Web Traffic
Time Series Forecasting

Kaggle Competition Review

Bill Tubbs

July 26, 2018

Competition Summary

- Goal: Forecast future web traffic for ~145,000 Wikipedia articles
- When: 8 months ago
- Sponsored by Google and Voleon
- 375 teams
- Prizes {1: $12,000, 2: $8,000, 3: $5,000}

+ Present at NIPS Time Series Workshop in California

URL: https://www.kaggle.com/c/web-traffic-time-series-forecasting

Presentation Overview

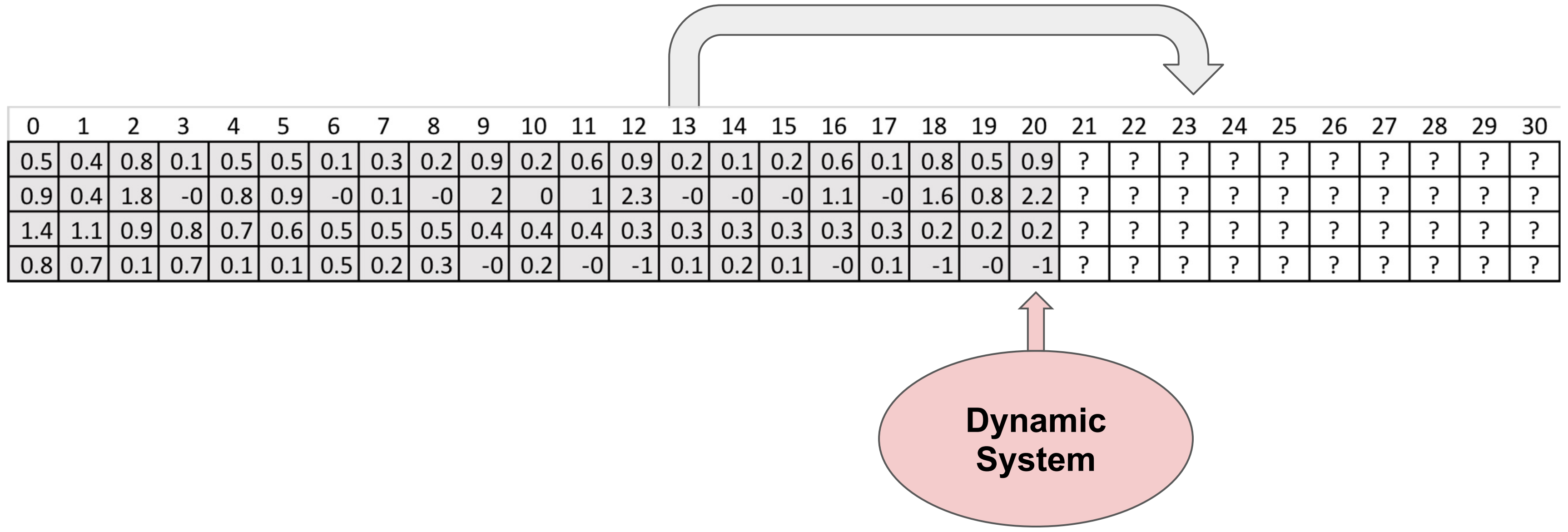
- The problem
- The data
- The evaluation metric
- The leaderboard
- Overview of winner’s solution
- What I learned

The Problem

- Forecasting future web traffic for approximately 145,000 Wikipedia articles.
  - Why?
- Forecasting future values of multiple time series is *“one of the most challenging problems in the field”*

URL: https://www.kaggle.com/c/web-traffic-time-series-forecasting

Time Series Analysis And Forecasting



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
0.5	0.4	0.8	0.1	0.5	0.5	0.1	0.3	0.2	0.9	0.2	0.6	0.9	0.2	0.1	0.2	0.6	0.1	0.8	0.5	0.9	?	?	?	?	?	?	?	?	?	?
0.9	0.4	1.8	-0	0.8	0.9	-0	0.1	-0	2	0	1	2.3	-0	-0	-0	1.1	-0	1.6	0.8	2.2	?	?	?	?	?	?	?	?	?	?
1.4	1.1	0.9	0.8	0.7	0.6	0.5	0.5	0.5	0.4	0.4	0.4	0.3	0.3	0.3	0.3	0.3	0.3	0.2	0.2	0.2	?	?	?	?	?	?	?	?	?	?
0.8	0.7	0.1	0.7	0.1	0.1	0.5	0.2	0.3	-0	0.2	-0	-1	0.1	0.2	0.1	-0	0.1	-1	-0	-1	?	?	?	?	?	?	?	?	?	?

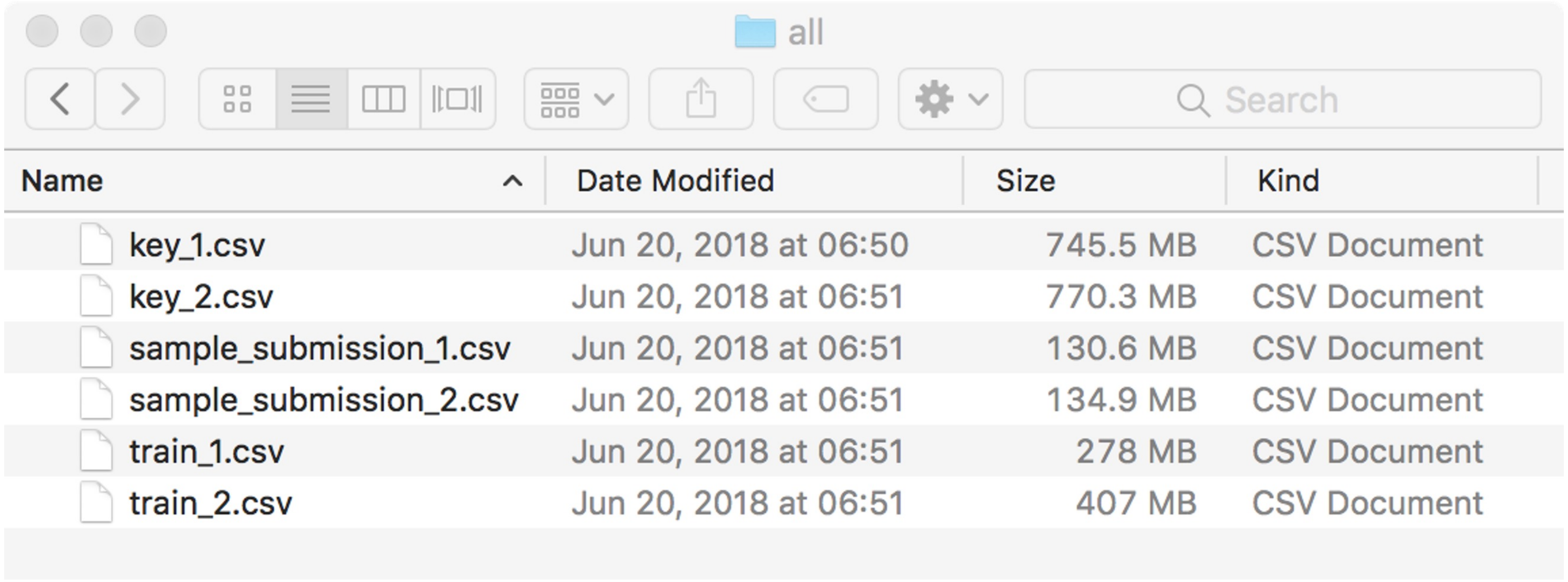
URL: https://en.wikipedia.org/wiki/Time_series

