LSTM Model for Temperature prediction at Capital Center

By Prasann Pandya

Problem

Using Tensorflow and Python, build and LSTM model that will predict the temperature for the next day at the Capital Center, using the historical precipitation and temperature data in the supplied data set.

Data Exploration

The Data consists of temperature and precipitation data from 28 sites all around North America recorded over almost 1 and half years (572 days).

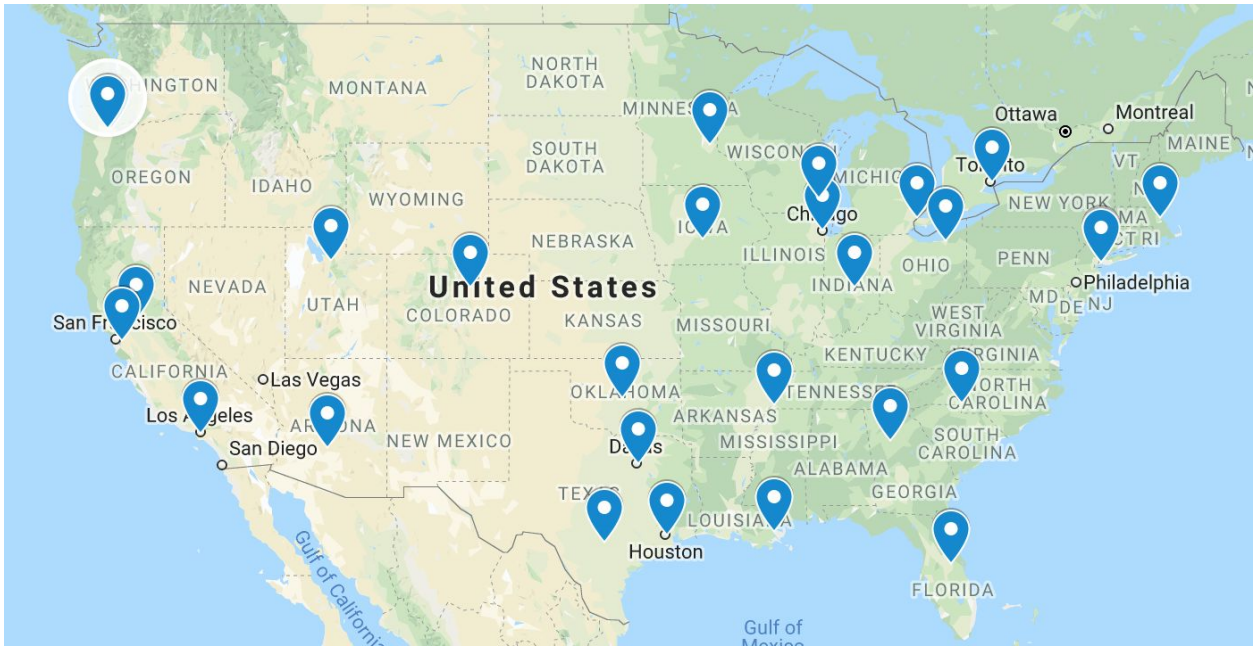

Figure 1: Sites contained in the data

The above Figure 1 shows the sites contained in the data. The highlighted pin in Oregon State is where the Capital Center is located. As you can see, the temperature at a particular day at Capital Center could be correlated to temperature and precipitation at other sites on previous days.

Another important factor to note is that the data does not contain values for every single site for each day. There is a lot of missing data. Some precipitation values are also labelled as NaNs.

The data is also not sorted by date. There are also duplicate values for certain centers for some days specially for “American Airlines Center” as shown in figure below.

id	date	venue	temperature	precipitation	day
1421700001	2017-07-01	American Airlines Center	0.851592	0.000000	0
1421700004	2017-07-01	Little Caesars Arena	0.662321	0.295720	0
1421700002	2017-07-01	Bankers Life Fieldhouse	0.656388	0.474319	0
1421700003	2017-07-01	American Airlines Center	0.851592	0.000000	0

Figure 2: Duplicate values of American Airlines Center on same day

The most useful column in the data for our purposes is Capital Center Temperature as we want to predict that. That column only contains data for 68 days. All the other values are missing. There are also inconsistencies in the interval between data points.

id	date	venue	temperature	precipitation	day
1521700024	2017-07-10	Capital Center	0.752486	0.0	9
1521700033	2017-07-11	Capital Center	0.792513	0.0	10
21700006	2017-10-18	Capital Center	0.549177	0.0	109
21700020	2017-10-20	Capital Center	0.630066	0.0	111
21700108	2017-11-01	Capital Center	0.482076	0.0	123

Figure 3: Inconsistencies in the day interval between values of Capital Center

As you can see in the figure above, valid Capital Center Temperature values go from day 9 to 10, 109, 111, 123, etc.

