Wikipedia Web Traffic Time Series Analysis

Gaurav Gopishetty, Michael Wang, Mona Yao, Serkan Comu

Introduction

Time series are one of the most commonly used data types in the real world. During the study in this semester, we have learned a lot of characteristics of the time series data and different techniques to deal with such data.

In this project, we are going to apply what we learned this semester on a real-world practice – Wikipedia web traffic time series data. We will combine the techniques we learned with the application and solve the problems we meet in the process.

In this report, we are going to talk about several analysis based on the Wikipedia web traffic time series, which includes web traffic pattern recognition and web traffic prediction using different methods.

Data Description

The data is from a Kaggle competition[1]. The original data contains 6 files, while what we used most in the file under the name of "train_2.cvs". The file contains 145,063 records and the time ranges from July 1st, 2015 to September 20th, 2017. Each of the records represents a number of daily views of different Wikipedia articles.

Another thing need to be noticed is that the data source for this dataset doesn't distinguish between the traffic values of zero and missing values, which means a missing value may mean the traffic was zero or the data is not available for that day.

Pattern Clustering

After talking with the instructor in the first discussion, we have been thinking is there any underlying patterns in these time series, and can we group the times series by these patterns. With this idea in the mind, we conducted an experiment to manage to cluster the time series by their patterns.

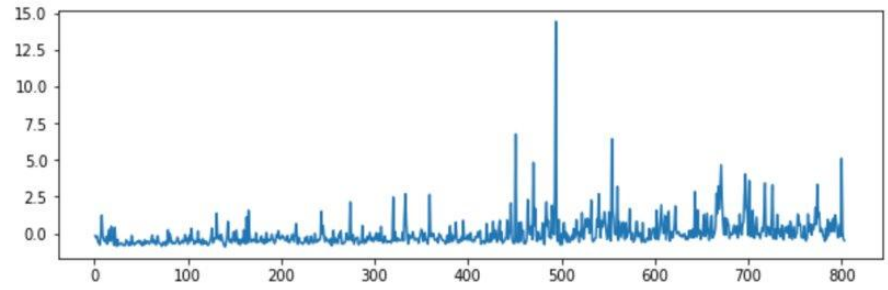
Data Exploration

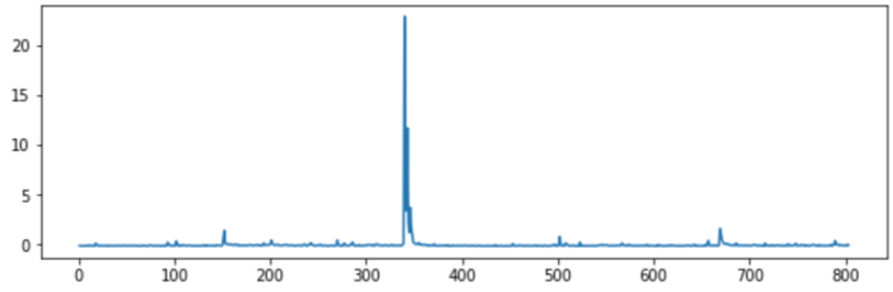
In order to understand the underlying patterns, we need to look at different time series in the data first. After plotting around 100 time series in the dataset, we have found some underlying patterns behind these time series. The following plots show some examples we picked from our

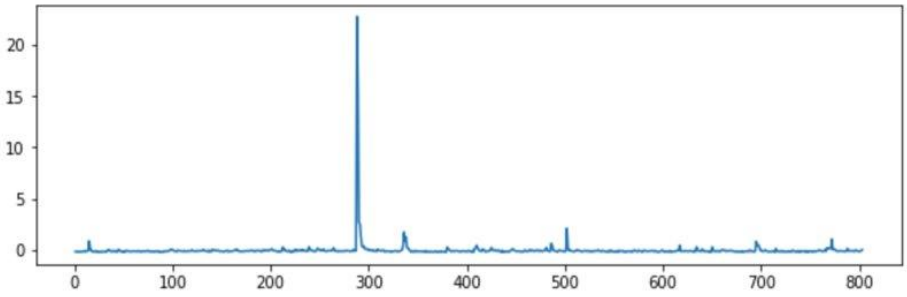
[1] https://www.kaggle.com/c/web-traffic-time-series-forecasting/overview

dataset. From these plots we can observe some patterns even though the patterns are not very obvious.

However, we also found some challenges by grouping different time series in one pattern together. For example, in the second plot and the third plot below, the patterns are similar for us, because there's only one obvious peak in the whole time series for both plots. However, because the peaks happened in different phases of time, we don't know whether the model would group them together. In addition, because the data was collected daily, there are a lot of variations in the time series. So, we need to find a way to help us smooth the data and ignore the phase difference.