Competition Timeline

- Training phase:
    - Forecast traffic in January and February 2017 based on historical data from July 2015 to December 2016
- Future phase:
    - Forecast future traffic between September 13th and November 13th, 2017 based on data up to September 1st, 2017
- September 1, 2017 - Final dataset released
- September 12, 2017 - Final submission deadline
- November 13, 2017 - Competition winners revealed

URL: https://www.kaggle.com/c/web-traffic-time-series-forecasting

The Data



Name	Date Modified	Size	Kind
key_1.csv	Jun 20, 2018 at 06:50	745.5 MB	CSV Document
key_2.csv	Jun 20, 2018 at 06:51	770.3 MB	CSV Document
sample_submission_1.csv	Jun 20, 2018 at 06:51	130.6 MB	CSV Document
sample_submission_2.csv	Jun 20, 2018 at 06:51	134.9 MB	CSV Document
train_1.csv	Jun 20, 2018 at 06:51	278 MB	CSV Document
train_2.csv	Jun 20, 2018 at 06:51	407 MB	CSV Document

URL: https://www.kaggle.com/c/web-traffic-time-series-forecasting/data

The Data

- 145,000 time series:
  - Daily page views, 2015-07-01 to 2016-12-31

traffic data

page name

train_1.csv

```
"Page","2015-07-01","2015-07-02","2015-07-03",… "2016-12-31"
"2NE1_zh.wikipedia.org_all-access_spider",18,11,5,… 20
"2PM_zh.wikipedia.org_all-access_spider",11,14,15,… 20
                                   …

"Bogotá_es.wikipedia.org_all-access_all-agents",2685,2849,3045,… 1967
                                   …

"陳法拉_zh.wikipedia.org_mobile-web_all-agents",293,474,252,… 192
                                   …
```

The Data

- List of keys

key_1.csv

shortened id

```
"Page","Id"
"!vote_en.wikipedia.org_all-access_all-agents_2017-01-01",bf4edcf969af
"!vote_en.wikipedia.org_all-access_all-agents_2017-01-02",929ed2bf52b9
...
"Bogotá_es.wikipedia.org_all-access_all-agents_2017-01-23",25e7cc352d8e
...
"陳法拉_zh.wikipedia.org_mobile-web_all-agents_2017-02-03",50fa6fe170be
...
```

The Data

- Sample submission

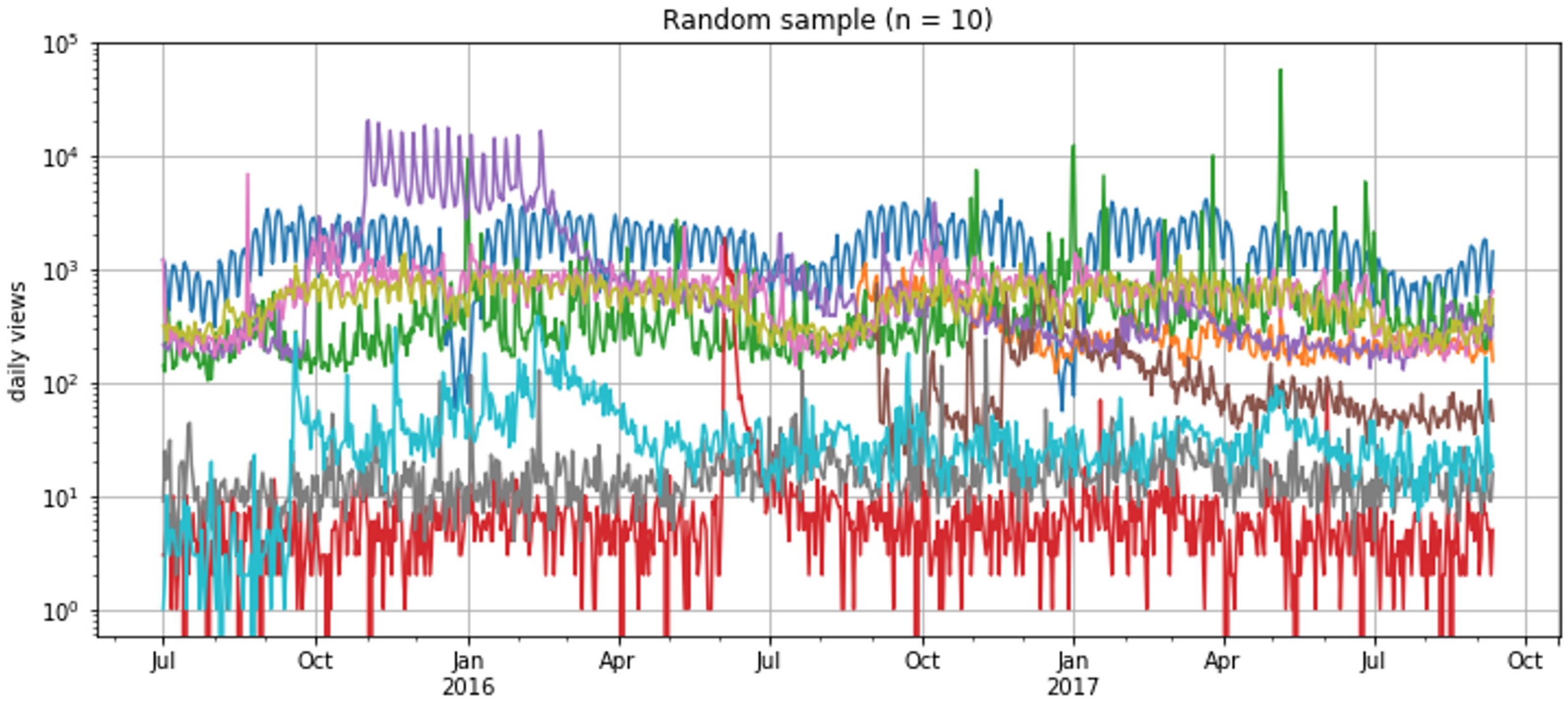
`sample_submission_1.csv`

```
Id,Visits
bf4edcf969af,0
929ed2bf52b9,0
ff29d0f51d5c,0
e98873359be6,0
fa01243426a,0
```

Your predictions go here!

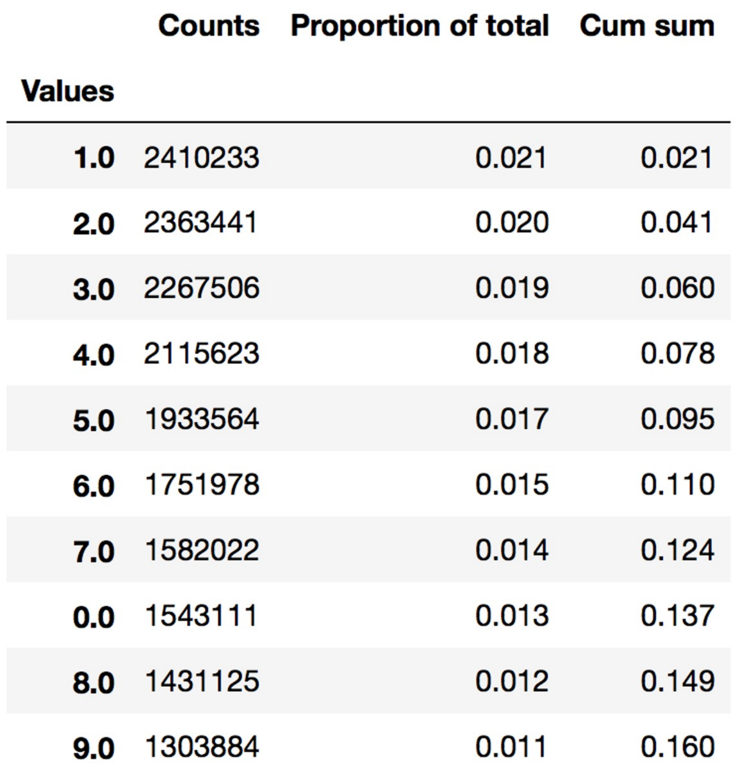
...

