NEURO-FUSSY COMPUTING
Programming work

ZIOGAS DIMITRIOS 2214
PETSIS KONSTANTINOS 2168

1) General information

The file extract_diff_length_of_timelines.py takes 2 arguments before it starts, which indicate the limits for which the article time series will be sorted by timeline length. That is, if we run it like this: python extract_diff_length_of_timelines.py 10 16 python extract_diff_length_of_timelines.py 10 16 then the program will read outputacm.txt and create 5 files. timeline10_years_refs.txt, timeline11_years_refs.txt … timeline15_years_refs.txt. Each such file has a series of citation numbers on one line and a series of years in which an article got them the references. timeline10_years_refs.txt contains such lists of 10 years, timeline11_years_refs.txt contains 11 and so on

The final1.py file reads timeseries0x.txt with x=[1,2,3...20], loads the model for the corresponding length timeseries, and makes a forecast for the next year and five years ahead.

The file make_keras_models.py and make_keras_models_5.py train models of 10,11,12...44 time series length for forecasting the next year and five years ahead respectively.

2) Network architecture

We use from keras an LSTM network, which is ideal for price predictions at regular time intervals. LSTM is a Recurrent neural network.

As an optimizer, we use adam(the algorithm at the end).optimizer), we use adam(optimizer), we use adam(the algorithm at the end).the algorithm at the end).

Suppose we are training a network for a timeline of length
10 let TRAIN_SIZE = 10 – 1 = 9.
Network 10 will have as its input a neuron, fed input length TRAIN_SIZE.

In the hidden layer we have TRAIN_SIZE hidden states and they output to the external layer of length TRAIN_SIZE.
As an output, a value is output, the forecast for 1 year or for 5 years.

