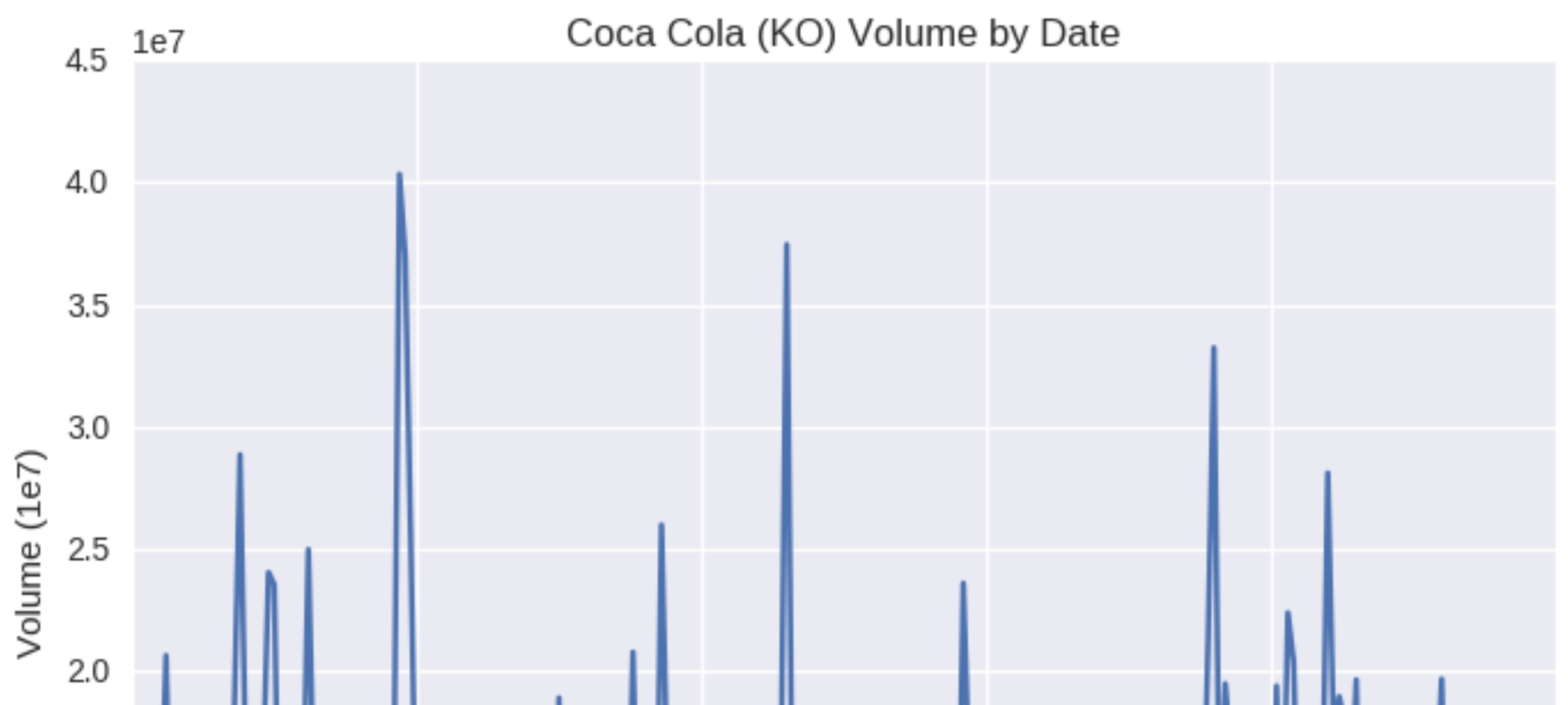
July 17, 2017 (/data-science-blog/2017/2/10/predicting-stock-volume-with-lstm)