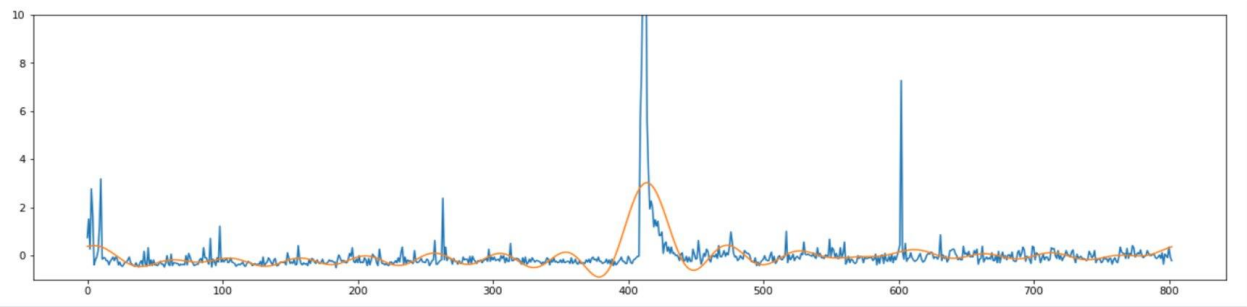


Data Preprocessing

After doing some research, we found Fourier transformation may solve our problems here. In one word, Fourier transformation decomposes the signal into its constituent frequencies.

After the decomposition, we can smooth the time series by removing those high frequencies. Besides, transforming the time series into its frequencies helps us ignore the time phase difference as well.

The following plot shows an example in our dataset. In the plot, the blue line represents the original time series, and the orange line is the inverse transformation from Fourier transformation. We can see the orange line depicts the pattern of the blue line very well by removing those high frequencies.

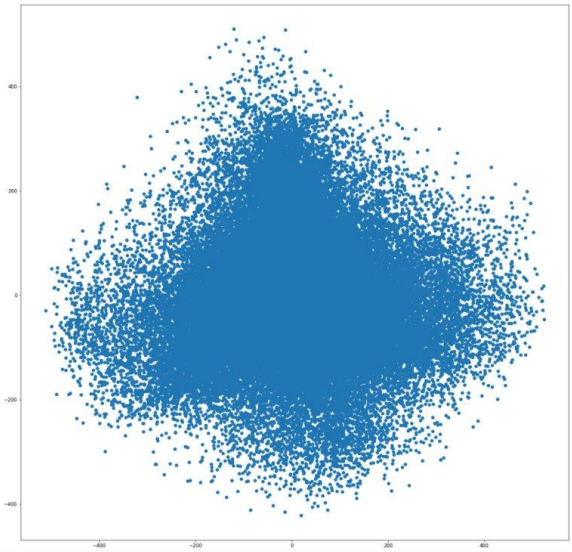


After we applied Fourier Transformation on our dataset, it gave us a list of frequencies for each time series. Here, we set the threshold as 0.02, which means we dropped any frequencies higher than 0.02. After dropping the high frequencies, there are 17 frequencies left in the list and each frequency is composed by a real number part and a complex number part. We simply decomposed every frequency by these two parts, and we have 34 numbers in the list for each time series now.

Dimensionality Reduction

We could use the 34 numbers as the total descriptors for every time series. However, 34 descriptors are still too many, so we decided to reduce the dimensionality.

First, we tried PCA to reduce the dimensionality. The following scatterplot shows the result from PCA. From the plot we can see, the performance of PCA wasn't good, and the data are mixed into a big cluster. We can also evaluate the performance by looking at the screenshot below. The screenshot contains a variance ratio list of PCA. The first two components together explain only 0.2 of the total variances.

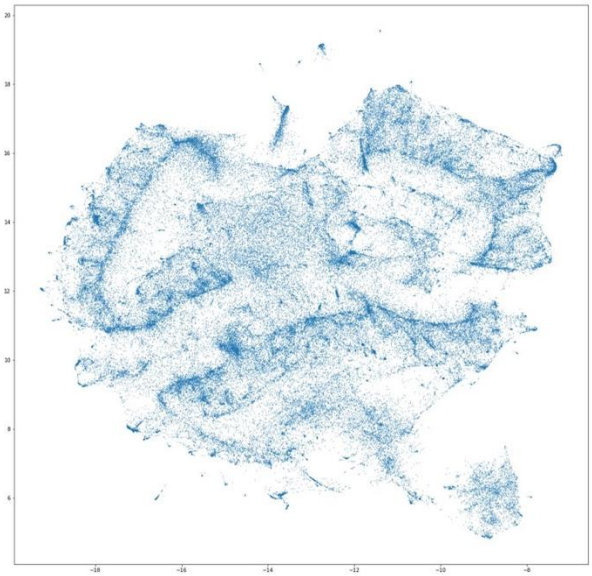


```
[1.38298516e-01 1.01017333e-01 7.46693514e-02 7.00236086e-02
 5.96037414e-02 5.36102120e-02 4.19405143e-02 3.93939370e-02
 3.56180832e-02 3.22240101e-02 3.04116973e-02 2.74661132e-02
 2.49390674e-02 2.33852194e-02 2.31600022e-02 2.13988218e-02
 2.11834387e-02 1.82669173e-02 1.77582022e-02 1.63493455e-02
 1.59132433e-02 1.43476273e-02 1.41185789e-02 1.29713909e-02
 1.17036755e-02 1.12694336e-02 1.01096823e-02 9.37637863e-03
 8.19636909e-03 8.01770695e-03 7.03829041e-03 6.21949097e-03
 2.91644169e-32 6.50802226e-34]
```

Because PCA is a linear dimensionality reduction method, we were considering trying some non-linear methods to see whether it will work better.

