Machine Learning(MO444)
Final Series Forecasting

Felipe Galvão
RA:116790
felipelemes@outlook.com

William Tustumi
RA:120281
whatust@gmail.com

Leo Yuuki Omori Omi
RA 138684
leoyuuki@gmail.com

João Pedro Ramos Lopes
RA 139546
email address

***Abstract*—**

I. Introduction

This project will focus on the problem of Web Traffic Time Series Forecasting hosted on Kaggle [2]. The problem focuses on the problem of forecasting the future values of multiple time series. Sequential or temporal observations emerge in many key real-world problems, ranging from biological data, financial markets, weather forecasting, to audio and video processing. The field of time series encapsulates many different problems, ranging from analysis and inference to classification and forecast. On our project we will forecast future web traffic for approximately Wikipedia pages, with a dataset provided by Kaggle [3].

II. Dataset Analysis

A. Dataset

The dataset provided is comprised of 803 days of visits measurements from 145063 combination of wikipedia page and access agent. An page name and header example can be seen on table I. The entries with NaN are correspondent to the days when the wikipedia page did not existed yet, thus those values where substituted by 0 in order to treat this edge case.

	2015 07/01	2015 07/02	2015 07/03	2015 07/04	2015 07/05
2NE1_zh.wikipedia.org_...	18 key	11	5	13	14
2PM_zh.wikipedia.org_a...	11	14	15	18	11
3C_zh.wikipedia.org_al...	1	0	1	1	0
4minute_zh.wikipedia.or...	35	13	10	94	4
52_Hz_I_Love_You_zh...	NaN	NaN	NaN	NaN	NaN
5566_zh.wikipedia.org_...	12	7	4	5	20
91Days_zh.wikipedia.org...	NaN	NaN	NaN	NaN	NaN
A'N'D_zh.wikipedia.org:.	118	26	30	24	29

Table I
Data Example from Kaggle Series Forecast

The first step was performing an analysis of the data to understand which patterns could leverage our learning methods. Plotting graphs of the traffic from a few pages we observed that different pages show remarkably distinguish trends in traffic. This suggests that a good model needs to explicitly differentiate pages. For specific entries we note a strong year-to-year autocorrelation. We also observed weaker week-to-week and month-to-month trends. One clear example of this case was the Thanksgiving holiday page that showed an yearly spike around a Thursday on late November, which is the day of the holiday.

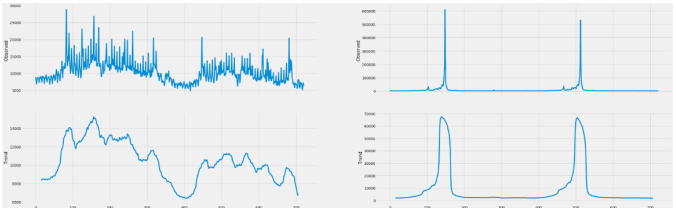

Figure 1. Observed Traffic and Traffic Trend from two different pages

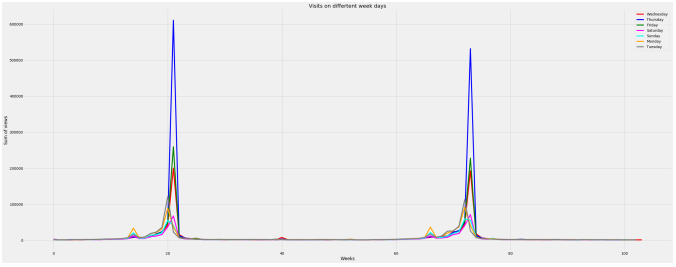

Figure 2. Observed Traffic separated by weekday for the thanksgiving page.

Additionally, we have to deal with spikes in traffic not following a regular pattern. We note a short-term dependency following those spikes, which means the days immediately before the prediction are important to consider.

III. Background

A. Time Series

B. Multi-Layer Fully Connected

C. Recurrent Neural Network

A Recurrent Neural Network (RNN) is very similar to a feed-forward network, except that the connections also links backwards on the network. So the output is defined similarly to equation 1. Note that the result from last step contributes to the next and so on, therefore the ith step depends on all previous steps, which takes a form of a memory [1].

$$Y_{(t)} = \phi(X_{(t)}\dot{W}_x + Y_{(t-1)}\dot{W}_y + b) \quad (1)$$

The training method analog to the feed-forward networks, the first step is a forward pass that unroll the network time

loop, the values for all $Y's$ in the time series are also calculated in the process. The gradient value is calculated using all the results in the cost function, for example if the time series generates $Y_1, \dots Y_n$ and the cost function uses only the last three values of Y, than the gradient is calculated using only Y_{n-2}, Y_{n-1}, Y_n.

