

# LSTM by Example using Tensorflow



Rowel Atienza · Follow

Published in Towards Data Science

6 min read · Mar 17, 2017



Listen



Share

In Deep Learning, Recurrent Neural Networks (RNN) are a family of neural networks that excels in learning from sequential data. A class of RNN that has found practical applications is Long Short-Term Memory (LSTM) because it is robust against the problems of long-term dependency. There is no shortage of articles and references explaining LSTM. Two recommended references are:

Open in app ↗

Sign up

Sign in



Search



There is also no shortage of good libraries to build machine learning applications based on LSTM. In GitHub, Google's Tensorflow has now over 50,000 stars at the time of this writing suggesting a strong popularity among machine learning practitioners.

What seems to be lacking is a good documentation and example on how to build an easy to understand Tensorflow application based on LSTM. This is the motivation behind this article.

Suppose we want to train a LSTM to predict the next word using a sample short story, Aesop's Fables:

*long ago , the mice had a general council to consider what measures they could take to outwit their common enemy , the cat . some said this , and some said that but at last a young mouse got up and said he had a proposal to make , which he thought would meet the case . you will all agree , said he , that our chief danger consists in the sly and*

*treacherous manner in which the enemy approaches us . now , if we could receive some signal of her approach , we could easily escape from her . i venture , therefore , to propose that a small bell be procured , and attached by a ribbon round the neck of the cat . by this means we should always know when she was about , and could easily retire while she was in the neighbourhood . this proposal met with general applause , until an old mouse got up and said that is all very well , but who is to bell the cat ? the mice looked at one another and nobody spoke . then the old mouse said it is easy to propose impossible remedies .*

Listing 1. A short story from Aesop's Fables with 112 unique symbols. Words and punctuation marks are both considered symbols.

If we feed a LSTM with correct sequences from the text of 3 symbols as inputs and 1 labeled symbol, eventually the neural network will learn to predict the next symbol correctly (Figure 1).

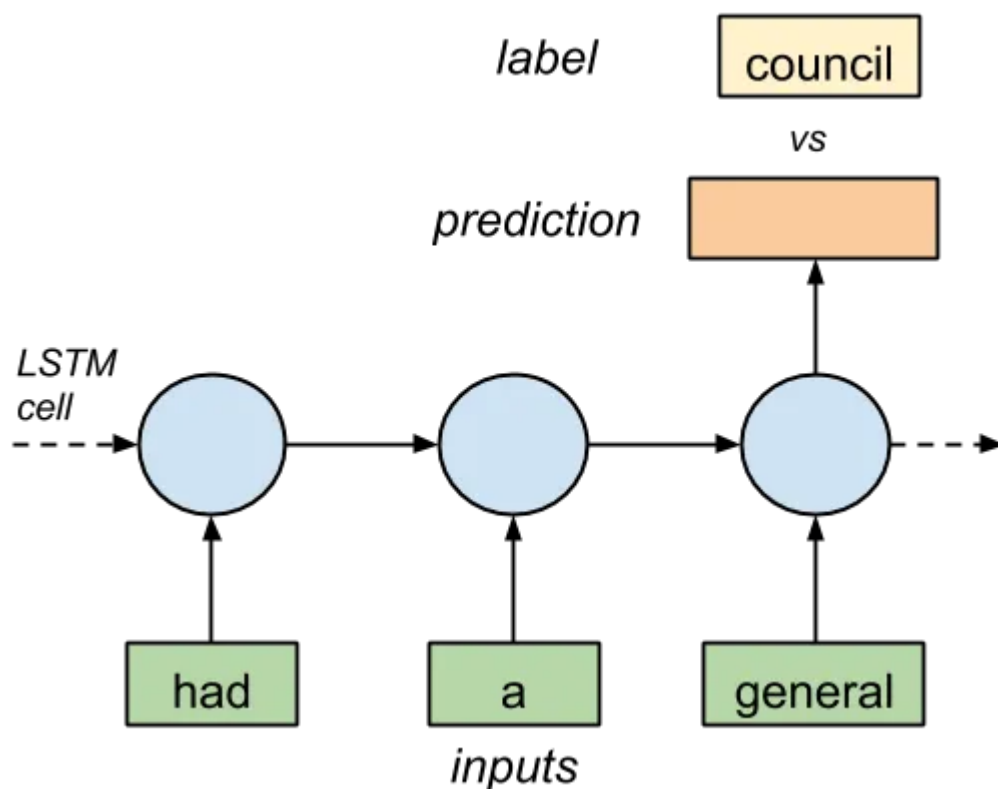


Figure 1. LSTM cell with three inputs and 1 output.