The Data

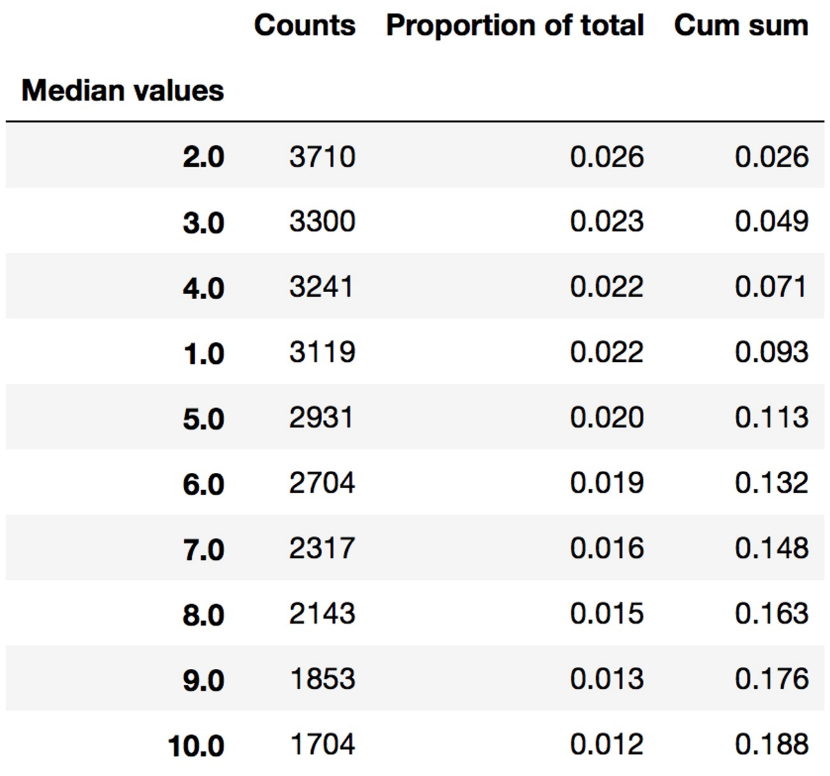


Random sample (n = 10)
daily views
10^5
10^4
10^3
10^2
10^1
10^0
Jul
Oct
Jan 2016
Apr
Jul
Oct
Jan 2017
Apr
Jul
Oct

The Data



Values	Counts	Proportion of total	Cum sum
1.0	2410233	0.021	0.021
2.0	2363441	0.020	0.041
3.0	2267506	0.019	0.060
4.0	2115623	0.018	0.078
5.0	1933564	0.017	0.095
6.0	1751978	0.015	0.110
7.0	1582022	0.014	0.124
0.0	1543111	0.013	0.137
8.0	1431125	0.012	0.149
9.0	1303884	0.011	0.160



Median values	Counts	Proportion of total	Cum sum
2.0	3710	0.026	0.026
3.0	3300	0.023	0.049
4.0	3241	0.022	0.071
1.0	3119	0.022	0.093
5.0	2931	0.020	0.113
6.0	2704	0.019	0.132
7.0	2317	0.016	0.148
8.0	2143	0.015	0.163
9.0	1853	0.013	0.176
10.0	1704	0.012	0.188

The Data

Site	Count
wikipedia.org	127208
commons.wikimedia.org	10555
www.mediawiki.org	7300

Country	Count
en	24108
ja	20431
de	18547
fr	17802
zh	17229
ru	15022
es	14069

Agent	Count
all-access_all-agents	39402
mobile-web_all-agents	35939
all-access_spider	34913
desktop_all-agents	34809

Marker	Count
File	5566
Category	3565
Special	1697
Help	1252
Commons	694
Manual	686
Extension	651
User	279
Project	229
API	192
Topic	188

Evaluation

- Metric:
  - SMAPE - Symmetric mean absolute percentage error
  - Based on percentage (or relative) errors

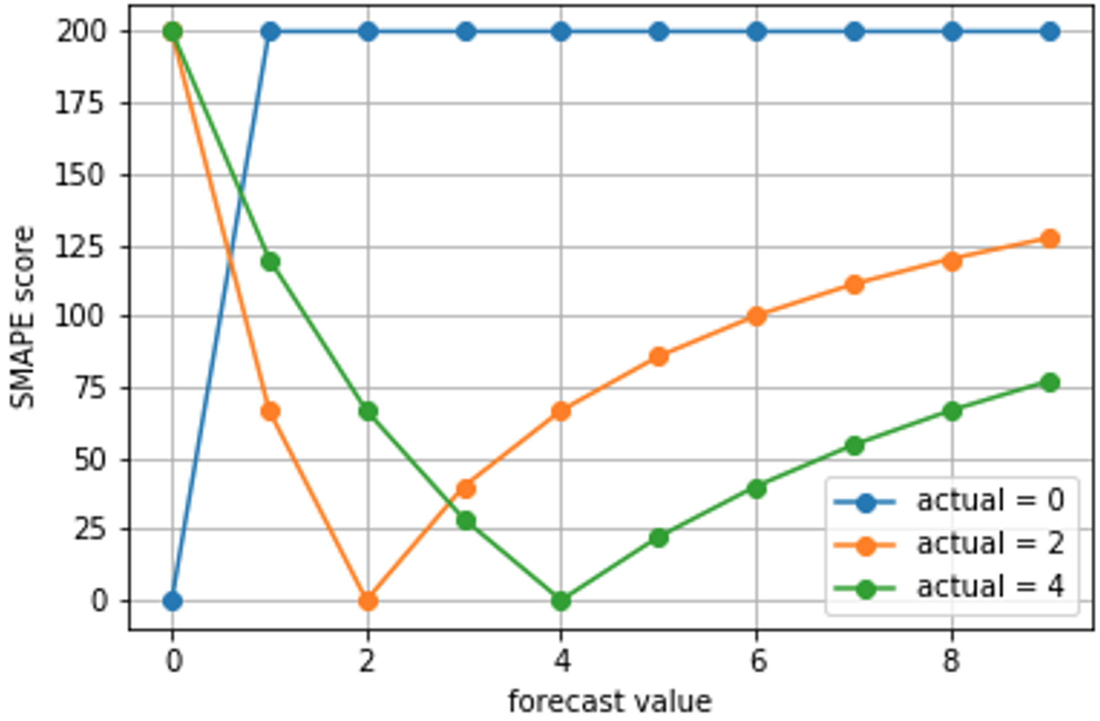
$$\text{SMAPE} = \frac{100\%}{n} \sum_{t=1}^{n} \frac{|F_t - A_t|}{(|A_t| + |F_t|)/2}$$