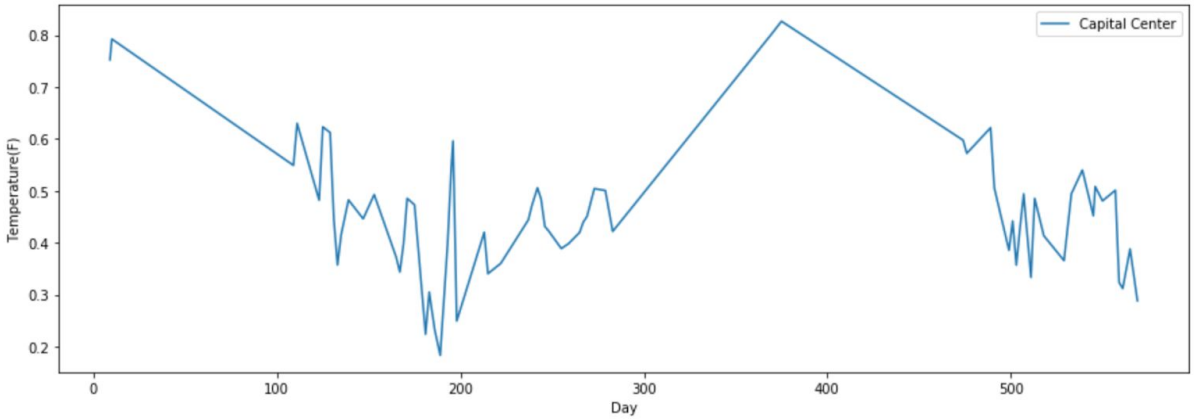

Figure 4: Capital Center temperature

The number of values for each center are on average 70. Below is a snapshow of all the sites along with their number of values in the dataset.

```
Total Places: 28

American Airlines Center : 148
Little Caesars Arena : 71
Bankers Life Fieldhouse : 71
Spectrum Center : 69
Chesapeake Energy Arena : 69
Vivint Smart Home Arena : 73
Wells Fargo Arena : 70
Madison Square Garden : 64
At&t Center : 74
Amway Center : 70
Staples Center : 146
Pepsi Center : 72
Golden 1 Center : 72
Quicken Loans Arena : 72
Smoothie King Center : 70
State Farm Arena : 68
United Center : 70
Target Center : 72
Oracle Arena : 70
Fedex Forum : 73
Barclays Center : 70
Td Garden : 71
Fiserv Forum : 72
Capital Center : 68
Talking Stick Resort Arena : 73
Scotiabank Arena : 70
Moda Center : 74
Toyota Center : 70
```

Figure 5: Sites with no. of values

Data Preprocessing

The goal of preprocessing the data is to prepare it to be fed into a neural network for prediction. Because we need to predict the Capital Center Temperature for the next day, the following preprocessing steps were performed on the data:

1. Scaling the data so that all values lie between 0 and 1
2. Sorting the data by date
3. Creation of a new dataframe with 1 day increments and 56 columns (28 for each site's temperature and 28 for each site's precipitation for the day)
4. Replacement of missing values with an unexpected value (-1) and removal of duplicate values

The first step was to scale the temperature and precipitation data so all values fall between 0 and 1. This was done using MinMaxScaler from Sklearn library. For simplicity, the dates were converted to number of days from the first day (2017-07-01) in the dataset. This was performed using datetime library in-built in python. Then, the data was sorted by day starting at day 0 and going till day 571.

Now, since there are multiple values for each column in the dataset corresponding to different sites, columns were created for each site's temperature and precipitation. All the values were initially populated with an unexpected value (-1) as there are many missing values and then the actual given values for each day were transferred from the original data to this new dataframe. The snapshot of the newly created data is shown below.

	day	american_airlines_center_temperature	little_caesars_arena_temperature	bankers_life_fieldhouse_temperature	spectrum_center_temperature
0	0.0	0.851592	0.662321	0.656388	-1.000000
1	1.0	0.731846	0.749979	-1.000000	0.693658
2	2.0	0.840896	-1.000000	-1.000000	-1.000000
3	3.0	0.833793	0.722821	-1.000000	-1.000000
4	4.0	-1.000000	0.726414	-1.000000	-1.000000
5	5.0	0.831370	-1.000000	0.759171	-1.000000
6	6.0	-1.000000	-1.000000	-1.000000	-1.000000
7	7.0	-1.000000	-1.000000	-1.000000	-1.000000
8	8.0	0.770452	-1.000000	-1.000000	-1.000000
9	9.0	-1.000000	-1.000000	-1.000000	-1.000000

Figure 6: New reformed dataframe

As you can see above, the columns were renamed with convention "site_name_temperature" and "site_name_precipitation". Since there are 28 sites, this resulted in 56 columns.