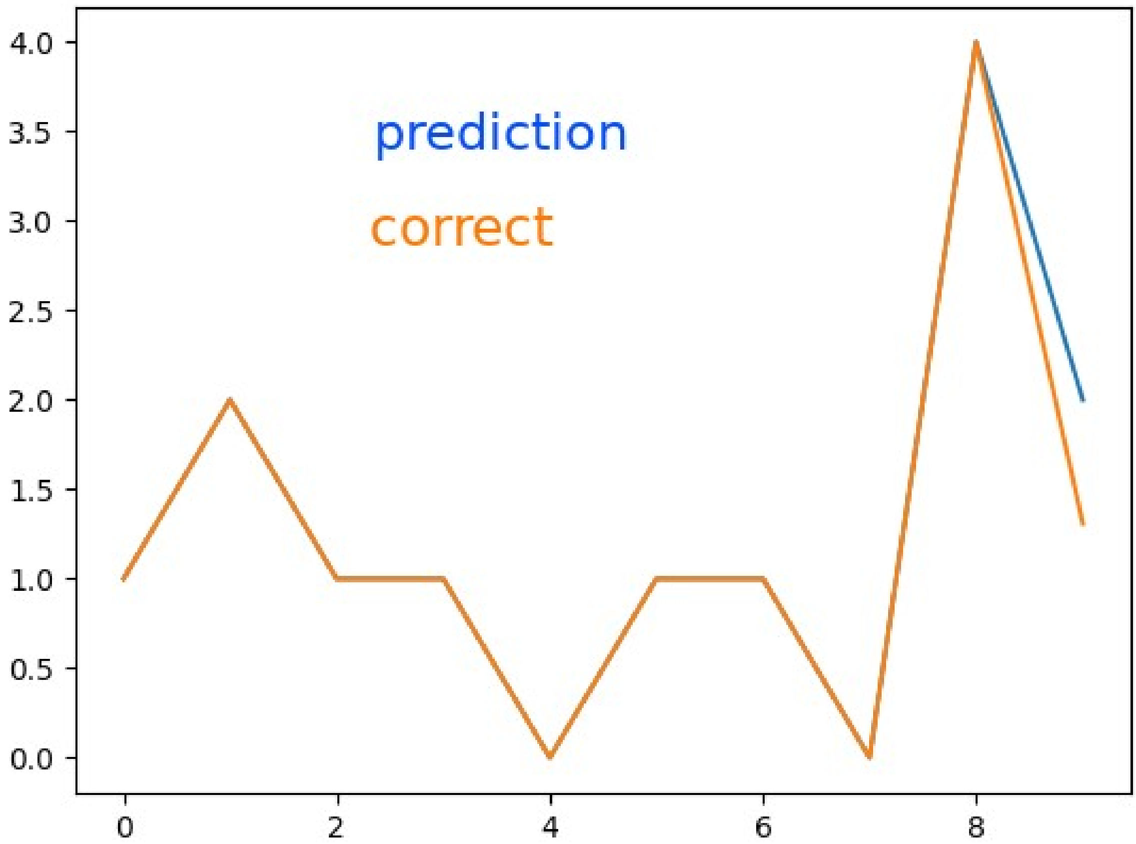
3) Evaluation of performance

For a time series of 10

loss:python extract_diff_length_of_timelines.py 10 16

3.3609317515163966 accuracy:python extract_diff_length_of_timelines.py 10 160.48408710217755446
Time for the system to answer is 0.6987732230008987 seconds.

Prediction for 1 year
[1 2 1 1 0 1 1 0 4 2]
[1, 2, 1, 1, 0, 1, 1, 0, 4, 1.308309]
Prediction = 1.308309
Correct = 2

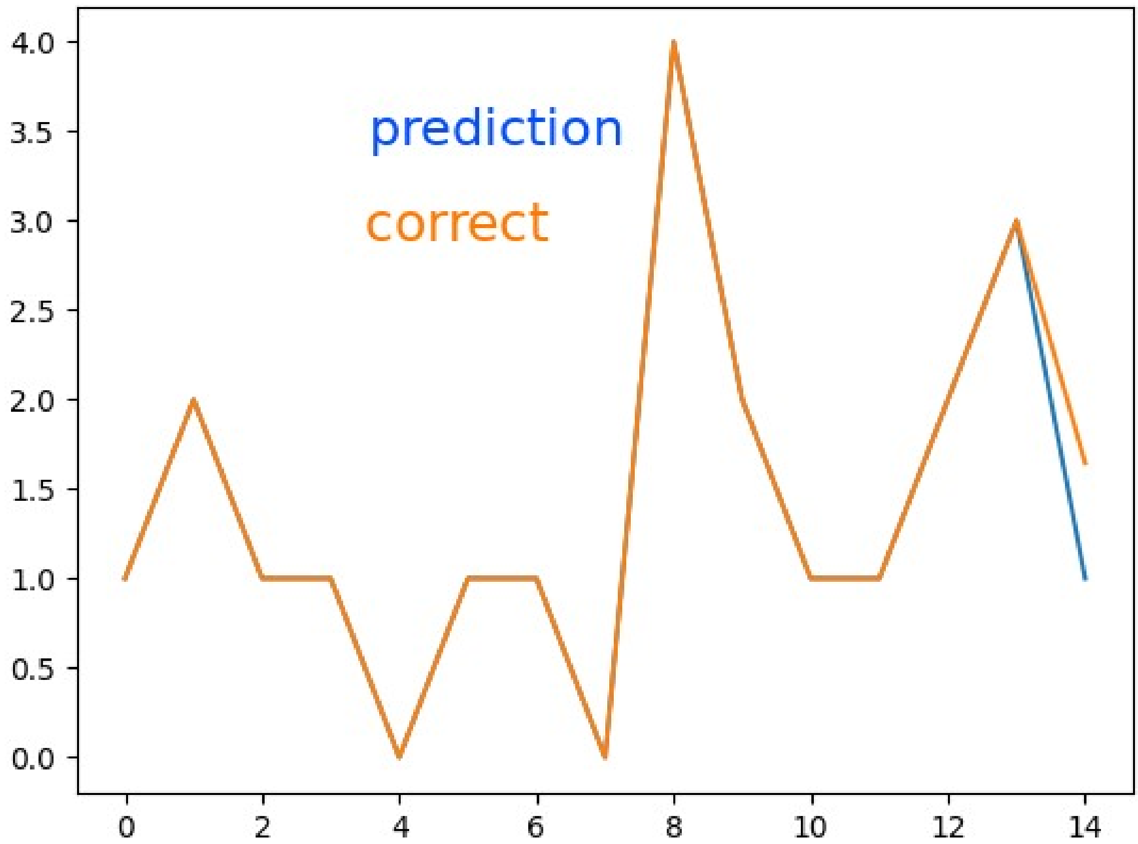


prediction for 5 years
[1 ,2 ,1 ,1 ,0 ,1 ,1 ,0 ,4,2,1,1,2,3,1]