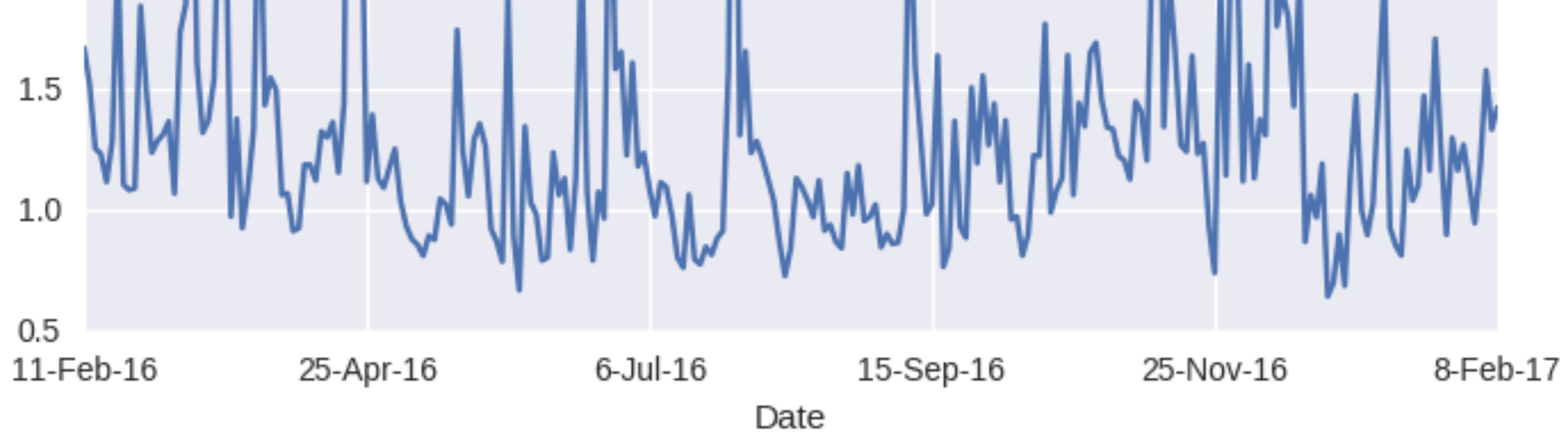
Predicting Stock Volume with LSTM (/data-science-blog/2017/2/10/predicting-stock-volume-with-lstm)

Michael Luk (/data-science-blog/?author=563c1649e4b0c01d8b265e60)

Much of the hype surrounding neural networks is about image-based applications. However, Recurrent Neural Networks (RNNs) have been successfully used in recent years to predict future events in time series as well. RNNs have contributed to breakthroughs in a wide variety of fields (http://karpathy.github.io/2015/05/21/rnn-effectiveness/) centered around predicting sequences of events. In this piece, however, we'll demonstrate how one type of RNN, the Long Short-Term Memory (LSTM) network, can be used to predict even financial time series data—perhaps the most chaotic and difficult of all time series.

For the sake of illustration, we'll specifically focus on predicting trends in Coca Cola's stock (KO) volume from this past year (see below). Volume is an important financial metric because changes in volume often precede price changes (http://www.investopedia.com/university/technical/techanalysis5.asp). As a result, predicting volume can be a useful tool for making informed decisions about stocks.