All the ‘NA’ values were replaced by -1. This value was chosen to be used for missing value as this value has no meaning in the dataset since all the values are between 0 and 1. The idea is that ‘-1’ values will be learned over time by the neural network as being missing values.

Since, we want to predict temperature value for the next day at Capital Center, the temperature values in the capital center column were shift up by 1 to create the target variable.

Finally, all the missing values of the target variable were removed and the values of only 68 days were kept for all columns.

Model Development

After preprocessing the data, the first step before feeding it into a neural network is to split the data into training, validation and test sets. So, the data was split into 80% training, 10% validation and 10% test set. The reason for splitting into three sets instead of two is that, the model uses validation set to improve the training and the model gets tested on completely unseen data. The validation set is used to check the training of the neural network and the test set is then used to test the performance on unseen data.

Since, the dataset is very small, there were 54 values used for training, 7 for validation and 7 for testing. After the data was split into three sets, the data now needs to be fed into a neural network.

The neural network used for this problem is Long Short Term Memory network (LSTM). LSTMs are good at learning long sequences of data. They are good at extracting patterns from long sequences. The downside of LSTMs is they require large amount of data.

After lot of trial and error, the network architecture designed for this problem is shows below:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_20 (LSTM)               (None, 1, 512)            1165312
_________________________________________________________________
dropout_25 (Dropout)         (None, 1, 512)            0
_________________________________________________________________
lstm_21 (LSTM)               (None, 1, 256)            787456
_________________________________________________________________
dropout_26 (Dropout)         (None, 1, 256)            0
_________________________________________________________________
lstm_22 (LSTM)               (None, 256)               525312
_________________________________________________________________
dropout_27 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_11 (Dense)             (None, 1)                 257
=================================================================
Total params: 2,478,337
Trainable params: 2,478,337
Non-trainable params: 0
```

Figure 7: Neural Network Model Architecture

From the image above, the model architecture can be described as having three LSTM layers. The first layer with 512 hidden neurons, second with 256 and third with 256. The output of the last LSTM layer is then fed into a fully connected dense layer as output. The shape of the output is ‘1’ since we only need to predict one value. Dropout layers are added after each layer to prevent overfitting. The network architecture was coded using the Keras API of Tensorflow.