Technically, LSTM inputs can only understand real numbers. A way to convert symbol to number is to assign a unique integer to each symbol based on frequency of occurrence. For example, there are 112 unique symbols in the text above. The function in Listing 2 builds a dictionary with the following entries [ “,” : 0 ] [ “the” : 1

], ..., [ "council" : 37 ],..., [ "spoke" : 111 ]. The reverse dictionary is also generated since it will be used in decoding the output of LSTM.

```
def build_dataset(words):  
    count = collections.Counter(words).most_common()  
    dictionary = dict()  
    for word, _ in count:  
        dictionary[word] = len(dictionary)  
    reverse_dictionary = dict(zip(dictionary.values(),  
dictionary.keys()))  
    return dictionary, reverse_dictionary
```

Listing 2. Function for building the dictionary and reverse dictionary.

Similarly, the prediction is a unique integer identifying the index in the reverse dictionary of the predicted symbol. For example, if the prediction is 37, the predicted symbol is actually "council".

The generation of output may sound simple but actually LSTM produces a 112-element vector of probabilities of prediction for the next symbol normalized by the softmax() function. The index of the element with the highest probability is the predicted index of the symbol in the reverse dictionary (ie a one-hot vector). Figure 2 shows the process.

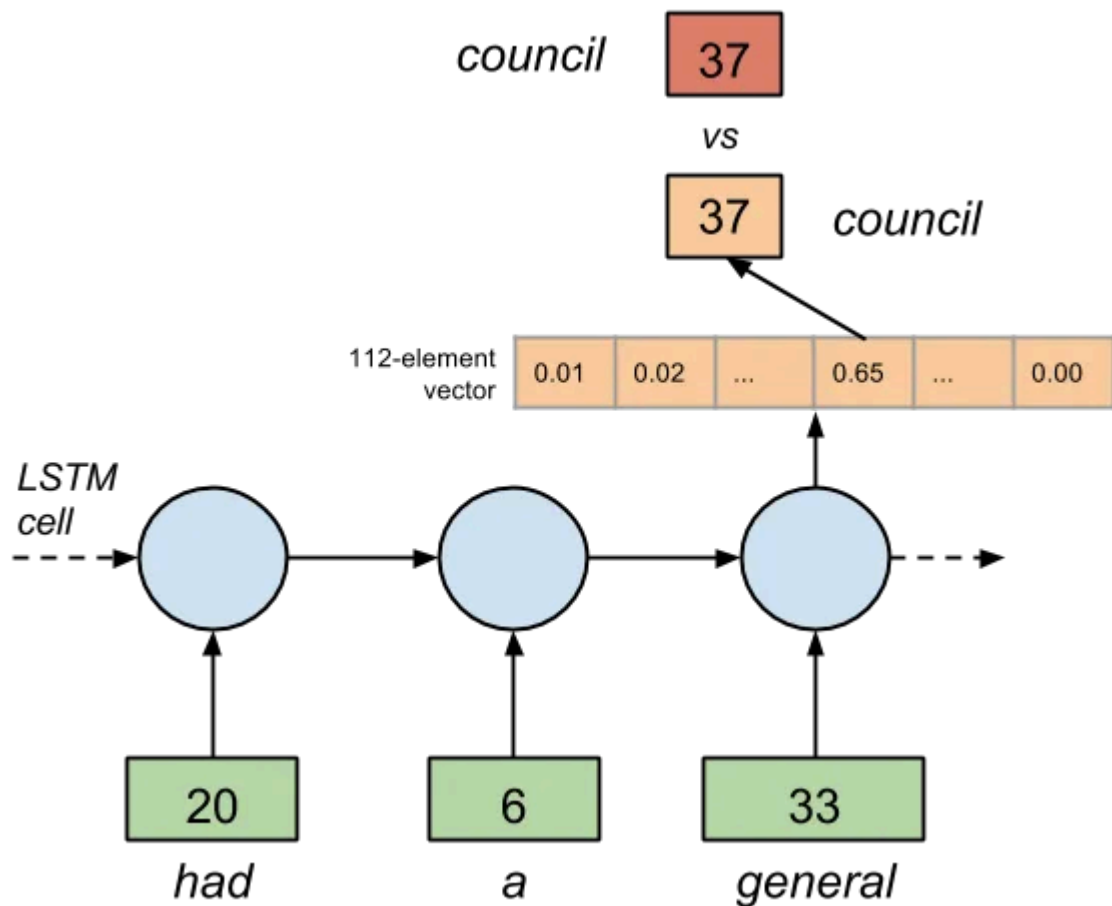


Figure 2. Each input symbol is replaced by its assigned unique integer. The output is a one-hot vector identifying the index of the predicted symbol in the reverse dictionary.

At the core of the application is the LSTM model. Surprisingly, it is very simple to implement in Tensorflow:

```
def RNN(x, weights, biases):

    # reshape to [1, n_input]
    x = tf.reshape(x, [-1, n_input])

    # Generate a n_input-element sequence of inputs
    # (eg. [had] [a] [general] -> [20] [6] [33])
    x = tf.split(x, n_input, 1)

    # 1-layer LSTM with n_hidden units.
    rnn_cell = rnn.BasicLSTMCell(n_hidden)

    # generate prediction
    outputs, states = rnn.static_rnn(rnn_cell, x, dtype=tf.float32)

    # there are n_input outputs but
```

```
# we only want the last output
return tf.matmul(outputs[-1], weights['out']) + biases['out']
```