Then, we found UMAP. UMAP is a dimensionality reduction technique for visualization. It works very similarly with t-SNE but offers a number of advantages over t-SNE. UMAP constructs a high dimensional graph representation of the data then optimizes a low-dimensional graph to be as structurally similar as possible.[2]

The following scatterplot shows the result after applying UMAP. In the plot, it's easy for us to tell there are some clusters in the dataset.



Pattern Clustering

After the dimensionality was successfully reduced, we used DBSCAN to show those clusters out. the result shows there are 357 clusters in the data, and there are also 11,313 noises as well. The following graph shows a part of the result from DBSCAN. For a better view of the graph, we only plotted the clusters with more than 100 records.

[2] https://pair-code.github.io/understanding-umap/

As what we described above, as an experiment, we have successfully clustered the data into different groups by their underlying patterns. However, there are a lot of implements we can do to improve the cluster result. On one hand, rather than DBSCAN, we can try other clustering techniques like hierarchical clustering in our data to see whether the clustering performance will improve. On the other hand, we can use some other features like mean, trend, variance to see how it will impact the result of the final clustering.

Web Traffic Prediction without Additional Variables

One of the features of this dataset is that it contains web traffic information of Wikipedia pages in 7 different languages: English, Chinese, German, Spanish, French, Japanese and Russian. If we search through this dataset for Kobe Bryant, then we would be able to find all the web traffic data related to him in different languages.

In addition, the web traffic was also recorded based on different access, either via mobile phones or desktop computers. For simplicity, this part of the project aggregated both traffic types and obtained the total webpage traffic regardless of access type.

The goal of this part of the project was to sample a few interesting topics from this dataset and first plot out the time-series data and observe the web traffic patterns for each of these topics.

In particular, 4 specific topics were chosen, “TFBOYS”, “Angelababy”, “Kobe Bryant” and “iPhone”. In case you are not familiar, the first 2 topics are all Chinese celebrities. Our group thinks that it would be interesting to know the web traffic patterns of these Chinese celebrities on Wikipedia. To mix things up, we have also included a western celebrity, “Kobe Bryant” in the list. Because we are interested to see how different the web traffic patterns would be between

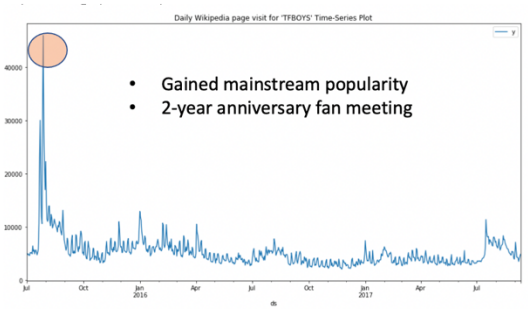
humans and products, so “iPhone” was selected as the last topic to explore, which is also a celebrity in some sense considering its popularity.

Data Preprocessing

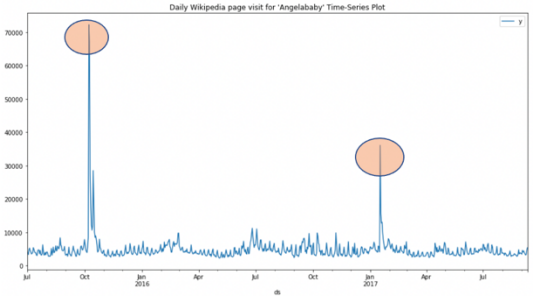
We began by doing some simple data transformation, originally, there were many dashes and underscores in the original page names, so through some simple string cleanup, topic selection, aggregation and performing data frame transpose, we arrived at a cleaner data format.

The graphs below show the initial plots of the time series data. Shown on the left is “Tfboys”, while right side is “Angelababy”. As we can see there are a few obvious peaks, which are circled in orange. For example, for the TFBOYS plot, we can see there’s a peak around August 2015, so we searched what happened in August 2015, and not surprisingly, TFBOYS held a 2-year-anniversary concert in Beijing that attracted a ton of interest from people all over the world. And If we look at Angelababy’s plot, we see two peaks in October 2015 and January 2016, the first peak happened when she got married and the 2nd peak occurred when she had her baby.