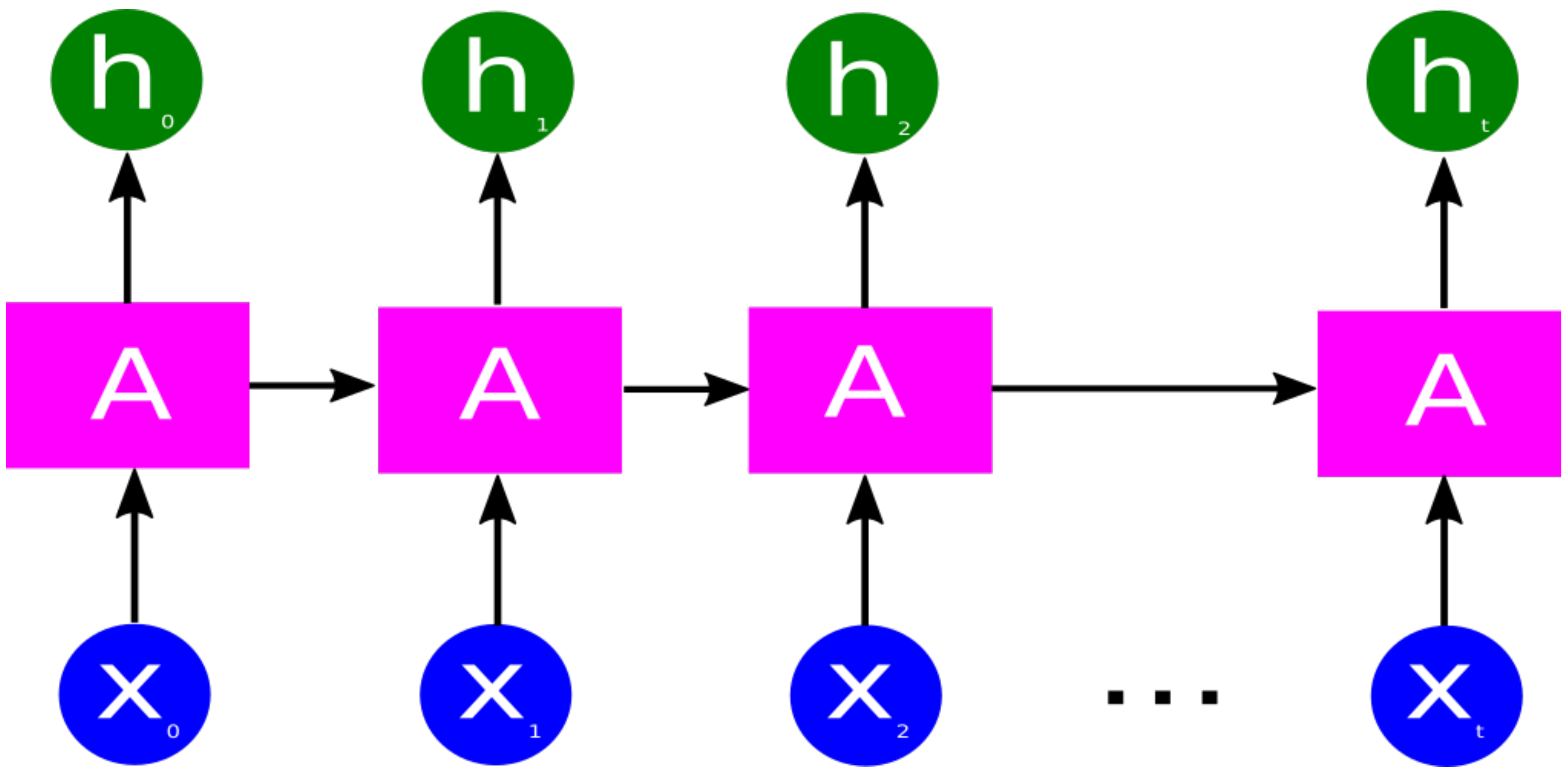




Before we dive into the training LSTMs, though, let's consider what we mean by the term "LSTMs".

What are LSTMs?

You can think about an RNN as a sequence of copies (http://colah.github.io/posts/2015-08-Understanding-LSTMs/) from the same neural network. Such a network can be trained so that each copy shares essential information with every other element in the sequence, making it possible to predict future sequential events based on past ones. Consider, for instance, a piece of neural network *A*, input value *xt* at time *t*, and output value *ht* at time *t*. The architecture of this RNN would have the following appearance:



Given the sequential nature of RNNs, they are perfect for predicting future events in a time series. When you train an RNN, you train it based on an entire sequence of events previous, making predictions based on their knowledge of a sequence of events. This memory of past events through a neural network has proven useful in a wide variety of domains.

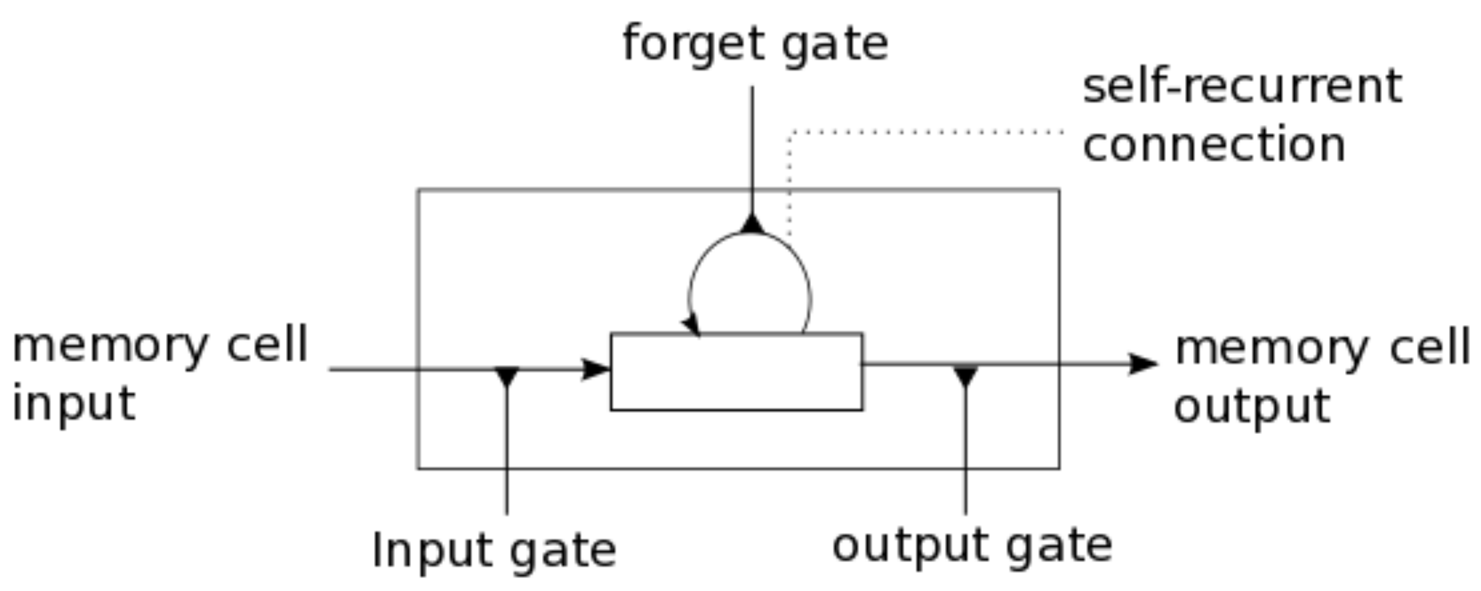
For instance, you can train an RNN to compose original Mozart scores (http://www.wise.io/tech/asking-rnn-and-ltsm-what-would-mozart-write) based on what it has learned about how Mozart strung together notes in a sequence. RNNs have also been used to great effect in text (http://karpathy.github.io/2015/05/21/rnn-effectiveness/) and handwriting (https://arxiv.org/abs/1308.0850) prediction, where predicting the next word in a sequence is about understanding the larger context of what has already been said.