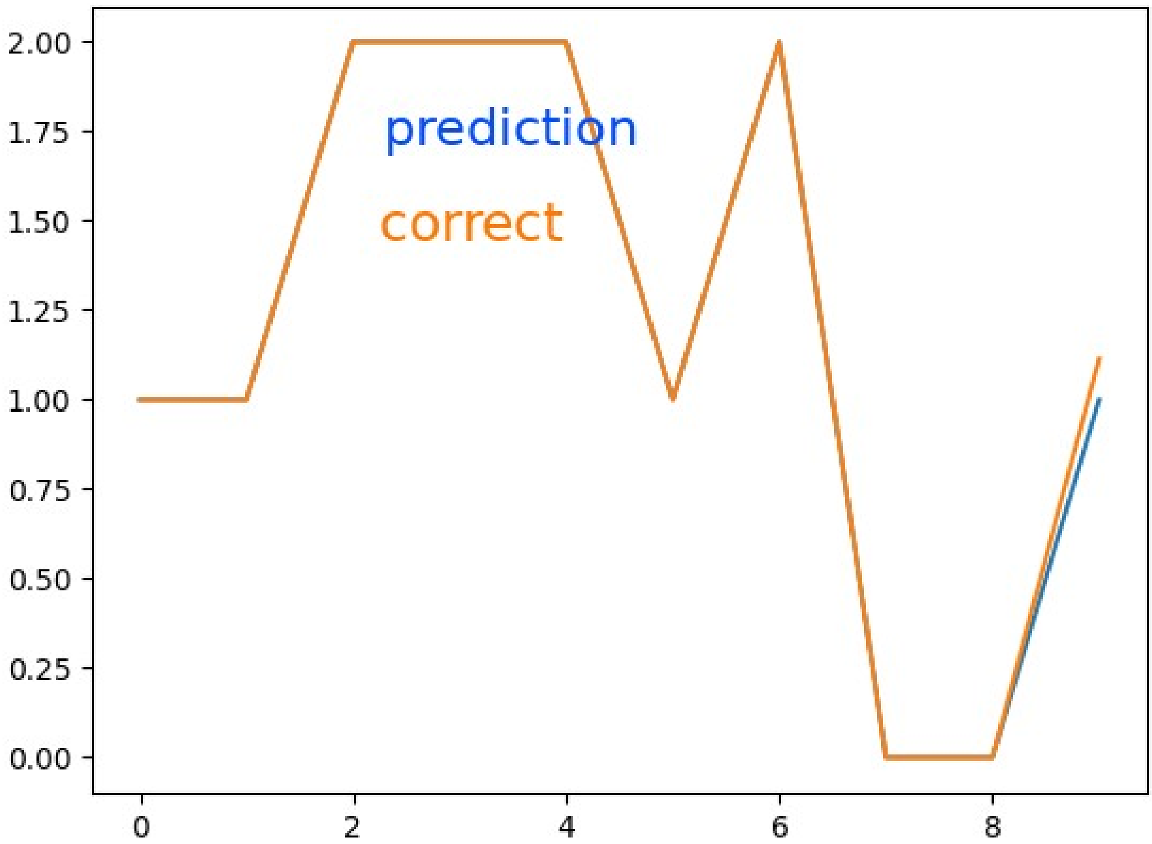
[1 ,2 ,1 ,1 ,0 ,1 ,1 ,0 ,4,2,1,1,2,3,1.6450815] Prediction is :python

extract_diff_length_of_timelines.py 10 16 1.6450815

Correct is :python extract_diff_length_of_timelines.py 10 161



prediction for 1 year
[1 1 2 2 2 1 2 0 0 1]
[1, 1, 2, 2, 2, 1, 2, 0, 0, 1.1137747]
Prediction = 1.1137747
Correct = 1