TFBOYS

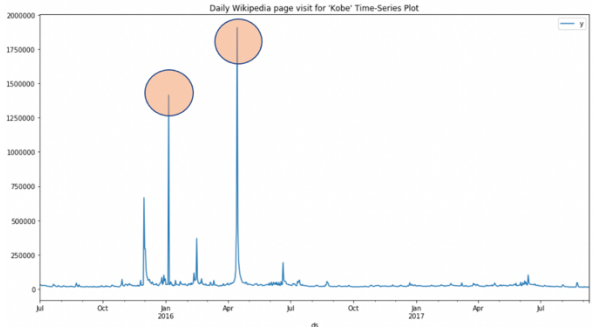


Angelababy

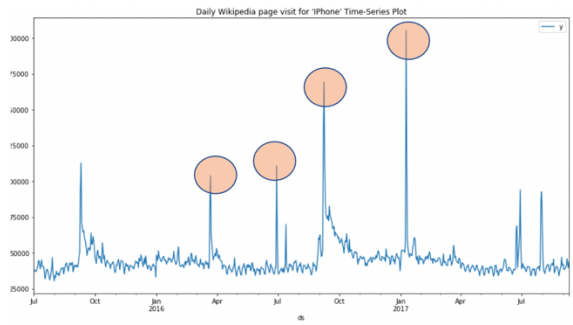


Moving on to Kobe Bryant and iPhone, the two peaks shown on Kobe’s plot are respectively when he had his last game in Sacramento, the capital city of California, and 2nd peak was the last game for his NBA career. Finally, iPhone seems to have a lot more peaks and most of these peaks occurred when Apple had a new product event, which usually is in spring, summer or fall.

Kobe Bryant

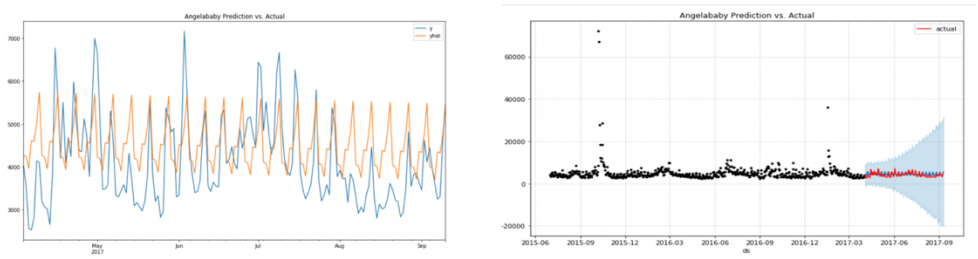


iPhone

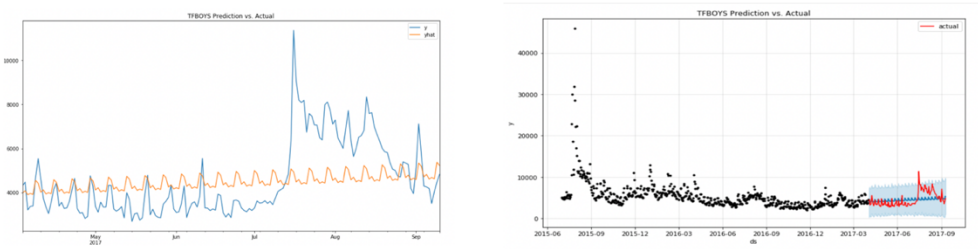


Model Evaluation and Results

we have decided to use Fb-prophet to make some predictions on the web traffic of these topics. In total there are 803 observations for this 2.5 year of data. we split the data to 80-20 and used the first 642 records as training and 161 record as testing.



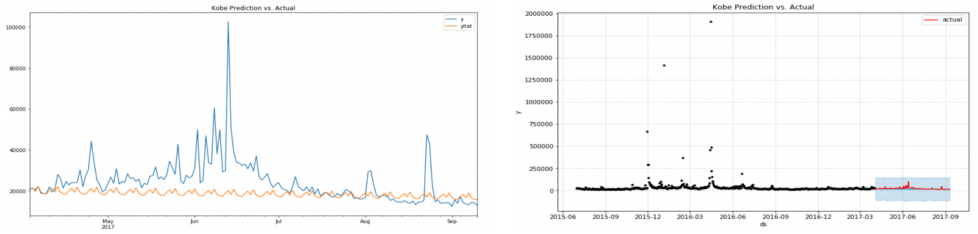
On Average there is a 20.869% difference between prediction and true value



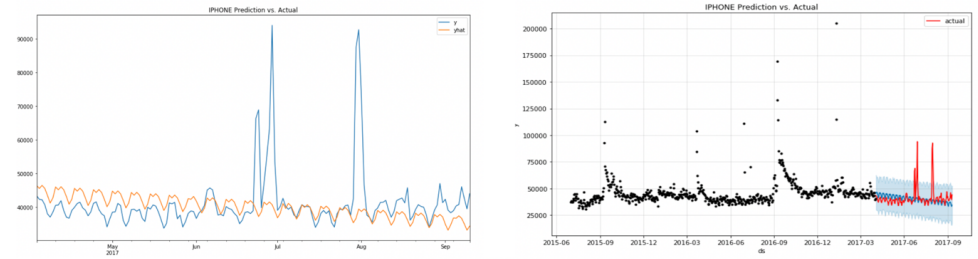
On Average there is a 25.744% difference between prediction and true value

The top half of the graph shows the prediction result for Angelababy, the left chart is basically a zoom in version of the right graph. On average, there's about 21% percent difference between the true value and the prediction, which is not so great, comparing to the predictions we made earlier in the mini project using Home-Depot google search data, which is around ~6.5%.