where A_t is actual value, F_t is forecast value

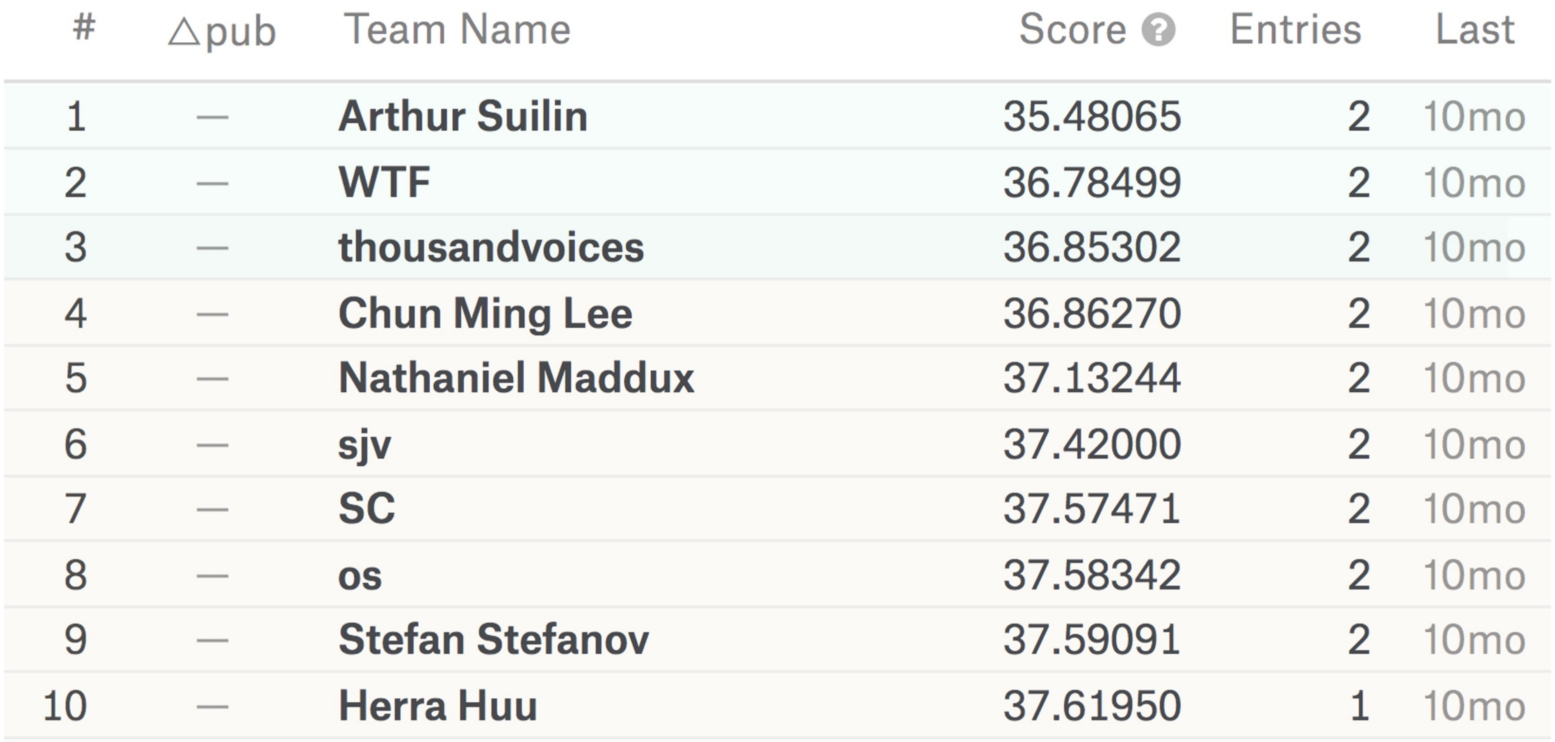
URL: https://en.wikipedia.org/wiki/Symmetric_mean_absolute_percentage_error

Evaluation

- SMAPE



Final Results



#	△pub	Team Name	Score	Entries	Last
1	—	Arthur Suilin	35.48065	2	10mo
2	—	WTF	36.78499	2	10mo
3	—	thousandvoices	36.85302	2	10mo
4	—	Chun Ming Lee	36.86270	2	10mo
5	—	Nathaniel Maddux	37.13244	2	10mo
6	—	sjv	37.42000	2	10mo
7	—	SC	37.57471	2	10mo
8	—	os	37.58342	2	10mo
9	—	Stefan Stefanov	37.59091	2	10mo
10	—	Herra Huu	37.61950	1	10mo

Winner

Suilin Arthur • 3rd

Education

Российский Химико-Технологический Университет им. Д.И. Менделеева / Mendeleyev University of Chemical Technology of Russia
Bachelor of Science (BS), Biotechnology
1991 – 1996

Deep Social

Chief Data Scientist
Deep.Social
Oct 2017 – Present • 10 mos

Yandex

Head of Yandex.Metrica
Yandex
Nov 2012 – Sep 2015 • 2 yrs 11 mos

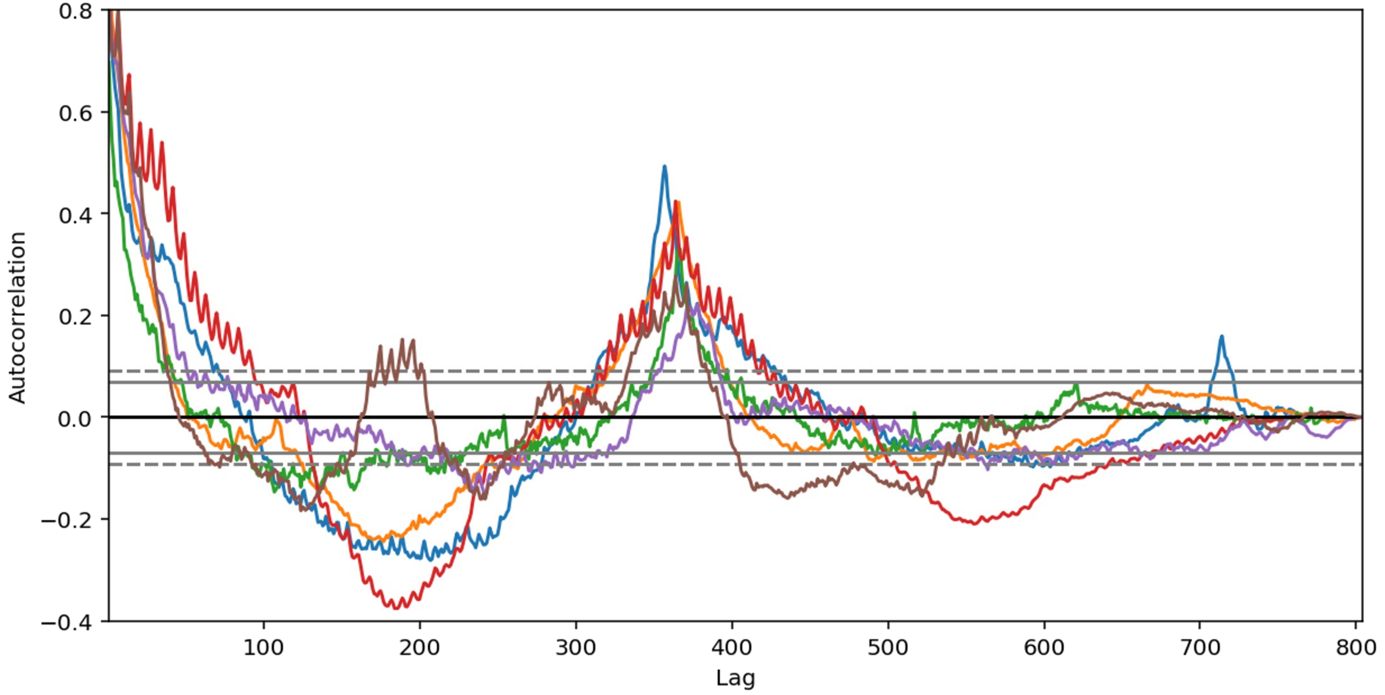
Intuition

There are two main information sources for prediction:

- Local features.
  - If we see a trend, we expect that it will continue (auto-regressive model)
  - If we see a traffic spike, it will gradually decay (moving-average model)
  - If we see more traffic on holidays, we expect to have more traffic on holidays in the future (seasonal model).
- Global features
  - If we look to autocorrelation plot, we'll notice strong year-to-year autocorrelation and some quarter-to-quarter autocorrelation.

Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

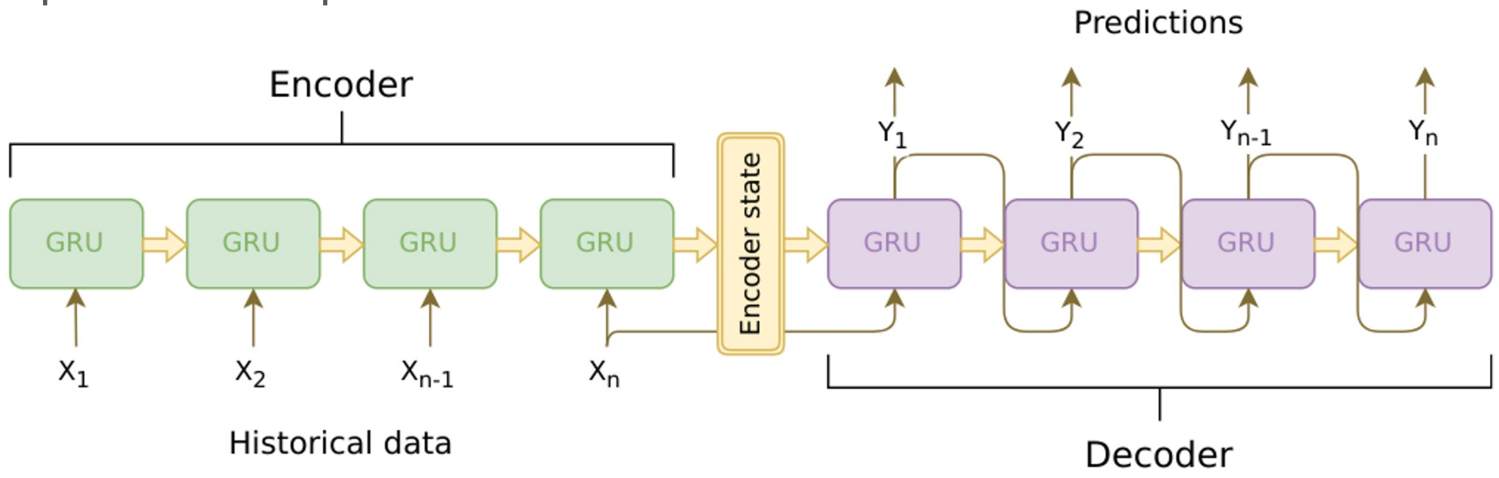
Autocorrelation Plot



Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Model

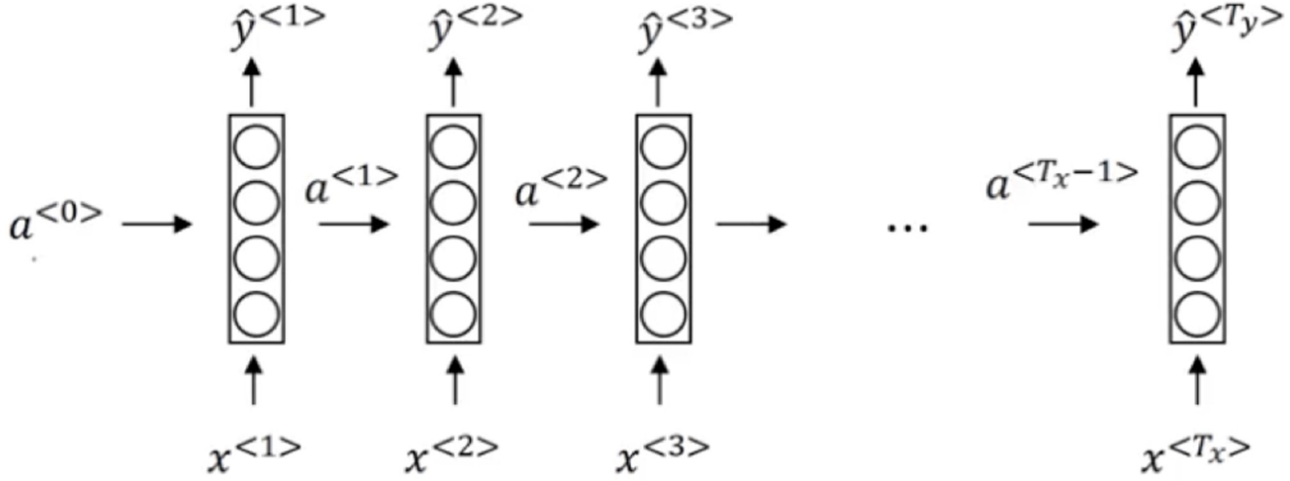
Sequence-to-sequence model



- Encoder is cuDNN GRU.
- Decoder is TF GRUBlockCell, wrapped in `tf.while_loop()` construct.

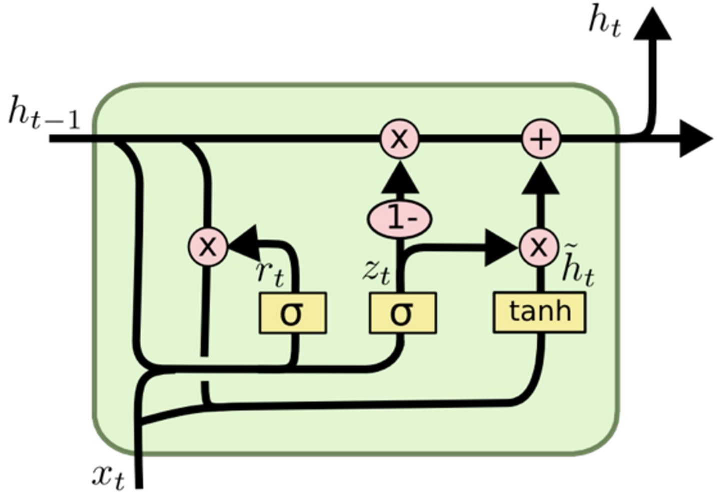
Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Recurrent Neural Networks (RNNs)



Source: https://www.coursera.org/specializations/deep-learning

Gated Recurrent Unit (GRU)



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Model Design Choice

I decided to use a *seq2seq* model (RNN) for prediction, because:

- RNN can be thought of as a natural extension of well-studied ARIMA models, but much more flexible and expressive
- RNN is non-parametric, that greatly simplifies learning
- Accepts any exogenous feature (numerical or categorical, time-dependent or series-dependent) can be easily injected into the model
- seq2seq seems natural for this task: we predict next values, conditioning on joint probability of previous values, including our past predictions
- Deep Learning is all the hype nowadays.

Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Feature Engineering

Minimalistic because RNN is able to discover and learn features on its own:

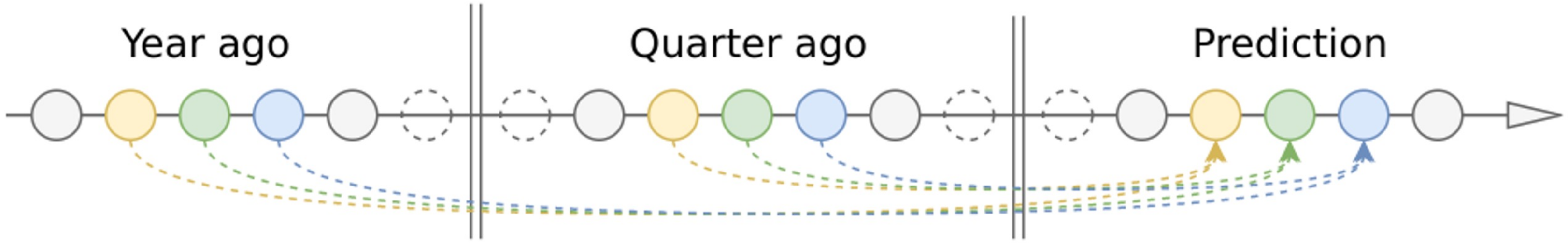
1. **pageviews** - raw values transformed by `log1p()` to get more-or-less normal intra-series values distribution, instead of skewed one.
2. **agent, country, site** - extracted from page urls and one-hot encoded
3. **day of week** - to capture weekly seasonality
4. **year-to-year, quarter-to-quarter autocorrelation** - to capture yearly and quarterly seasonality strength
5. **page popularity** (median of pageviews) - helps to capture traffic scale. High traffic and low traffic pages have different traffic change patterns.
6. **lagged pageviews** - I'll describe this feature later

Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Model

Working with long timeseries

- On sequences longer than 100-300 items, even LSTM/GRU can gradually forget the oldest items
- First method was to use some kind of attention
- E.g. Fixed-weight sliding-window attention:

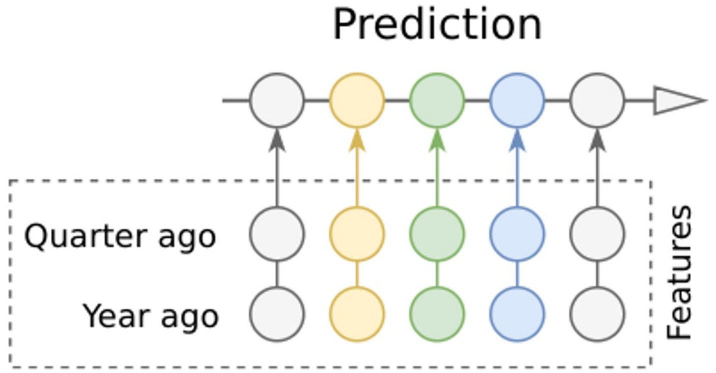


Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Model

Working with long timeseries

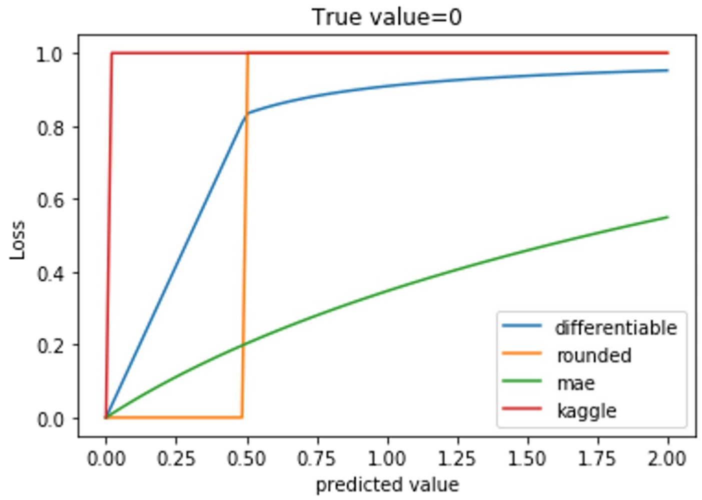
Unsatisfied by complexity of attention mechanics, I removed attention completely and just took important (year, half-year, quarter ago) data points from the past and used them as additional features for encoder and decoder.



Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Training

- SMAPE can't be used directly, because of unstable behavior near zero values (loss is a step function if truth value is zero)
- Used a smoothed differentiable SMAPE variant, which is well-behaved at all real numbers:

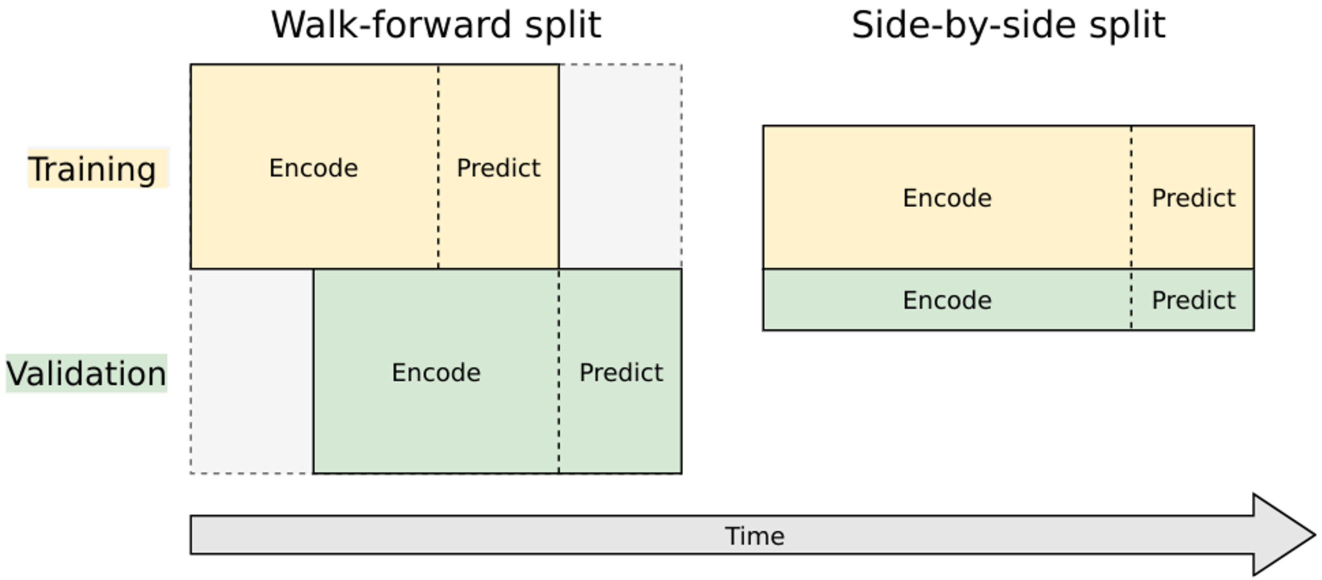


```
epsilon = 0.1
summ = tf.maximum(tf.abs(true) + tf.abs(predicted) + epsilon, 0.5 + epsilon)
smape = tf.abs(predicted - true) / summ * 2.0
```

Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Training

There are two ways to split time series into training and validation datasets:



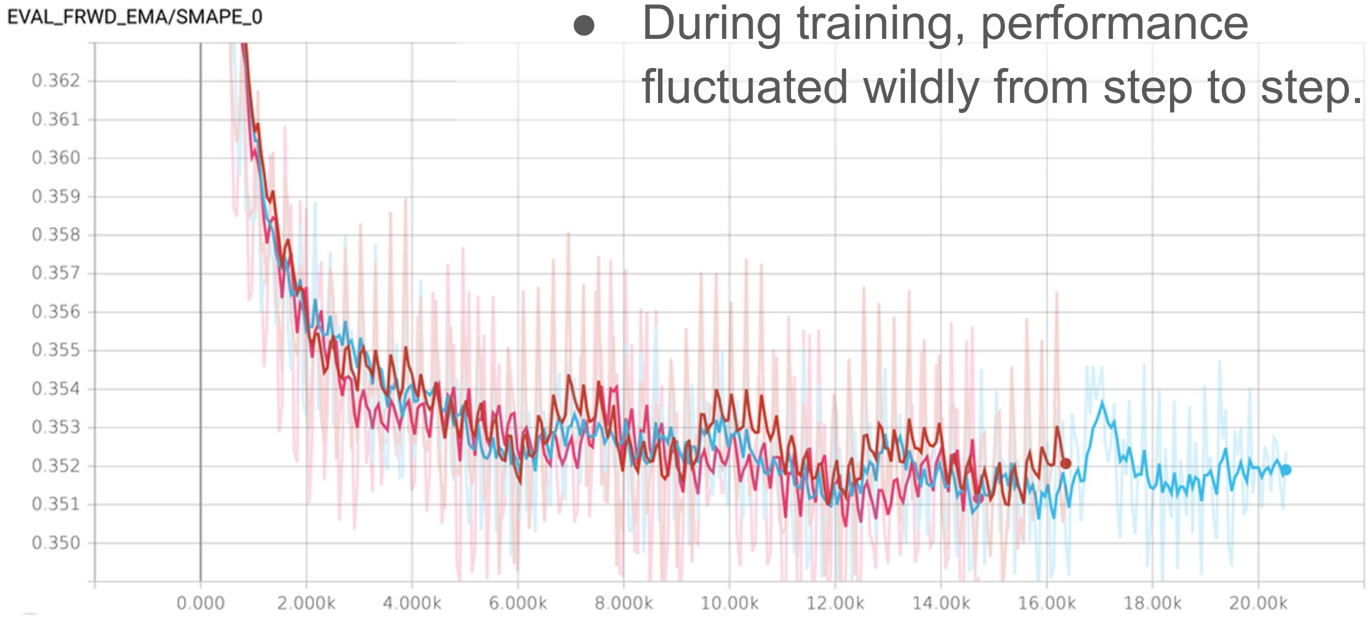
Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Training

- Model trains on random fixed-length samples from original time series.
- Training code randomly chooses starting point for each time series on each step, generating endless stream of almost non-repeating data.
- This sampling is effectively a data augmentation mechanism
- Used COCOB optimizer for training, in combination with gradient clipping
- COCOB tries to predict optimal learning rate for every training step, so you don't have to tune learning rate at all*
- Converges considerably faster than traditional momentum-based optimizers, especially on first epochs, allowing unsuccessful experiments to be stopped.

* See paper Training Deep Networks without Learning Rates Through Coin Betting.
Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Training

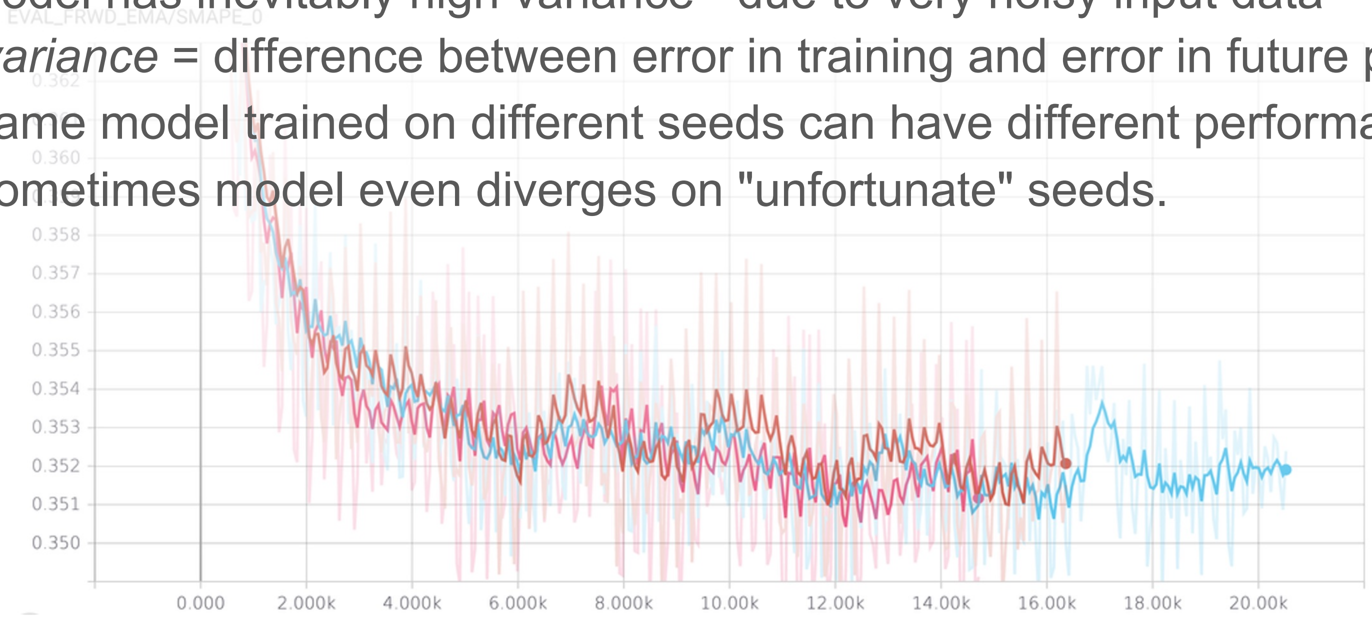


- During training, performance fluctuated wildly from step to step.

Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Issues

- Model has inevitably high variance - due to very noisy input data (*variance* = difference between error in training and error in future prediction)
- Same model trained on different seeds can have different performance
- Sometimes model even diverges on "unfortunate" seeds.



Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Reducing Model Variance

1. Chose a region when model is trained well enough, but has not started to overfit: 10,500 and 11,500 training steps then saved checkpoints every 100 steps in this region
2. Trained 3 models on different seeds and saved checkpoints from each model. Took average predictions from all 30 (10x3) models
3. Used SGD averaging (ASGD) - maintain moving averages of network weights during training and use these instead of original ones, during inference

- Combination of the three methods worked well
- Got roughly the same SMAPE error on leaderboard (for future data) as for validation on historical data.

