly pattern
3. Baseline.ipynb: Baseline model (median model)
4. ARIMA.ipynb: Auto-Regressive Integrated Moving Average (ARIMA) model
5. Encoder-decoder GRU.ipynb: encoder-decoder RNN(GRU) model
6. Prophet.ipynb: Prophet model
7. Moving-average.ipynb: moving average (MA) model