### Listing 3. The model with a 512-unit LSTM cell

The trickiest part is feeding the inputs in the correct format and sequence. In this example, the LSTM feeds on a sequence of 3 integers (eg 1x3 vector of int).

The constants, weights and biases are:

```
vocab_size = len(dictionary)
n_input = 3

# number of units in RNN cell
n_hidden = 512

# RNN output node weights and biases
weights = {
    'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
}
biases = {
    'out': tf.Variable(tf.random_normal([vocab_size]))
}
```

### Listing 4. Constants and training parameters

In the training process, at each step, 3 symbols are retrieved from the training data. These 3 symbols are converted to integers to form the input vector.

```
symbols_in_keys = [ [dictionary[ str(training_data[i])]] for i in
range(offset, offset+n_input) ]
```

### Listing 5. Symbols to vector of int as input

The training label is a one-hot vector coming from the symbol after the 3 input symbols.

```
symbols_out_onehot = np.zeros([vocab_size], dtype=float)
symbols_out_onehot[dictionary[str(training_data[offset+n_input])]] =
1.0
```

## Listing 6. One-hot vector as label

After reshaping to fit in the feed dictionary, the optimization runs:

```
_, acc, loss, onehot_pred = session.run([optimizer, accuracy, cost,
pred], feed_dict={x: symbols_in_keys, y: symbols_out_onehot})
```

## Listing 7. Training step optimization

The accuracy and loss are accumulated to monitor the progress of the training. 50,000 iteration is generally enough to achieve an acceptable accuracy.

```
...
Iter= 49000, Average Loss= 0.528684, Average Accuracy= 88.50%
['could', 'easily', 'retire'] - [while] vs [while]
Iter= 50000, Average Loss= 0.415811, Average Accuracy= 91.20%
['this', 'means', 'we'] - [should] vs [should]
```

## Listing 8. Sample prediction and accuracy data per training subsession (1000 steps)

The cost is a cross entropy between label and softmax() prediction optimized using RMSProp at a learning rate of 0.001. RMSProp performs generally better than Adam and SGD for this case.

```
pred = RNN(x, weights, biases)

# Loss and optimizer
cost =
tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=pred,
labels=y))
optimizer =
tf.train.RMSPropOptimizer(learning_rate=learning_rate).minimize(cost
)
```