Recurrent neural network tends to suffer from exploding and vanishing gradients more than a convolution or fully connected network, since the number of steps in time it unrolls also adds to the depth of the network. To solve this problem techniques like batch normalization, non-saturating activation functions and gradient clipping are exploited.

Another problem with the simple approach shown previously happens when the network has to remember inputs far early in the time steps, most of the information is lost by transversing the network. To overcome this problem a more sophisticated cell was developed, Long Short Term Memory (LSTM). The full explanation of how it works is given on subsection III-D.

D. Long Short Term Memory

The main improvement LSMT cell wants to achieve is learn what to store in the long term memory, what to discard and what to use from the input. The LSTM cell keeps two state matrix that holds information from previous time sequences, one for long an the other for short term memory, those states are than used as input for the next iteration. The figure 3 show a LSTM diagram, the $c_{(t-1)}$ represents the long term memory input and the $h_{(t-1)}$ represent the short term memory, note that the short term output is also the $Y_{(t)}$.

The LSTM cell uses one fully connected layer with hyperbolic tangent activation function as the main flow of input and three other fully connected with logistic activation function as controllers. Each controller is attached to a gate, basically an element-wise multiplication, since the logistic function output range is 0 to 1 the gate is either close when 0 or open when 1.

Each gate receives a name based on their function and is controlled by one of the matrix controllers:

1) Forget Gate: controls what information from the input is retained on the long term state, it receives $f_{(t)}$ (logistic function controllers) and $c_{(t-1)}$ (long term state) as input.

2) Input Gate: controls which information on the main flow will be added to the long term memory and the output memory, receives $g_{(t)}$ (main input flow) and $i_{(t)}$ (logistic controller).

3) Output Gate: controls the information that goes to the output and short term memory, receives the addition result from *Input Gate* and *Forget Gate* after an hyperbolic tangent activation function and $o_{(t-1)}$ (logistic function controller).

IV. Experiment Results & Discussion

We first separated or data in train, validation and test datasets, comprising of respectively 70%, 20% and 10% of the total data. We broke our data in the time axis, so the validation and test data are referents to the later days instead of different pages. After we separated the dataset we end up

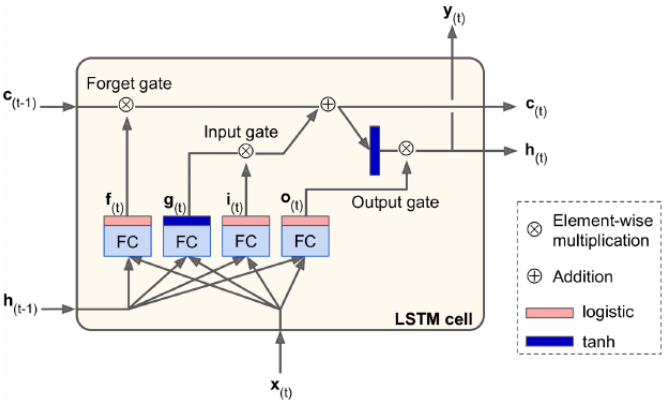

Figure 3. LSTM Diagram from [1]

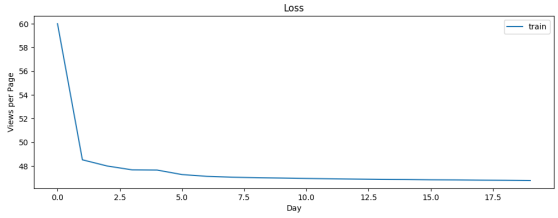

Figure 4. Loss graph from model MLP

with, 559 samples for the training set, 159 samples for the validation set and 79 samples for the test set.

The code of the experiments were written in python with the module for neural network keras using the tensorflow as back-end. The models were ran in a intel core i7-6700k, with 16GB of memory and Geforce-980 Ti.

Due to memory constraints we decided to limit the number of pages on the test data, from 145063 to 50000.

A. MLP

The MLP model receives an input composed of an unique identifier of the wikipage accessed, the number of days passed since the first measurement was collected, and the number of visits from the last 7 days.

The MLP is built with one input layer of size 1024, two hidden layers of size 1024 and a output layer of size 1. The optimizer chosen was Adam with learning rate of 10^{-7} during 20 epochs and using batch size of 1000. The learning rate does not decrease over time. The loss graph can be seen on figure 4.

It is easy to see on figure 5 that the model achieved a pretty good accuracy on the maintaining close to the behavior of the curve, however the model undershoot the largest spikes, which probably leads to the biggest percentage of lost score.

The graph for thanksgiving is a very odd one, the only spikes happens on thanksgiving day, which is once a year and does not occur neither on validation data or test data, leaving the hardest prediction to appear only on training data.