At the bottom, it shows TFBOYS prediction, as we can see it fails to catch the peak that was in the dataset and the result is even worse comparing to the result of Angelababy.



On Average there is a 23.095% difference between prediction and true value



On Average there is a 10.345% difference between prediction and true value

The prediction for Kobe Bryant suffers from the same problem, it failed to catch the peak that appeared in the testing data.

Finally, for iPhone, although it failed to catch the peaks, with an average error of 10%, the error was much better as compared to predicting the rest of the topics. We have tried to add-in dates when Apple release new products, because there's a parameter in Fb-Prophet called holidays, where we can set specific dates for special events, it helped to catch the peaks last time for Home-Depot, but this time, it did not improve the result.

In conclusion, predicting web traffic for a celebrity is somewhat like predicting stock price, there's so much unpredictability in terms of what explosive news might happen and it will greatly impact on the web traffic.

However, these time series plot, although hard to predict, can still give us a lot of useful information. We can use the peaks to trace back to what happed at a specific time point and gain information about the topic.

Predicting the web traffic for products can generate more reliable results because it does not have much explosive news associate with it and usually there's more obvious seasonality and trends involved.

For the future work, we can try to give a score to each of the clusters presented in the previous part base on how hard the data is to make predictions on. So, in the future, we can know ahead of time what topics are more likely to generate good prediction results.

Web Traffic Prediction with Additional Variables

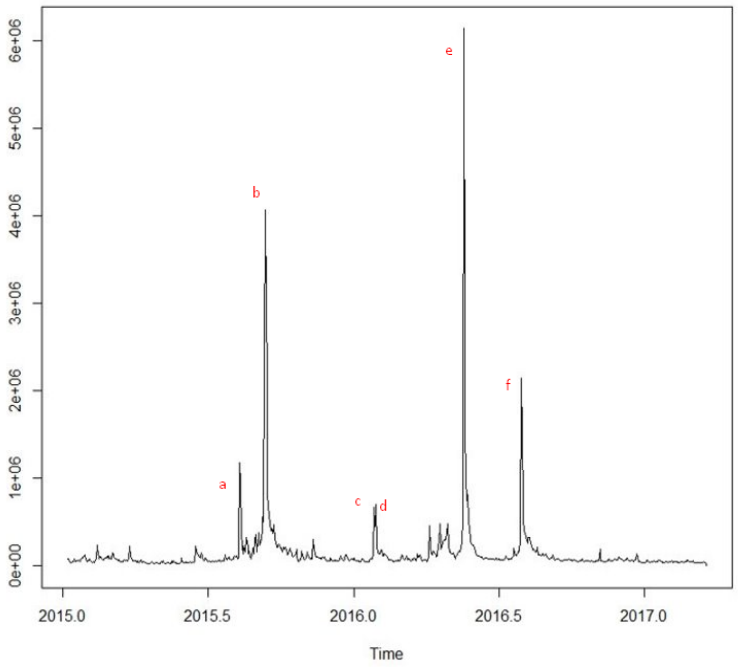
As mentioned in the past analysis, we weren't able to make predictions that were the result of external news. However, a site like Wikipedia would be able to set up a system to scrape the web for news stories about its different articles. Therefore, for this analysis, we wanted to build a model that could predict what spikes were likely to happen and how the general trend was going to move from additional variables.

Data Preprocessing

We began by doing some data manipulation. Similar to the methodology of the model before, we began by separating the long string of the title into the title, region, access points and agents. Then, we filtered the dataset to only include English Wikipedia articles.

Donald_Trump_en.wikipedia.org_desktop_all-agents	Donald_Trump	en	desktop	all-agents	35886	37554

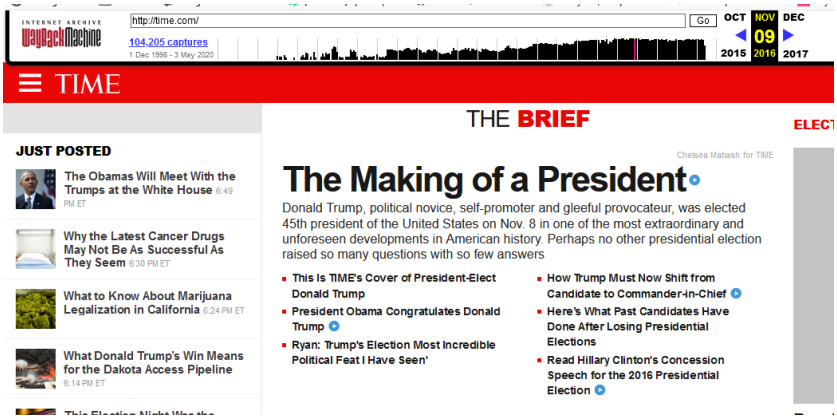
After looking at the data left, we decided to do this analysis on Donald Trump as he would surely have many news articles about him. Sure enough, as you can see in the graph below, he has numerous peaks in his articles popularity that coincide with different news stories.



Spike A is when he won his first caucus. Spike B is when John Oliver first did a piece about him, taking his candidacy seriously. Spikes C and D are from the beginning and end of the GOP convention. Spike E is when he got elected. Spike F is his inauguration.