## Listing 9. Loss and optimizer

The accuracy of the LSTM can be improved by additional layers.

```
rnn_cell =
rnn.MultiRNNCell([rnn.BasicLSTMCell(n_hidden),rnn.BasicLSTMCell(n_hidden)])
```

### Listing 10. Improved LSTM

Now, the fun part. Let us generate a story by feeding back the predicted output as next symbol in the inputs. The input for this sample output is “had a general” and it predicted the correct output “council”. The “council” is fed back as part of the new input “a general council” to predict a new output “to”, and so on. Surprisingly, LSTM creates a story that somehow makes sense.

```
had a general council to consider what measures they could take to
outwit their common enemy , the cat . some said this , and some said
that but at last a young mouse got
```

### Listing 11. Sample story generated story. Truncated to max of 32 predictions.

If we feed another sequence (eg “mouse”, “mouse”, “mouse”) but not necessarily a sequence found in the story, another narrative is automatically created.

```
mouse mouse mouse , neighbourhood and could receive a outwit always
the neck of the cat . some said this , and some said that but at
last a young mouse got up and said
```

### Listing 12. Inputs with a sequence not found in the story.

The actual sample code can be found [here](#). The sample text file is [here](#).

### Final notes:

1. Using int to encode symbols is easy but the “meaning” of the word is lost. Symbol to int is used to simplify the discussion on building a LSTM application using Tensorflow. Word2Vec is a more optimal way of encoding symbols to vector.
2. One-hot vector representation of output is inefficient especially if we have a realistic vocabulary size. Oxford dictionary has over 170,000 words. The example

above has 112. Again, this is only for simplifying the discussion.

3. The code used here is inspired by [Tensorflow-Examples](#).
4. The number of inputs in this example is 3, see what happens when you use other numbers (eg 4, 5 or more).
5. Running the code each time may generate different results and predictive capabilities since the accuracy is dependent on the initial random values of the parameters. Better accuracy could be achieved at higher training steps (over 150,000). Expect a different dictionary as well per run.
6. Tensorboard is useful in debugging especially in figuring out if the graph is correctly built by the code.
7. Try using another story especially using a different language.

[Machine Learning](#)[Deep Learning](#)[TensorFlow](#)[Lstm](#)[Towards Data Science](#)[Follow](#)

## Written by Rowel Atienza

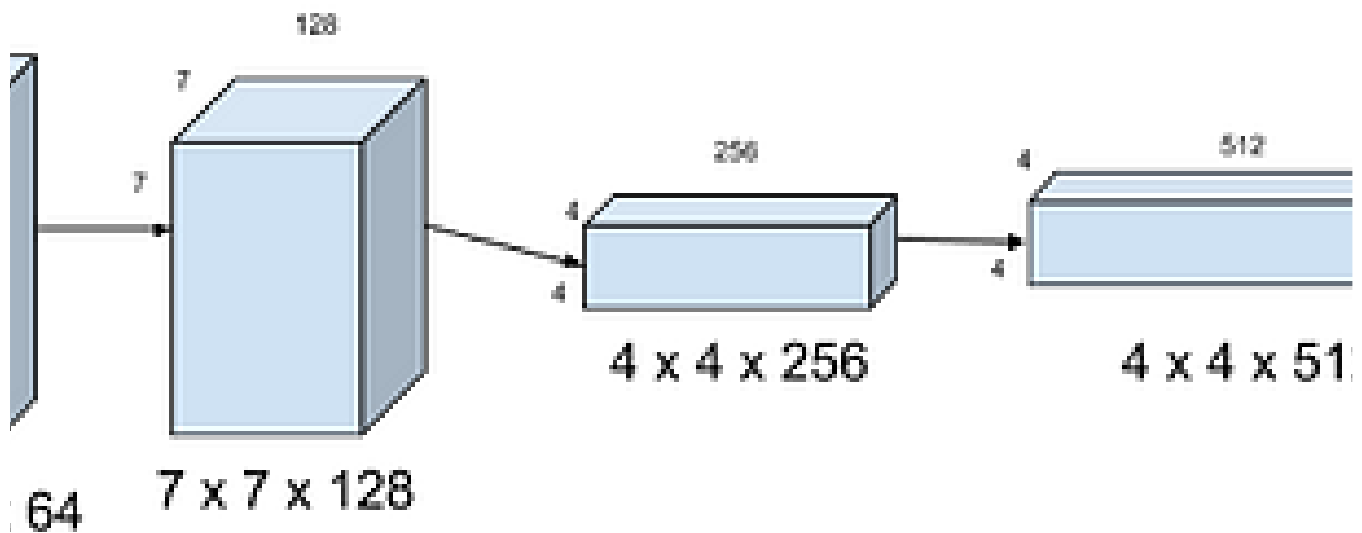
953 Followers · Writer for Towards Data Science