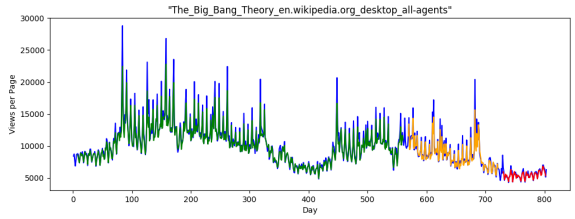

Figure 5. Plot of the values of visits per day compared to the prediction of MLP of the page The Big Bang Theory (en) of all desktops

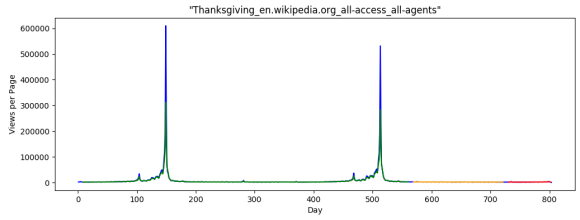

Figure 6. Plot of the values of visits per day compared to the prediction of MLP of the page Thanksgiving (en) of all desktops

B. LSTM-stateless

The LSTM model only uses as input, the page id, the number of days pass the first measurement and the last count of visits of that page.

The LSTM model uses 4 layers of LSTM cells of size 256, a normalization batch, two fully connected layers of size 1024 with ReLU as activation function and an output layer. parallel to the LSTM layer there is a fully connected layer of size 1024 to try improving the use of recent data, a batch normalization is also used after the fully connected layer too.

The batch normalization layers are used to contain the exploding gradients of deeper neural networks, as well as the ReLU activation function. The Adam optimizer was chosen using learning rate of 10^-5, whiteout learning rate decay, the model was trained during 5 epochs with batch size of 1000. The loss curve from the model training is displayed on figure 7.

the figure 8 shows that the LSTM model also follows the general format of the visit graph like the MLP model, however it undershoots not only the high spikes but, also some lower and more plateau segments. which increases the loss value of the model.

On the figure 9 it is possible to see that the LSTM model has a lot of difficult to follow the hight spikes from some pages.

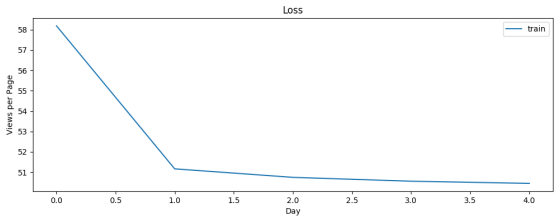

Figure 7. Loss graph from model LSTM

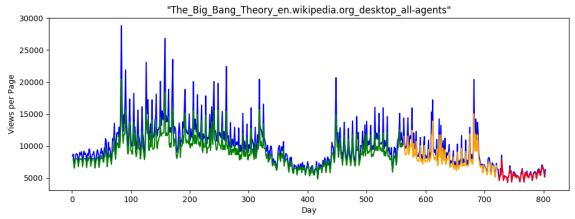

Figure 8. Plot of the values of visits per day, of the page The Big Bang Theory (en) of all desktops

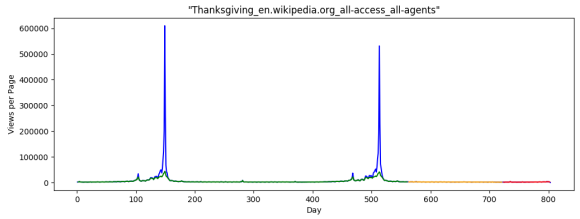

Figure 9. Plot of the values of visits per day, of the page Thanksgiving (en) of all desktops

C. LSTM-stateful

V. Conclusion & Future Work

Our best model was the MLP which used values from 7 days previous the one intended to be predicted, however this model has a obvious limitation, because it cannot learn predictions and relation between visits longer than the 7 days it receives as input. In other words no long time sequence is learned using this model only small (7-days), page and day visit relation, since both data are also inputs. The stateless LSTM functions closely to an LSTM and end-up achieving a very similar performance to the MLP model, specially with the fully connected bypassing the LSTM layers.

The stateful LSTM takes a lot longer to train and since we could only test one architecture it is unclear if a model based on this component can achieve significant better results. This was a good base work to future experiments on the wikipedia page visit forecast problem.

Possible ways to improve our models are:

- Use embedding to cluster pages with similar behaviors (all models);
- Use visit measurements from days before 7 days prior, by calculating mean or media of a window of days (MLP model);
- Use a different ways to feed inputs to the model, by increasing the time-steps in each backpropagarion rollback (LSTM stateful model);
- increase the number of LSTM cells (LSTM stateful model);

REFERENCES

[1] Aurélien Géron. Hands-on machine learning with scikit-learn and tensorflow: concepts, tools, and techniques to build intelligent systems, 2017.

[2] Kaggle. Web traffic time series forecasting. https://www.kaggle.com/c/web-traffic-time-series-forecasting.

[3] Kaggle. Web traffic time series forecasting data.