It definitely seems like different news stories have impact on the Wikipedia articles traffic numbers. But how do we quantify this. The solution we found was to take a look at time.com in Wayback Machine and give the events scores out of 5 in terms of importance. The top news stories got a 5, if the news was in the top 10 it received a 3 and so on.

In this picture, you can see an example of what we found on November 9th, 2016.

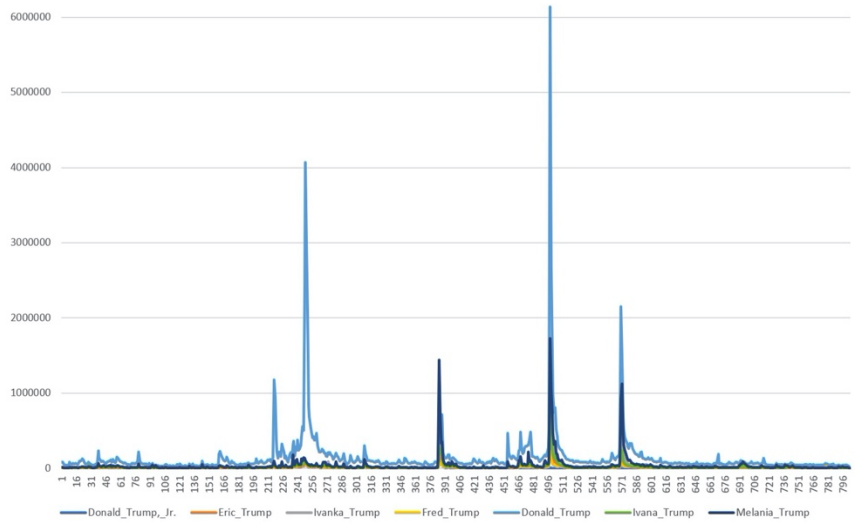


As you can see, the news story about Donald Trump's win is at the top. Thus, this event got a 5.

As another example the Iowa Caucus received a 1 as it was replaced by news of the Zika virus and did not see headlines much that day.

After this we took a look at the similar articles. To simulate what wikipedia might do, we simply took the information of every Trump family member, namely Donald Trump Jr, Eric Trump, Ivanka Trump, Fred Trump, Donald Trump, Ivana Trump and Melania Trump.

This gave us a graph like so:

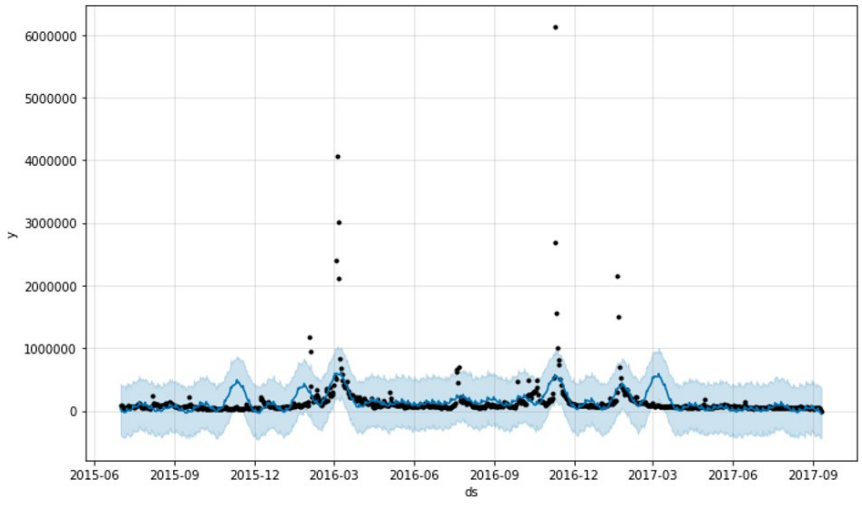


As you can see, the peaks are all in quite similar places to each other and the data follows a similar pattern all around. This is likely because of the existence of hyperlinks in Wikipedia articles, however, for now we can just say that they are correlated.

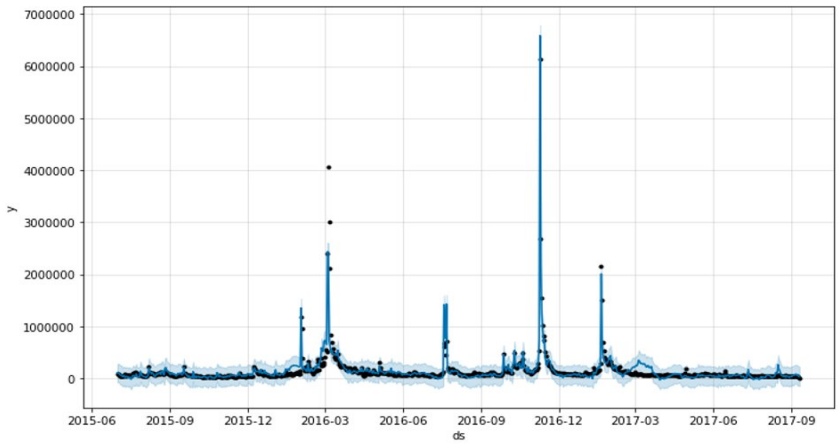
Model Evaluation and Results

After the modification of the data to include these new columns, we used FB Prophet on python to make some predictions about the data. Since there were 2 years of data, we used 1.5 years of it as training data and the last half a year of it as test data.