Professor & Scientist, University of the Philippines. PhD Robotics, The Australian National University. Author of Advanced Deep Learning with Keras with TF2.

---

### More from Rowel Atienza and Towards Data Science





Rowel Atienza in Towards Data Science

## GAN by Example using Keras on Tensorflow Backend

Generative Adversarial Networks (GAN) is one of the most promising recent developments in Deep Learning. GAN, introduced by Ian Goodfellow...

6 min read · Mar 30, 2017



3.7K



49



Tim Sumner in Towards Data Science

## A New Coefficient of Correlation

What if you were told there exists a new way to measure the relationship between two variables just like correlation except possibly...

10 min read · Mar 31, 2024



3.4K



43



Youness Mansar in Towards Data Science

## Meet the NiceGUI: Your Soon-to-be Favorite Python UI Library

Build custom web apps easily and quickly

8 min read · Apr 17, 2024



1.2K



5



Users

Front-End

Back-End

 Rowel Atienza

## Gradio & Hugging Face for Rapid Deep Learning App Development

A useful deep learning model must be appreciated by a wider community of potential end users. Most of the time, GitHub, Jupyter Notebook or...

3 min read · Mar 14, 2022

 151 [See all from Rowel Atienza](#)[See all from Towards Data Science](#)

### Recommended from Medium





Michael Keith in Towards Data Science

## Five Practical Applications of the LSTM Model for Time Series, with Code

How to implement an advanced neural network model in several different time series contexts

11 min read · Sep 22, 2023



208



1



Iqra Muhammad

## Time Series Forecasting using LSTM: An Introduction with Code Explanations

Time series forecasting plays an important role in stock market forecasting, finance, and weather prediction. Long Short-Term Memory (LSTM)...

3 min read · Nov 15, 2023

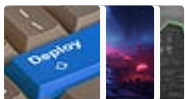


## Lists



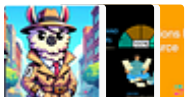
### Practical Guides to Machine Learning