Not all RNNs are created equal, though. Notably, standard RNNs are often difficult to train if they have long term dependencies (http://www-dsi.ing.unifi.it/~paolo/ps/tnn-94-gradient.pdf). Traditional RNNs tend to only learn short term dependencies when they are trained using standard back-propagation methods. Because backpropagation computes gradients (used to tune the network) by multiplying values for each element in an RNN sequence (see here (http://cs231n.github.io/optimization-2/) for a nice overview on backpropagation), we run into problems with long sequences. Specifically, if the multiplied values are greater than 1, the gradient can explode (http://neuralnetworksanddeeplearning.com/chap5.html#discussion_why), and if the values are less than 1, the gradient can vanish (http://neuralnetworksanddeeplearning.com/chap5.html#discussion_why). Gradient instability of this sort can prematurely slow or stall the training process, making it impossible for the RNN to learn long-term dependencies.

This means, for instance, that while we may successfully predict the last word in a short, simple sentence like "the clouds are in the *sky*" with standard RNNs, it would more challenging in a sentence requiring greater long-term dependency recognition like "I grew up in France...I speak fluent *French*" (see these examples and more here (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)). The word "French" in the sentence depends on the network learning to look far enough back in the message to see that the speaker "grew up in France."

While it's possible to train RNNs to recognize long-term dependencies by applying supervised weights (http://www-dsi.ing.unifi.it/~paolo/ps/tnn-94-gradient.pdf#page=5) if there is prior information about how such dependencies occur, this is often not feasible in practice. The LSTM variant of RNNs corrects this long-term learning deficiency, however, by introducing additional neural network layers to the basic RNN model.

Specifically, instead of having a single neural network layer (within the "A" rectangles in the diagram above) at each step in the sequence, LSTMs use multiple interacting layers at each step. They employ a series of strategically-placed "gates" (http://www.deeplearning.net/tutorial/lstm.html#lstm) to temper the interactions between each cell and the rest of the sequence of cells, giving even the most distant step (http://colah.github.io/posts/2015-08-Understanding-LSTMs/) of a sequence a role in prediction. Thus, in place of the "A" in our simple RNN model above, we now have:



An in-depth discussion of all of the features of a LSTM cell is beyond the scope of this article (for more detail see excellent reviews here (https://arxiv.org/abs/1506.00019) and here (http://colah.github.io/posts/2015-08-Understanding-LSTMs/)). However, the bottom line is that LSTMs provide a useful tool for predicting time series, even when there are long-term dependencies--as there often are in financial time series among others such as handwriting and voice sequential datasets.

Using LSTMs to predict Coca Cola's Daily Volume

Now, let's train an LSTM on our Coca Cola stock volume data for a demonstration of how you use LSTMs. We'll be working with Python's Keras (https://keras.io/) library to train our neural network, so first let's take our KO data and make it Keras compliant.