Below is without the use of the additional variables and you can clearly see that the peaks are not accurate, the confidence intervals are very wide, and the prediction is overall poor.



The graph below is the model with the extra information. As you can see, even though the model cannot predict the exact peaks in the data, it manages to peak at the correct positions. it even manages to find some of the smaller peaks that were not induced with the manual flagging of important news stories. The confidence intervals are also overall tighter because of the related pages information.



In conclusion, prediction of even news influenced articles is possible. It only requires some web scraping and some related article searching.

As future works, we are interested in hyperlink analysis of the pages of interest, to find which links people go to from there and be able to map these interactions more closely. Wikipedia should have information about which links are clicked more and can therefore use those to make edge maps of the different articles, linking them to each other for processing.

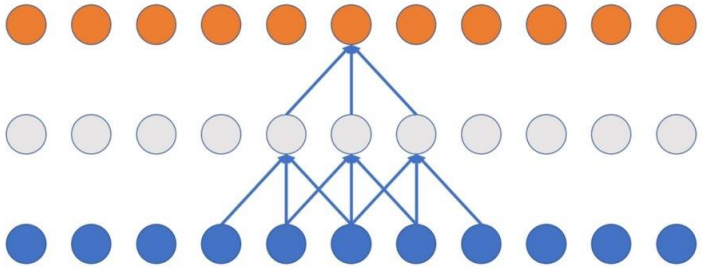
Next, we would want to automate the news categorizing, to be feasible for a large website like Wikipedia. It would be impossible to go through every day and manually categorize the different news. So perhaps a scraping of multiple news sites could be done to understand with what depth, different news are reaching different people, to be able to prepare the website for sudden spikes due to recent news.

Using these two methods, the analysis could then be expanded to include many articles and be completely automated to explore data on its own.

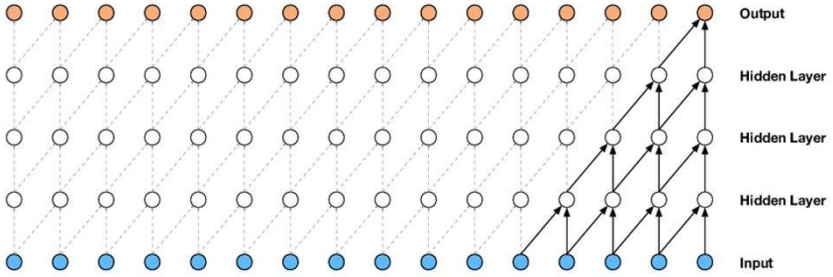
Web Traffic Forecast – Deep Learning

Traditional time series forecasting models such as ARIMA are slow to fit for multiple time series. Using deep learning models to forecast multiple time series maybe another choice. Here we have a total of 803 timepoints (time series), LSTM tends to forget data from distance past >300 time points whereas WaveNet can predict data taking distance past (>1000 datapoints) into consideration so we chose WaveNet for forecasting.

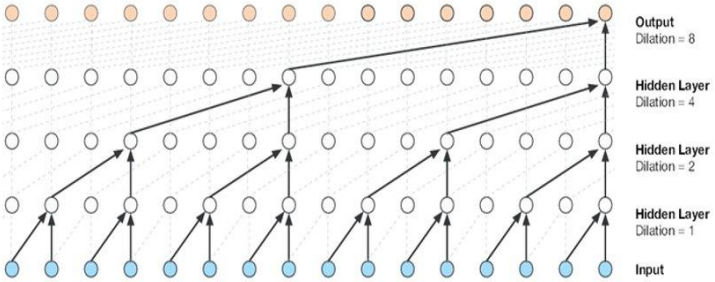
Convolutional layers are used to extract features in deep learning; however, it cannot be used in time series as it will not consider direction and future values are used to predict past. As you can see that nodes after time t are connected to t which causes bias.



By using Casual convolutional layers, we are giving a direction to the model and at any time t only inputs <t can be connected. But the problem with casual convolutional layer is that more layers must be added for distance past to have influence on the output. For instance, in below network at time t only 5 points at max can influence the output and if we need output to be influenced by distance past more layers must be added. 11 more layers must be added if we want input at time t1 to have an influence on the output which is not computationally efficient as it increases the number of parameters to be estimated.