10 stories · 1396 saves



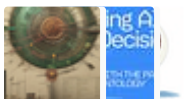
### Predictive Modeling w/ Python

20 stories · 1155 saves



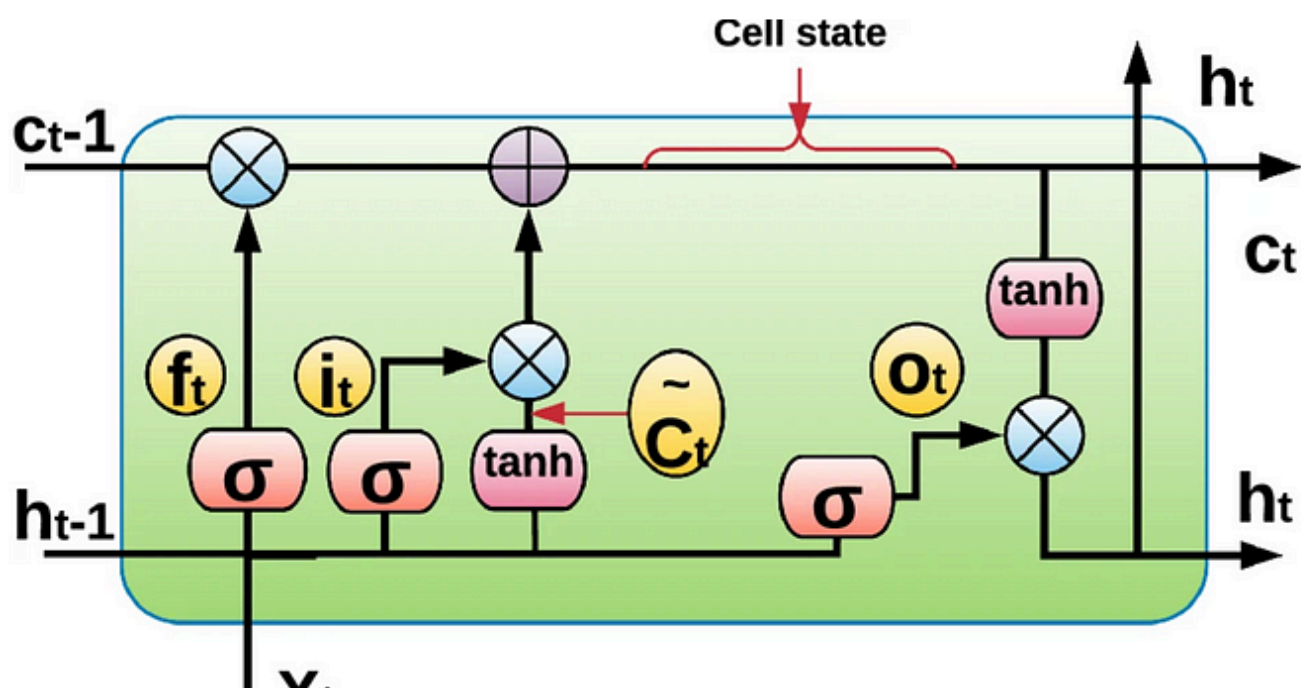
### Natural Language Processing

1431 stories · 926 saves



### data science and AI

40 stories · 146 saves



Rebeen Hamad

## What is LSTM? Introduction to Long Short-Term Memory

Long Short-Term Memory (LSTM) is a type of artificial recurrent neural network (RNN) architecture used in the field of deep learning...