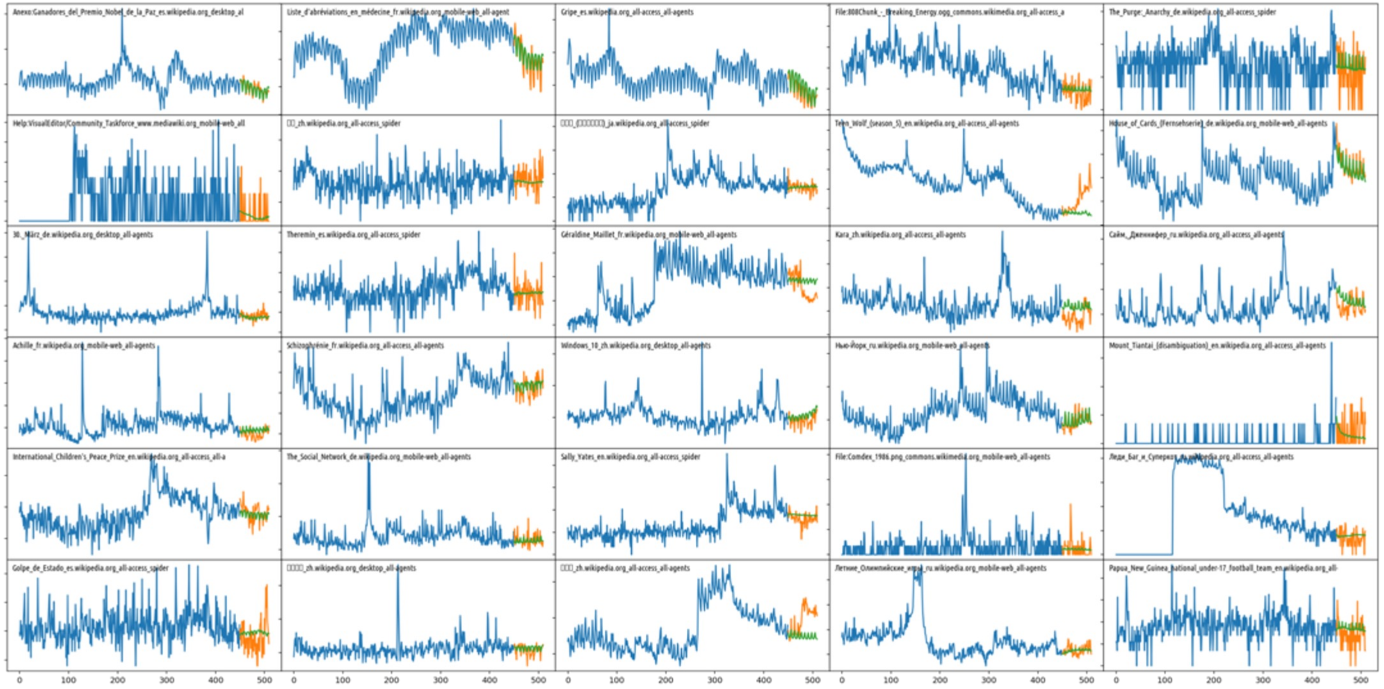
Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Hyper-parameter Tuning

- There are many model parameters (number of layers, layer depths, activation functions, dropout coefficients, etc) that can be tuned to achieve optimal performance.
- Used the SMAC3 package to automate hyperparameter search.
- Contrary to my expectations, hyper-parameter search did not find well-defined global minima
- All best models had roughly the same performance, but different parameters.
- Probably RNN model is too expressive for this task, and best model score depends more on the signal-to-noise ratio than on the model architecture.

Source: https://github.com/Arturus/kaggle-web-traffic/blob/master/how_it_works.md

Predictions



URL: https://github.com/Arturus/kaggle-web-traffic

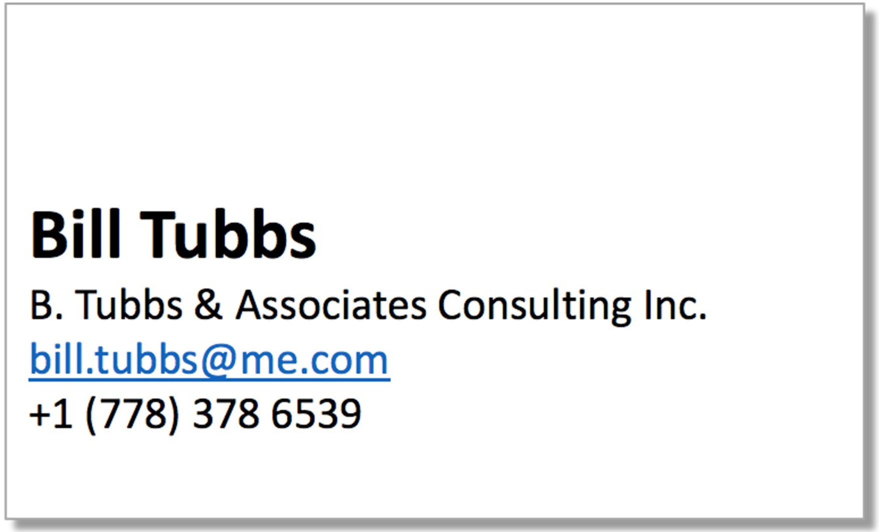
What I Learned

- Explore the data
- Look at performance of traditional methods
- Use intuition to find features that could be useful
- Choose the right cost function - matched to evaluation metric + optimizer
- The rest is luck or magic!
- ...

  Also:

  Trying to run one person’s code on another person’s machine is a nightmare!

Thank You



Bill Tubbs

B. Tubbs & Associates Consulting Inc.

bill.tubbs@me.com

+1 (778) 378 6539

2nd Place

Jean-Francois Puget, PhD • 3rd

“Got a PhD in machine learning in a previous millennium, ML was very different from now, and therefore this PhD is useless…”

“this is the first time I'm using deep learning”

Distinguished Engineer, Machine Learning, Optimization, Advanced Analytics

IBM

Jul 2016 – Present • 2 yrs 1 mo

Nice Area, France

2nd Place Submission

Based on 5 ideas:

1. Use the yearly seasonality of the data - it is huge
2. Don't use RMSE. Approximate SMAPE with log1p transformed data and use custom objective functions for each optimizer
3. Get rid of outliers
4. Ensemble everything in xgboost by training it on the residuals of the Keras predictions and the same features as my XGBoost model plus out of fold predictions from Huber regressor and Keras model
5. Use medians as features instead of raw values.

Source: https://www.kaggle.com/c/web-traffic-time-series-forecasting/discussion/39395

2nd Place Submission

- Feedforward network: [200, 200, 100, 200] units in each layer
- Input is concatenated again with the output of the first layer (I don't know why but this boosted accuracy)
- Activation is relu except for last one which is linear
- Used dropout of 0.5 for almost all layers (0.5 was selected by CV)
- Used a batch normalization for the middle layer
- Model is compiled with adam optimizer and the loss function defined above.
- I tried CNNs and RNNs (LSTM, and Seq2Seq) but did not get results as good as the simple feed-forward network.

Source:
https://www.ibm.com/developerworks/community/blogs/jfp/entry/2nd_Prize_WInning_Solution_to_Web_Traffic_Forecasting_competition_on_Kaggle?lang=en
https://github.com/jfpuget/Kaggle/tree/master/WebTrafficPrediction