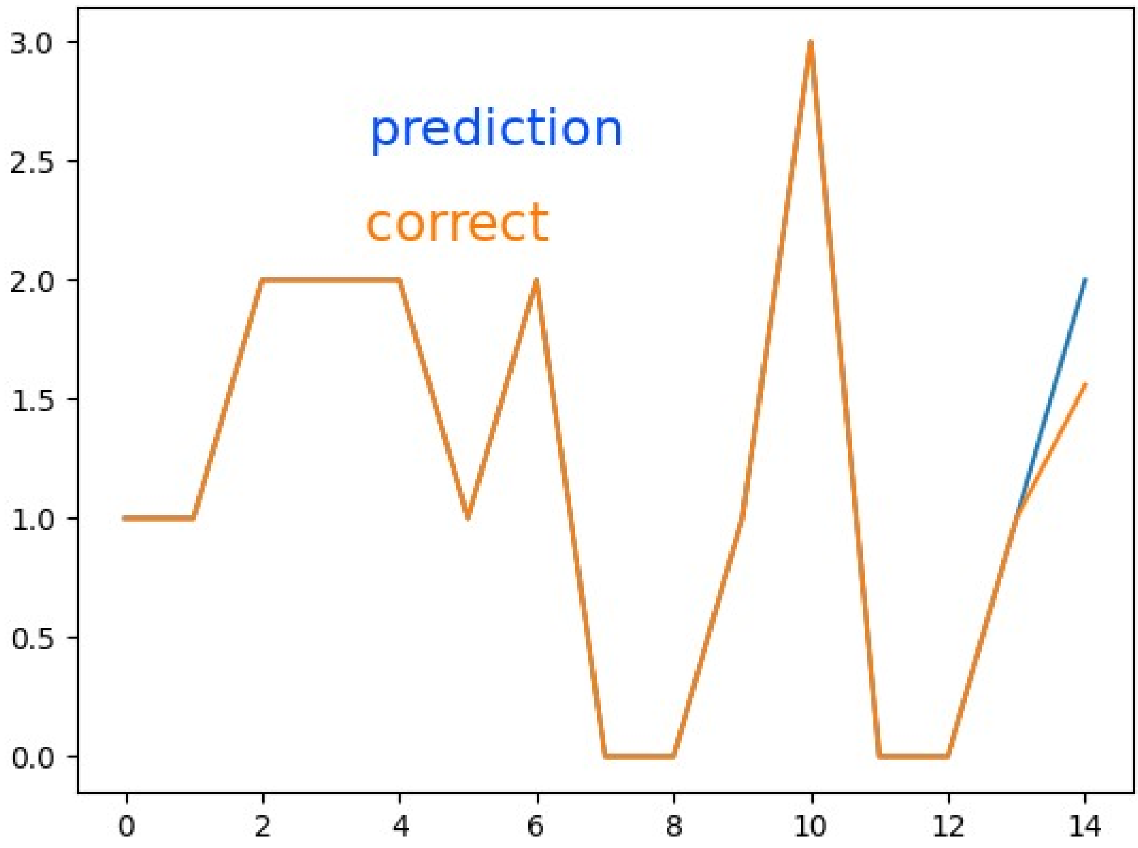
prediction for 5 years

[1 ,1 ,2 ,2 ,2 ,1 ,2 ,0, 0,1,3,0,0,1,2]