We'll first read in the data, then follow Jakob Aungiers' method (http://www.jakob-aungiers.com/articles/a/LSTM-Neural-Network-for-Time-Series-Prediction) for transforming the data into usable form in Keras.

```
>>> df = pd.read_csv('ko.csv')
>>> #Reorder Data to start in 2016 and end in 2017:
>>> ko_df = df.sort_index(ascending=False).reset_index(range(len(df)))[[1,6
```

First, let's define a function (adapted from Jakob Aungier (http://www.jakob-aungiers.com/articles/a/LSTM-Neural-Network-for-Time-Series-Prediction)) to normalize the volume data based on the first entry in a month. This will make the (otherwise gigantic) numbers more managable for the LSTM:

```
>>> def normalise_windows(window_data):
>>>     normalised_data = []
>>>     for window in window_data:
>>>         window.index = range(30)
>>>         normalised_window = [((float(p) / float(window[0])) - 1) for p
>>>         normalised_data.append(normalised_window)
>>>     return normalised_data
```

We then divide the data up into 30 day sequences (about a month), so the network will learn to predict patterns in volume based on the prior month's data. 90% of the sequences will go in our training dataset and 10% will go in our test dataset. Each sequence is a sliding window, shifting forward one day with each new sequence. This means that there is a constant overlap with prior windows and we will have plenty of training data for the demonstration.

```
>>> sequence_length = 30
>>> result = []
>>> for index in range(len(ko_df) - sequence_length):
>>>    result.append(ko_df['Volume'][index: index + sequence_length])
>>> result = normalise_windows(result)
>>> result = np.array(result)

>>> row = round(0.9 * result.shape[0])
>>> train = result[:row, :]
>>> np.random.shuffle(train)
>>> X_train = train[:, :-1]
>>> y_train = train[:, -1]
>>> X_test = result[row:, :-1]
>>> y_test = result[row:, -1]

>>> X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
>>> X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))
```

Now we're ready to create a simple LSTM. Here, we will use a 100 cell network, for the sake of simplicity, feeding in a single sequence of 100 dates at a time. The input layer takes in one sequence of 30 volumes organized by date. The output layer then gives a predicted value for the next time step based on the results of the 100 cell LSTM layer. While this is a simple model, if we wanted to, we could stack multiple LSTM layers on top of one another, to learn higher-level temporal representations (https://keras.io/getting-started/sequential-model-guide/). The principle remains the same, but higher-level temporal representations could lead to better results.

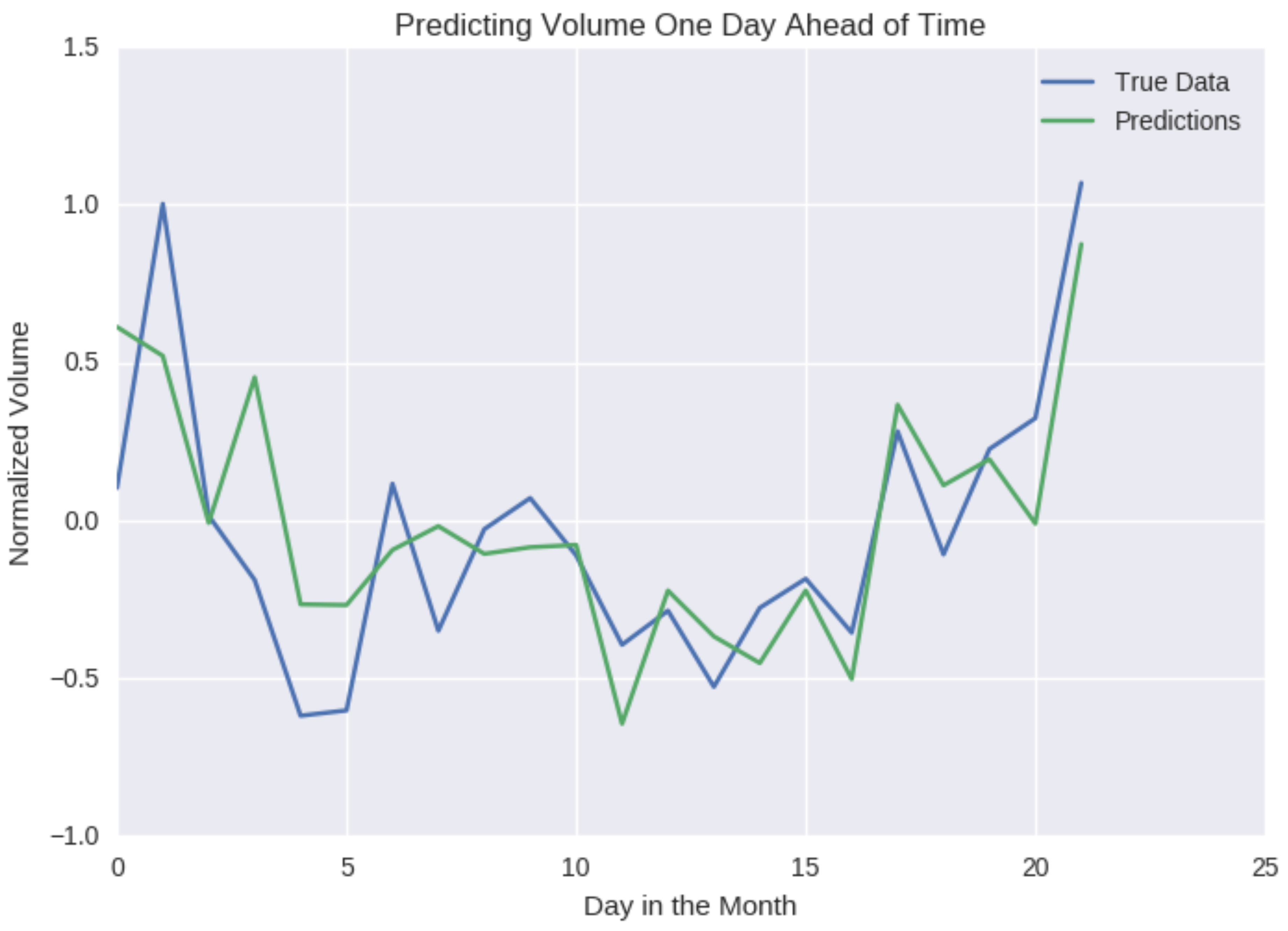
```
>>> from keras.models import Sequential
>>> from keras.layers import LSTM, Dense

>>> model = Sequential()

>>> model.add(LSTM(
>>>     input_dim = 1,
>>>     output_dim = 100
>>>     ))
>>> model.add(Dense(
>>>     output_dim = 1,
>>>     activation = 'linear'
>>>     ))

>>> model.compile(loss = 'mse', optimizer = 'rmsprop')
```