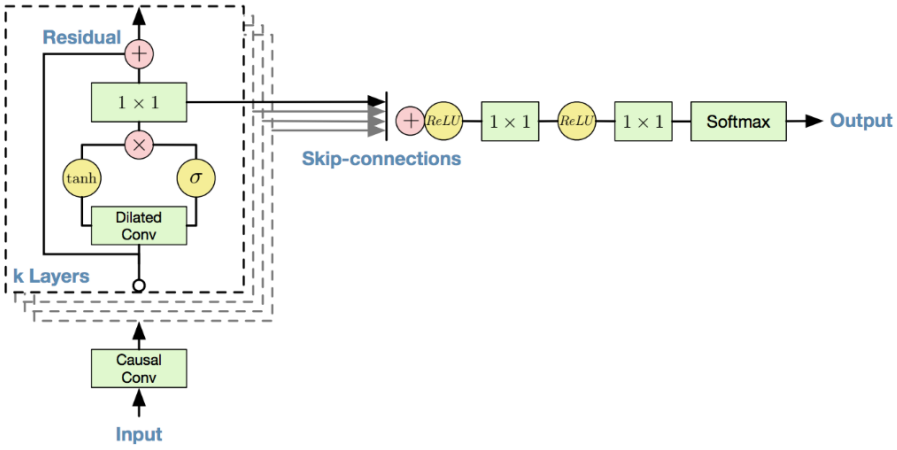
Using dilated convolutional layers solves this problem. Adding Dilation to convolution layer will give the model to learn the influence of distance past with less no. of hidden layers, in below figure you can see that with dilation 1 every input is considered, with dilation 2 every other input is considered, with dilation 4 every 4 inputs is considered and by dilation 8 every 8 inputs is considered. By increasing dilation factor exponentially, we can cover the entire input time series.



Data Preprocessing

For the sake of building the model with limited RAM and GPU, we only considered 1000 English articles for forecasting. As the number of views are dispersed with values ranging from 0-7000 log(views) was used to scale down the model and as we also have inputs with 0 log1p(views) was used as log(0) is undefined. Day of the week was also given to the model explicitly as a one hot encoder variable to remove weekly seasonality.

Architecture

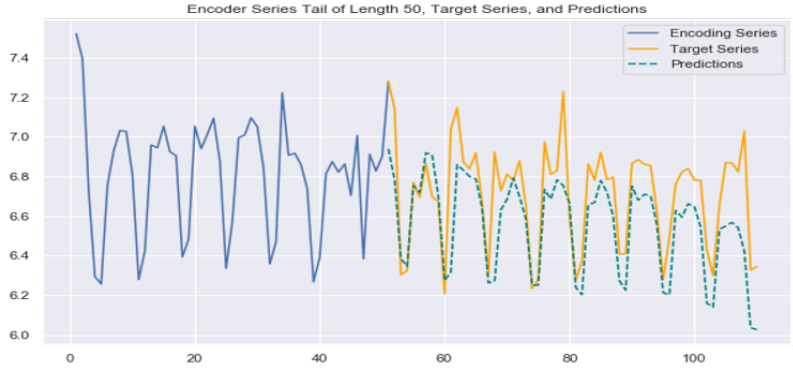


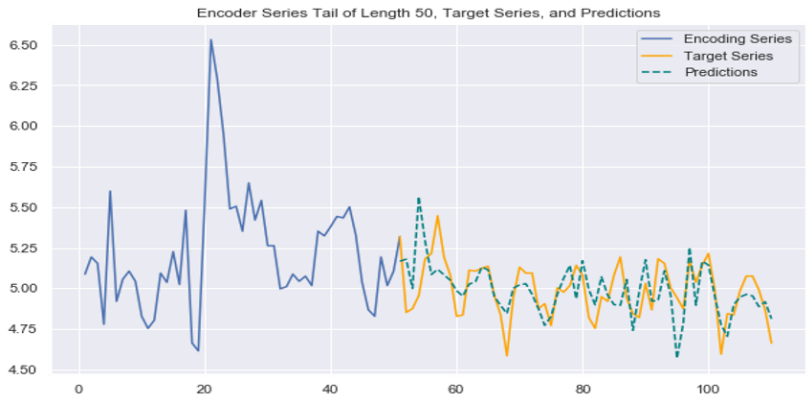
For the architecture of the model we used 16 dilated causal convolutional blocks with 32 filters of width 2 per block and exponentially increasing dilation rate with power of 2 followed by gated activations and residual skip connections. At the end two fully connected layers with “softmax”.

The loss function used was mean absolute error. Adam optimizer was used because it converges quicker batch Size of 128 and epochs of 2000.

Predictions

The following two graphs show our prediction results.





For the model the SMAPE value of 31.5% which was quite promising and in future works we want to fit the model for all the 145k rows.

Conclusions

Overall, our analysis of these 145 thousand rows of Wikipedia articles led us all in different directions. Some of our analysis looked at overall patterns and some looked at specific articles, while others looked at different ways of predicting the future using this data. From our overall success, we conclude that Wikipedia article analysis is certainly possible. If the data is grouped properly into clusters that all are indicative of different things, we can then use event based prophet analyses(for celebrities or brands), or non-event based prophet analyses(for seasonal events like Santa or the Easter Bunny) or even our deep learning models(for articles prophet can't deal with). This really showed us the variety of the data set, as we had to use more than 3 different approaches to really tackle the problem, however it made us realize that no matter how complicated the system, by clustering up the different approaches, we could arrive at a greater understanding than we began with.