[1 ,1 ,2 ,2 ,2 ,1 ,2 ,0, 0 ,1,3,0,0,1,1.5597068] Prediction is :python

extract_diff_length_of_timelines.py 10 16 1.5597068

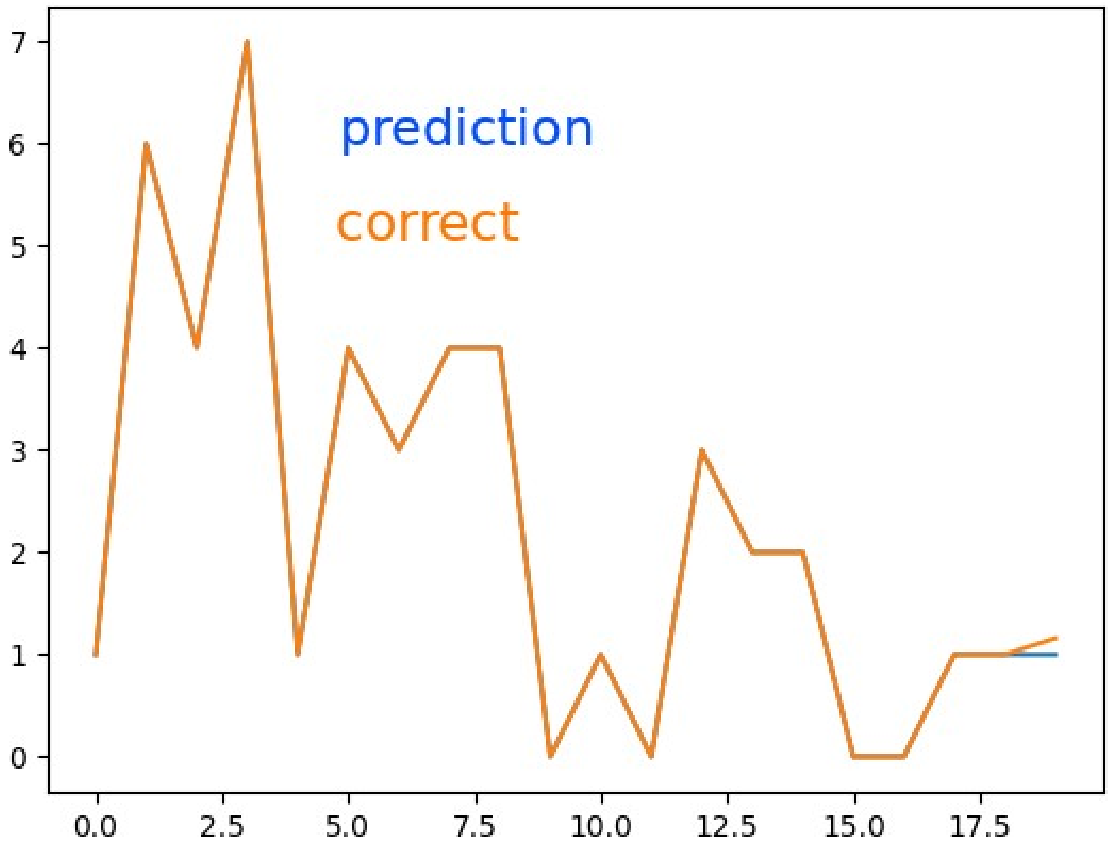
Correct is :python extract_diff_length_of_timelines.py 10 162