6 min read · Dec 3, 2023



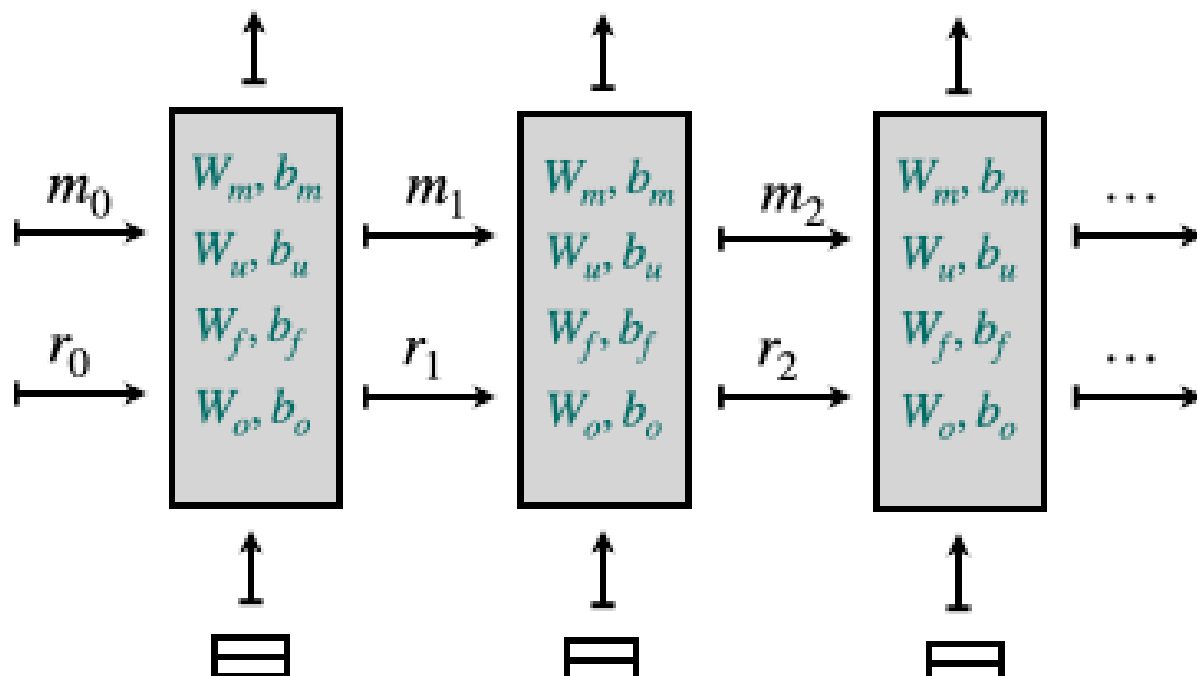
Fahadp

## Cracking the Code: Machine Learning for Next Word Predictions

Let Me Predict What Comes Next in Your Text!

10 min read · Dec 15, 2023





 Ellie Arbab

## RNN, LSTM and GRU Networks

with an application in NLP text generation.

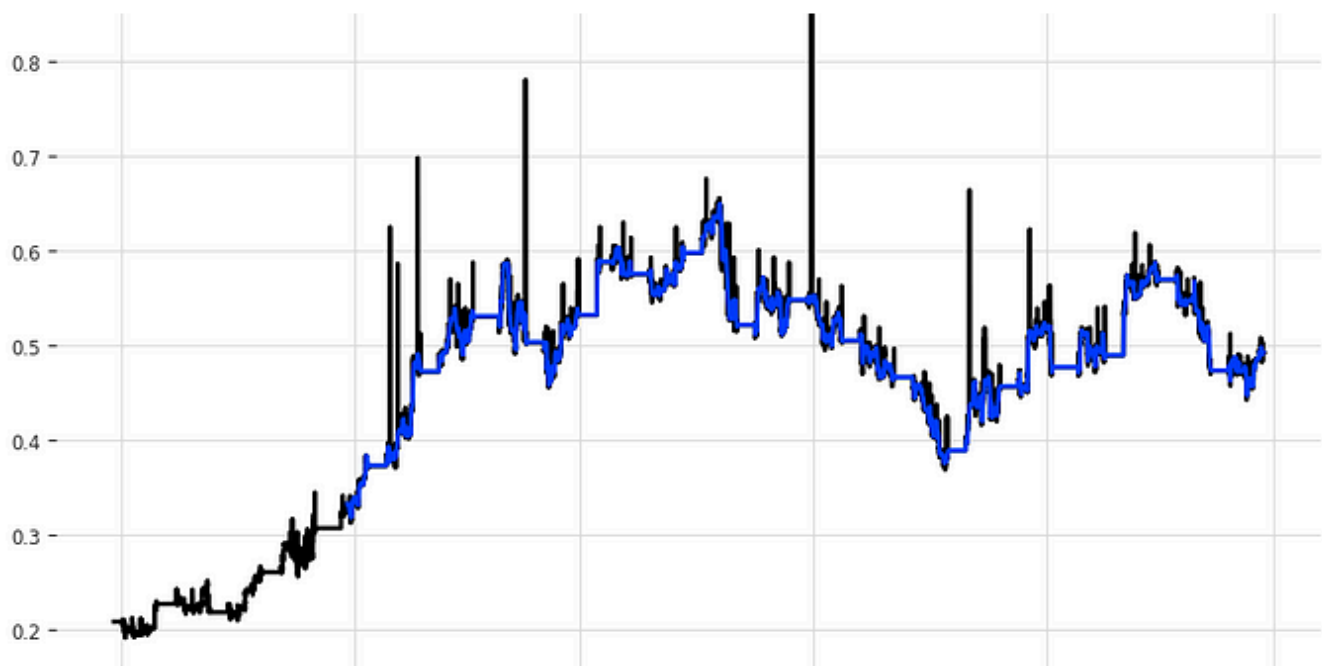
6 min read · Nov 30, 2023



34



1



 John You

## Stock price prediction using Temporal Convolutional Networks

A short tutorial using the darts Python library

3 min read · Nov 9, 2023



15



1



See more recommendations