Now, let's fit the model. For this demonstration, we'll use 50 epochs (the number of passes over the training data):

```
>>> model.fit(X_train, y_train, nb_epoch = 50, validation_split = 0.05)
```

Let's make a few predictions and see how good our model is. First, let's predict each day one ahead of time:

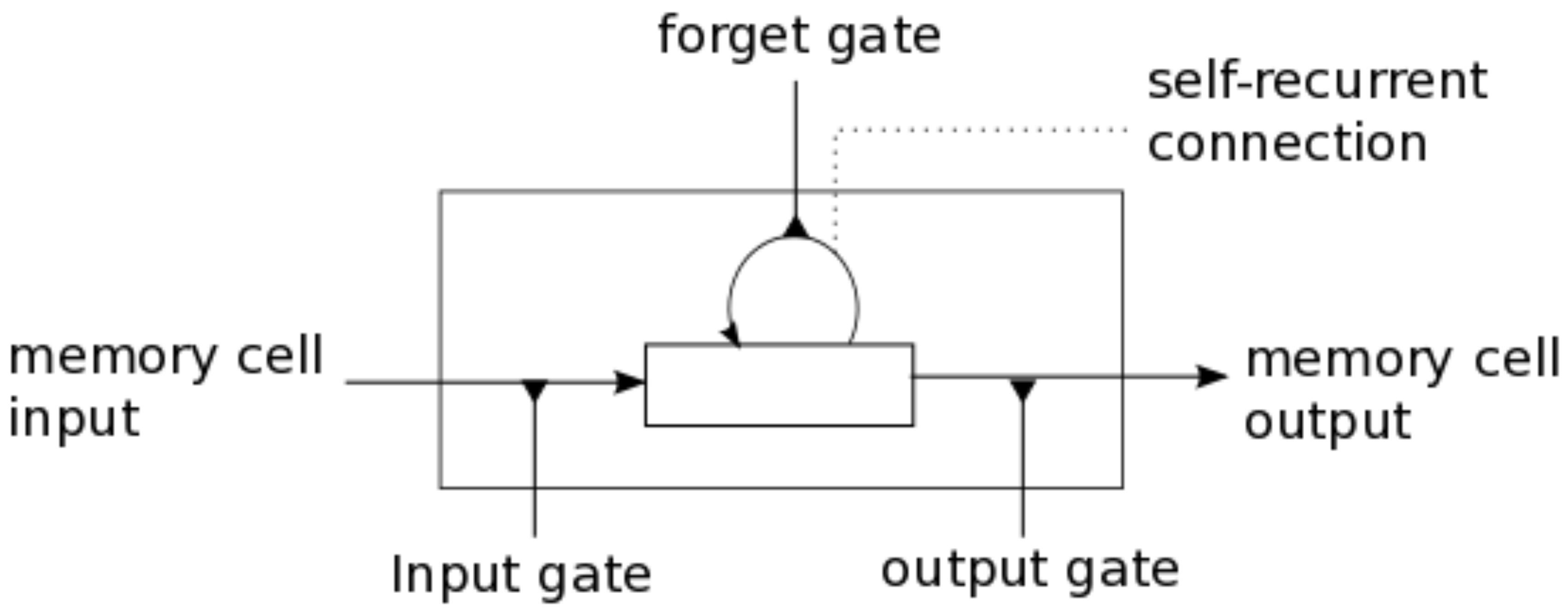
```
>>> predictions = model.predict(X_test)
>>> predictions = np.reshape(predictions, (predictions.size,))
```