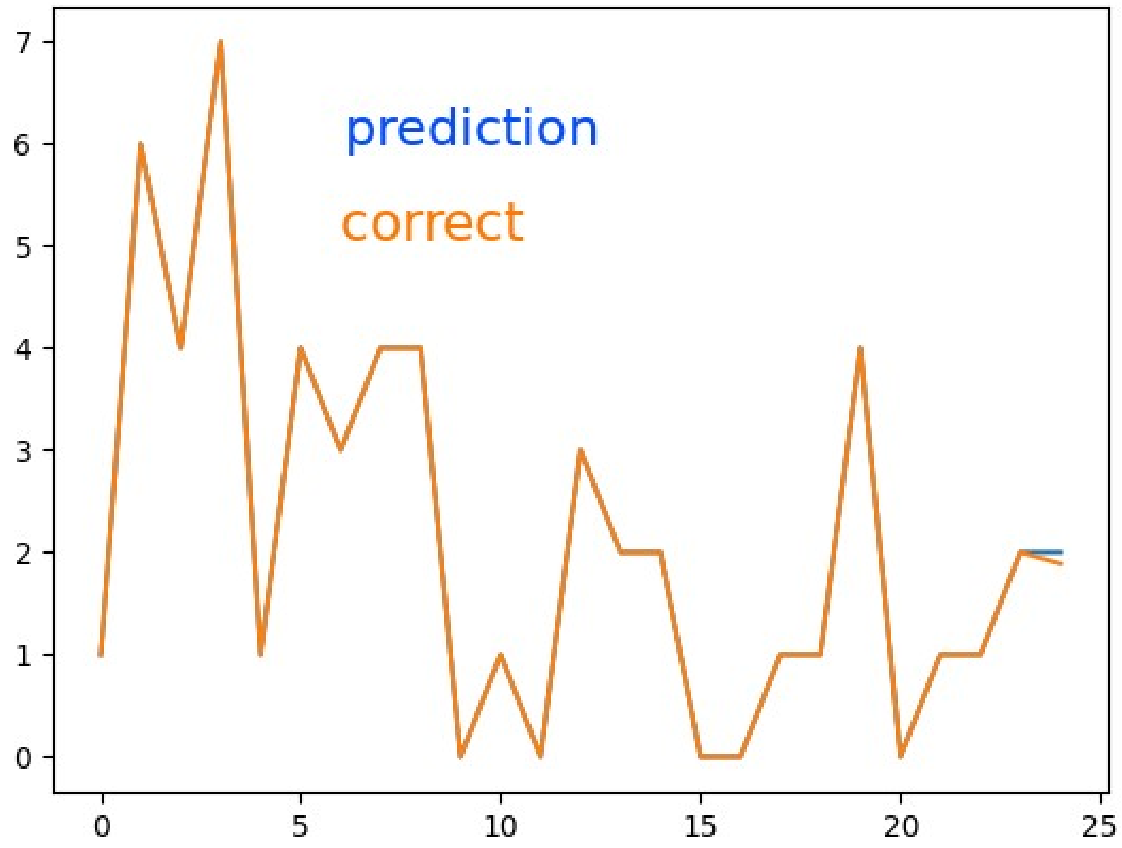
For a time series of 20

prediction for 1 year loss:python extract_diff_length_of_timelines.py 10 160.7404102678881602 accuracy:python extract_diff_length_of_timelines.py 10 16 0.9588014981273408 [1 6 4 7 1 4 3 4 4 0 1 0 3 2 2 0 0 1 1 1]

[1, 6, 4, 7, 1, 4, 3, 4, 4, 0, 1, 0, 3, 2, 2, 0, 0, 1, 1, 1.1598287]
correct=1
prediction=1.1598287



prediction for 5 years
[1 6 4 7 1 4 3 4 4 0 1 0 3 2 2 0 0 1 1 1 4 0 1 1 2 2]
[1 6 4 7 1 4 3 4 4 0 1 0 3 2 2 0 0 1 1 1 4 0 1 1 2 1.8887237]
correct=2
prediction=1.8887237



4)Training times as a function of input data size

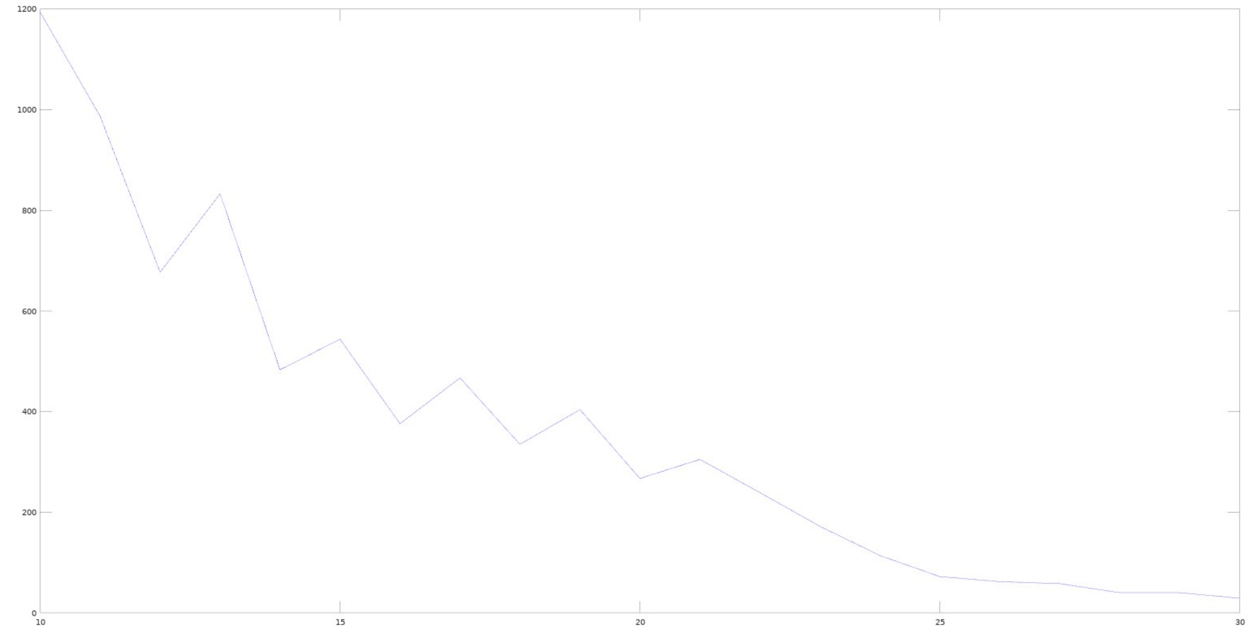
For date length 10, 1194 articles
330.084248505 seconds.