The input to LSTM has to have a 3D shape:

- No. of samples (54)
- Time_step (1)
- Features (56)

So, a matrix with an input shape of [54, 1, 56] is used as input to the first LSTM.

The reason to use more hidden layers (3) and more neurons in each layer (256) is to learn complex patterns from the dataset as there are 56 features as long as the model did not overfit.

The loss function used for this model is Mean Squared Error. Mean Squared Error finds the difference between predicted and true values and provides a good estimate on performance on time-series prediction problems. The loss function is used as a metric to find the best model.

The optimizer used for this problem is “Adam” (Adaptive moment optimization). It is a standard used for most recurrent networks models. The optimizer is responsible for calculating the gradients of the weights of the network.

To train the network, a model checkpointer was created using “Keras.Callbacks” to save the model weights for model at epoch with best validation loss. The number of Epochs were chosen as 30 after trial and error.

Model Evaluation and Validation

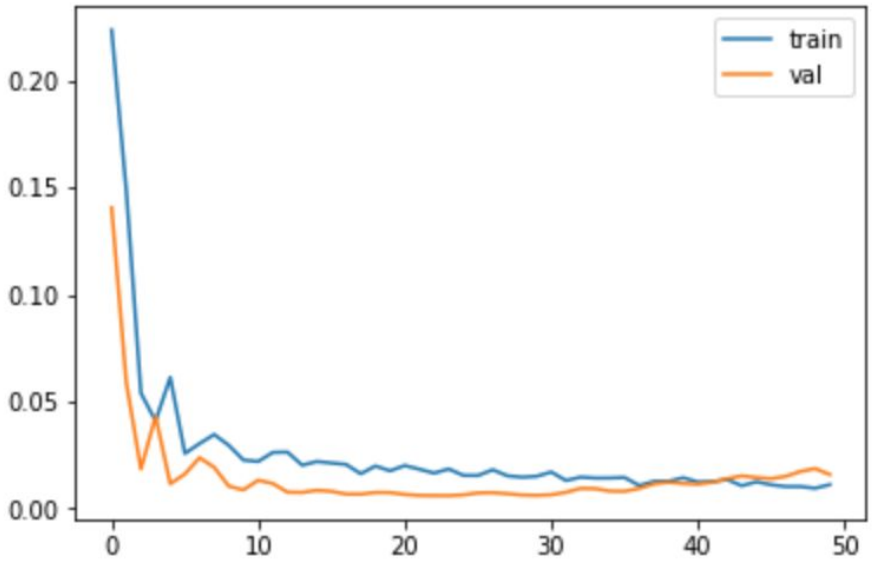

Figure 8: Plot of the Training and Validation loss vs no. of epochs

As shown in the figure above, the validation loss goes down almost at the same rate as training loss. This means that the model is not overfitting or underfitting. Around epoch 40, the model starts to overfit and so the validation loss starts increasing. Hence, the appropriate no. of epochs for this model would be 30.

To test the model, the scaled data was converted back into real temperature values in Fahrenheit. Also, the Root Mean Squared Error (RMSE) value was calculated on the test dataset as a metric to test the performance of the model. The RMSE value for the test set was found to be around 16 for the best model. This is not a good sign as this means that the real value and predicted value are off by 16 Farenheit on average. After changing the model parameters and much trial and error, the test set RMSE value remained around the same value. To investigate this further, here's a plot of the real and predicted values from the test set.

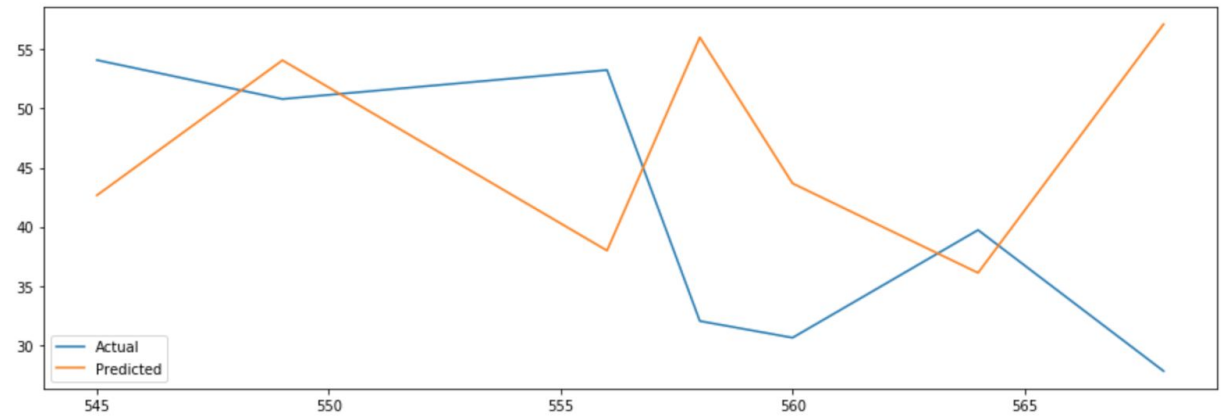

Figure 9: Plot for Real and Predicted values for Test Data

As it can be seen from the plot above, the real and predicted value around Day 550 and Day 565 are close to each other and all other values are off by a big margin. So, the model does not generalize well for unseen new data. Suggestions for improving the model are provided in the next section.

To investigate whether the model worked well on data it has seen, I calculated the RMSE for validation set and train set.

For validation set, the RMSE was lowest at around 8. The plot of predicted and actual values is shown below.

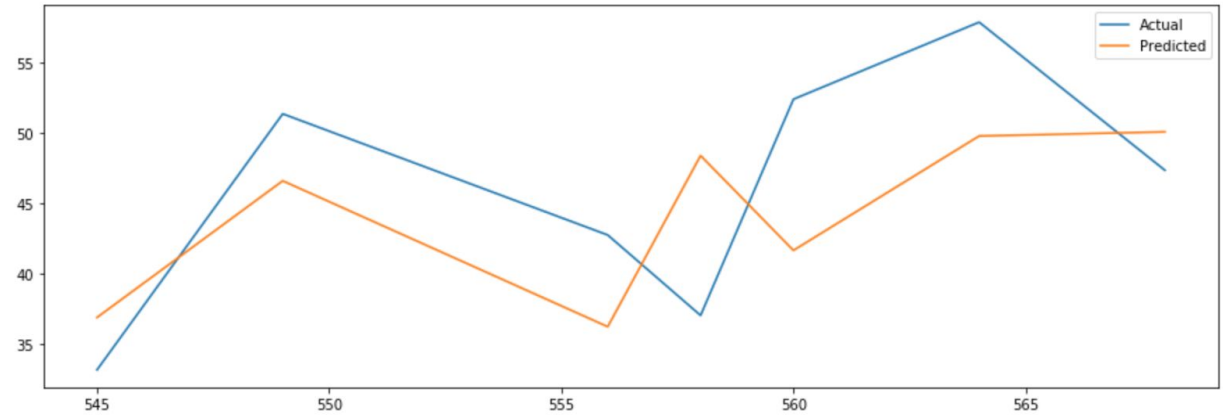

Figure 10: Plot Real and Predicted Values for Validation Data