Notice how well we can predict Coca Cola volume one day ahead of time. Over the course of the month that was held out as a test dataset, there is a close correspondence between the predictions and actual values. However, it isn't especially impressive to be able to predict volume only one day ahead of time. The model bases its information on the value previous to it, so any given prediction is unlikely to be too far off the mark.

How about if we could predict a month in advance, though? Let's see whether or not our model can successfully predict trends over the course of an entire month. We do this by making predictions on the basis of past predictions:

```
>>> from numpy import newaxis
>>> curr_frame = X_test[0]
>>> predicted = []
>>> for i in xrange(len(X_test)):
>>>     predicted.append(model.predict(curr_frame[newaxis,:,:])[0,0])
>>>     curr_frame = curr_frame[1:]
>>>     curr_frame = np.insert(curr_frame, len(X_test[0])-1, predicted[-1],
```



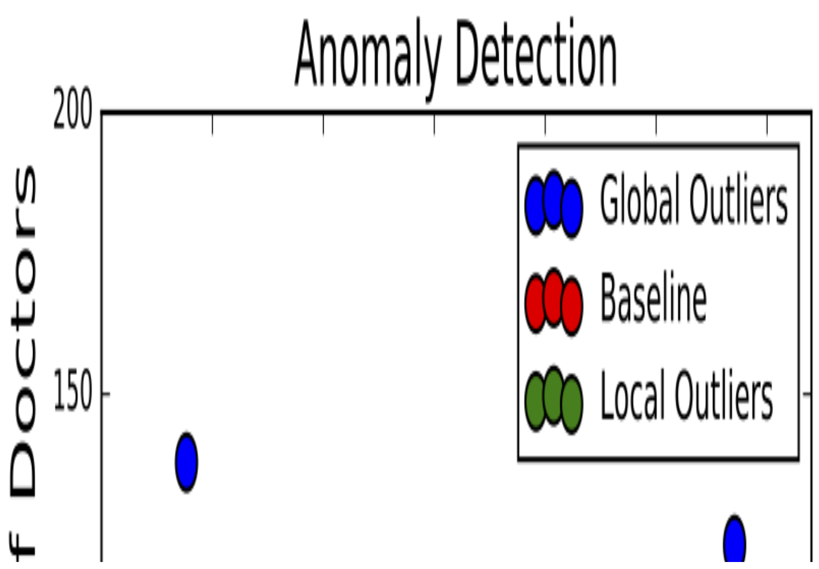
You can see that the results are actually not too bad, considering the LSTM is predicting a month's worth of volume data. While the predicted volumes are not perfectly aligned with the true values, they do follow the rough contours of the real historical volume data.

Concluding Thoughts

While LSTMs performed well out of the box in this context, be careful making any decisions based on these kinds of simplistic demonstrations. Tuning and cautious testing could reveal many areas for improvement. This simple LSTM does a decent job here, but prediction success could vary widely for different time intervals and different companies. No doubt the predictions could also be improved by adding in other relevant variables and financial expertise.

With intelligent use, LSTMs have proven again and again that they can effectively predict outcomes from even the most complicated time series or even sequence based data. Indeed, a lot of today's translation services that model languages as long sequences of words, can be well modeled by LSTMs as well.