For date length 11,987 articles
147.79023395500008 seconds

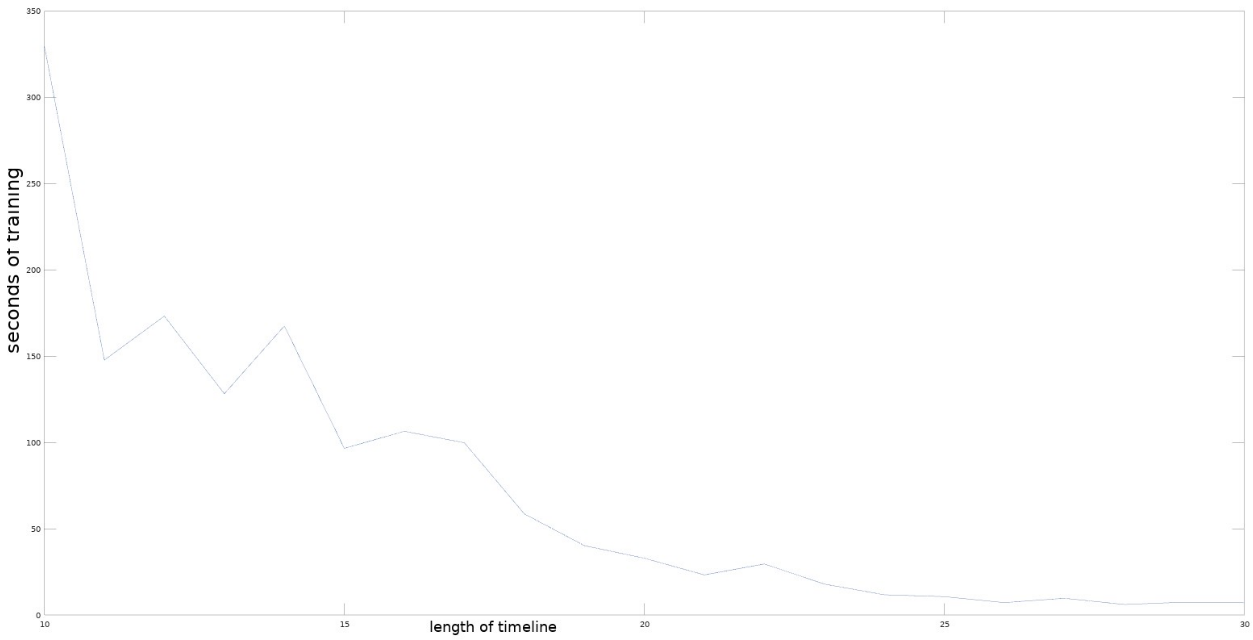
For date length 12, 677 articles
173.29364429700036 seconds
.
.
.
For date length 29, 40 articles
7.437733655999182 seconds

For date length 30, 29 articles
7.076767807999204 seconds

In the diagram below is the length of the time series (optimizer), we use adam (the algorithm at the end).10,11,12...30) to the number of articles each time series has.