As you can see above, the predicted values for validation set try to follow the pattern except for one data point between Day 555 and Day 560.

For the train set, the plot is shown below.

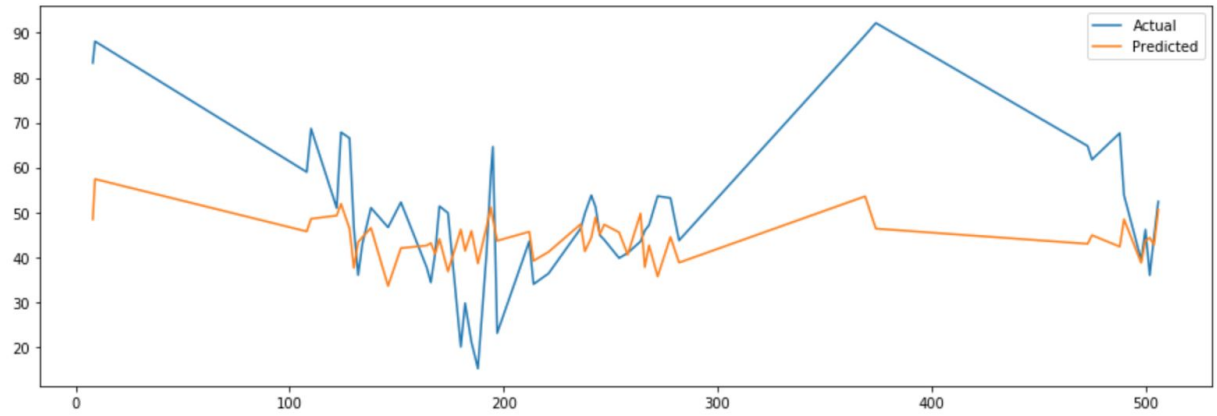

Figure 11: Plot Real and Predicted Values for Training Data

For the training data, you can see that the model is able to copy the pattern of the data. However, the model is not able to do well where there are sudden big changes in temperature and changes in time interval between values.

Conclusion and Future Improvements

As mentioned before, LSTM models require a lot of data to train and perform well. This problem only had 68 values in the dataset with lot of missing data points for different centers. The network did not perform well on the test dataset but during evaluation we noticed above that the network performed better than random and tried to capture the pattern in validation set.

There many different ways the model can be improved further. They are listed below:

Adding more features:

Currently to predict the temperature for next day, the model only sees features for the previous day. More features can be added by also taking into account values 2-5 days before to uncover hidden patterns. This could improve the model but also increase the dimensionality of the dataset as any additional day of data would double the dataset. For e.g., adding two days of data would lead to 112 features (56*2).

Also, month of the year can be added as a feature in the dataset since temperature in a particular month is correlated with season of the year.

Since, the time intervals between values of the target variable (Capital Center Temperature) are inconsistent, it is hard for the model to learn an appropriate sequence for the data. So, a new variable that calculated the difference between current and previous value could be added and experimented with to see if it affects the model performance.

Adding more data:

There are many API's online for providing weather data on a particular day on a particular city. This could be used to add missing data and potentially years more of weather data at these locations. Using this, we would get a lot more data for LSTM to learn the long sequence and predict more accurate temperature.

Trying different model architecture:

LSTMs are better than Recurrent Neural Networks for capturing long term dependencies and learning long sequences very well. However, there are many different types of LSTM architectures. There is also another type of recurrent network that is good at capturing long term dependencies called "Gated Recurrent Unit (GRU)". That could be tried to see if it improves model performance.

Overall, the model performs better than random but is not usable in real-world scenario. This is due to very small amount of data available to train as LSTMs require large amount of data. Without adding more data, it is unlikely that the model will perform well. In this case, other statistical techniques to forecast time-series models might be better suited.