FEATURED

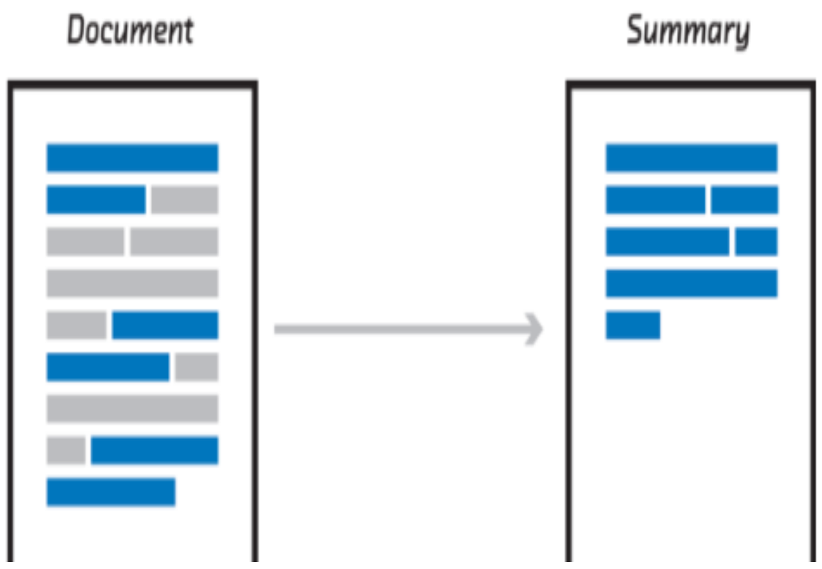


ANOMALY DETECTION: NETWORK INTRUSION DETECTOR (/DATA-SCIENCE-BLOG/2016/7/30/ANOMALY-DETECTION-NETWORK-INTRUSION-DETECTOR)

Anomaly detection is a common problem that can be solved using machine learning techniques. Simple density based algorithms provide a good baseline for such projects, and can be used to solve a variety of problems from defect detection in manufacturing to network attacks in IT.

Read More → (/data-science-blog/2016/7/30/anomaly-detection-network-intrusion-detector)

May 19, 2018



DOCUMENT SUMMARISATION (/DATA-SCIENCE-BLOG/2018/4/3/DOCUMENT-SUMMARISATION)

Automated text summarization through machine learning can be an extremely valuable tool to increase efficiency in both our everyday life and professional endeavors if the important information in a document can be extracted and accurately summarized.

Read More → (/data-science-blog/2018/4/3/document-summarisation)

Apr 18, 2018

Tagged: LSTM (/data-science-blog/?tag=LSTM), machine learning (/data-science-blog/?tag=machine+learning), deep learning (/data-science-blog/?tag=deep+learning), RNN (/data-science-blog/?tag=RNN), recurrent neural network (/data-science-blog/?tag=recurrent+neural+network), neural nets (/data-science-blog/?tag=neural+nets), time series (/data-science-blog/?tag=time+series), stock market (/data-science-blog/?tag=stock+market), prediction (/data-science-blog/?tag=prediction)

❤ 7 Likes

❤ Share

❤ Share

COMPANY

About Us > (https://sflscientific.com/about-us/)
Media > (https://sflscientific.com/media/)
Blog > (/blog-index)
Careers > (https://sflscientific.com/careers/)
Contact > (https://sflscientific.com/contact/)

SERVICES

Solutions > (https://sflscientific.com/solutions/)
Our Work > (https://sflscientific.com/our-work/)
Partners > (https://sflscientific.com/our-partners/)
Case Studies > (https://sflscientific.com/case-studies-index/)

(http://www.sfls

INDUSTRIES

Advertising & Marketing > (https://sflscientific.com/solutions/advertising-and-marketing)
Agriculture > (https://sflscientific.com/solutions/agriculture)
Consumer Electronics > (https://sflscientific.com/solutions/consumer-electronics)
Cybersecurity > (https://sflscientific.com/solutions/cybersecurity)
Insurance > (https://sflscientific.com/solutions/insurance)
Internet of Things > (https://sflscientific.com/solutions/internet-of-things)
Life Sciences > (https://sflscientific.com/solutions/life-sciences)
Manufacturing > (https://sflscientific.com/solutions/manufacturing)
Oil & Gas >

CAPABILITIES

Data Science & Predictive Analytics > (https://sflscientific.com/services/data-science)
Data Strategy & Business Case > (https://sflscientific.com/services/data-strategy)
Business Intelligence > (https://sflscientific.com/services/business-intelligence)
Information Management > (https://sflscientific.com/services/information-management)
Software Development > (https://sflscientific.com/services/software-development)
Scientific Advisory > (https://sflscientific.com/services/scientific-advisory)