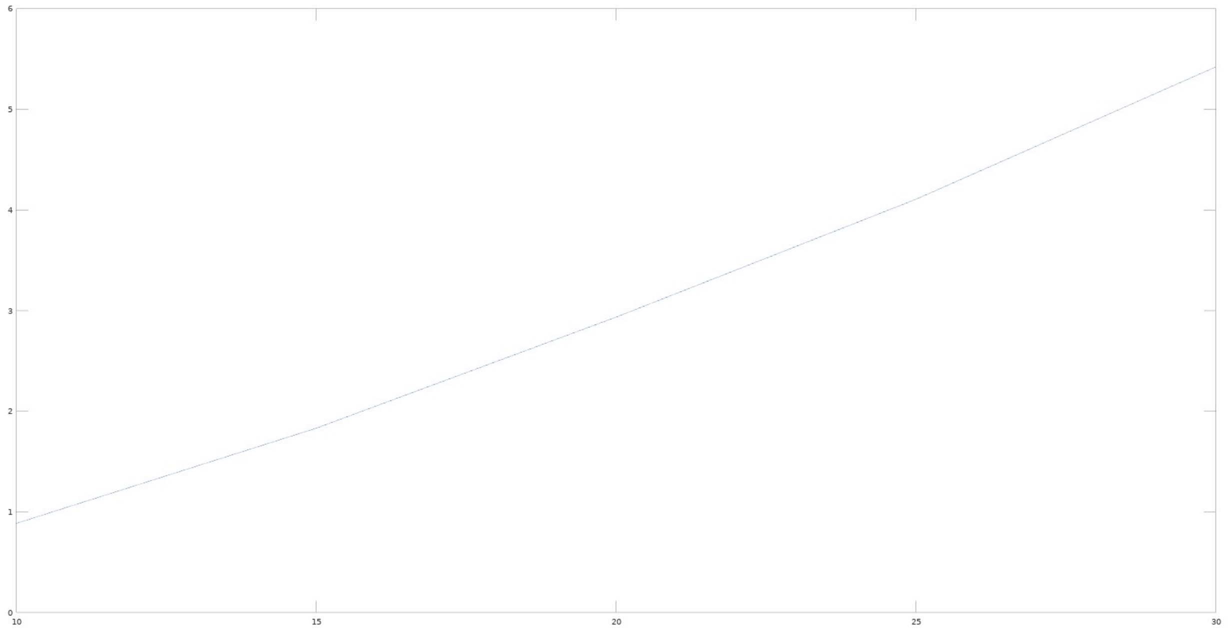
In the diagram below is the length of the time series (optimizer), we use adam (the algorithm at the end).10,11,12...30) to the training time of each model.



5) Network response time

The network response time, i.e. reading a file and the network response as a prediction for a time series of:python extract_diff_length_of_timelines.py 10 16

10→0.8857718209983432 seconds 15
→ 1.8314685660006944 seconds 20→
2.935860141999001 seconds 25→
4.106523941001797 seconds 30→
5.422715535998577 seconds



6)Final trained values of the parameters learning rate = 0.001 beta_1 = 0.9

beta_2 = 0.999
epsilon=0.00000001
decay=0.0

7) Training algorithm

Adam optimization algorithm

$Vdw=0, Sdw=0, Vdb=0, Sdb=0$ $On iteration t$:python

extract_diff_length_of_timelines.py 10 16

$$Compute dW, db using causal mini-batch\ Vdw=b_1 Vdw+(1-b_1)dw,\ Vdb=b_1 Vdb+(1-b_1)db$$

$$Sdw=b_2 Sdw+(1-b_2)dw_2,\ Sdb=b_2 Sdb+(1-b_2)db_2$$

$$V^{corrected}_{dw}=\frac{Vdw}{1-b_{t_1}},\ V^{corrected}_{db}=\frac{Vdb}{1-b_{t_2}}$$

$$S^{corrected}_{dw}=\frac{Sdw}{1-b_{t_2}},\ Sdb_{corrected}=\frac{Sdb}{1-b_{t_2}}$$

$$W=W-a\frac{Vdw_{corrected}}{\sqrt{(Sdw_{corrected})}+e},\ b=b-a\frac{Vdb_{corrected}}{\sqrt{(Sdb_{corrected})}+e}$$

SOURCES:

Adam optimization algorithm.

-https:python extract_diff_length_of_timelines.py 10 16//www.youtube.com/watch?v=JXQT_vxqwIs&fbclid=IwAR1lHXiNhduMfPWXKXTCJYoCIGH8XWcHgMO_D5EWYwUU1 bFLqeqXQed064s

Examples for implementing an lstm recurrent network in keras.

- https:python extract_diff_length_of_timelines.py 10 16//machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networkspython-keras/?
fbclid=IwAR2qQE6zteIqyXHNtbNOoioz74qRAWadTah5VV0NV3A4N67Pnm9823jNYpQ