Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras

by **Jason Brownlee** on July 26, 2016 in **Deep Learning**

Sequence classification is a predictive modeling problem where you have some sequence of inputs over space or time and the task is to predict a category for the sequence.

What makes this problem difficult is that the sequences can vary in length, be comprised of a very large vocabulary of input symbols and may require the model to learn the long-term context or dependencies between symbols in the input sequence.

In this post, you will discover how you can develop LSTM recurrent neural network models for sequence classification problems in Python using the Keras deep learning library.

After reading this post you will know:

- How to develop an LSTM model for a sequence classification problem.
- How to reduce overfitting in your LSTM models through the use of dropout.
- How to combine LSTM models with Convolutional Neural Networks that excel at learning spatial relationships.

Let's get started.

- **Update Oct/2016**: Updated examples for Keras 1.1.0 andTensorFlow 0.10.0.
- **Update Mar/2017**: Updated example for Keras 2.0.2, TensorFlow 1.0.1 and Theano 0.9.0.

Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
Photo by photophilde, some rights reserved.

Problem Description

The problem that we will use to demonstrate sequence learning in this tutorial is the IMDB movie review sentiment classification problem. Each movie review is a variable sequence of words and the sentiment of each movie review must be classified.

The Large Movie Review Dataset (often referred to as the IMDB dataset) contains 25,000 highly-polar movie reviews (good or bad) for training and the same amount again for testing. The problem is to determine whether a given movie review has a positive or negative sentiment.

The data was collected by Stanford researchers and was used in a 2011 paper where a split of 50-50 of the data was used for training and test. An accuracy of 88.89% was achieved.

Keras provides access to the IMDB dataset built-in. The **imdb.load_data()** function allows you to load the dataset in a format that is ready for use in neural network and deep learning models.

The words have been replaced by integers that indicate the ordered frequency of each word in the dataset. The sentences in each review are therefore comprised of a sequence of integers.

Word Embedding

We will map each movie review into a real vector domain, a popular technique when working with text called word embedding. This is a technique where words are encoded as real-valued vectors in a high dimensional space, where the similarity between words in terms of meaning translates to closeness in the vector space.

Keras provides a convenient way to convert positive integer representations of words into a word embedding by an Embedding layer.

We will map each word onto a 32 length real valued vector. We will also limit the total number of words that we are interested in modeling to the 5000 most frequent words, and zero out the rest. Finally, the sequence length (number of words) in each review varies, so we will constrain each review to be 500 words, truncating long reviews and pad the shorter reviews with zero values.

Now that we have defined our problem and how the data will be prepared and modeled, we are ready to develop an LSTM model to classify the sentiment of movie reviews.

Simple LSTM for Sequence Classification

We can quickly develop a small LSTM for the IMDB problem and achieve good accuracy.

Let's start off by importing the classes and functions required for this model and initializing the random number generator to a constant value to ensure we can easily reproduce the results.

```
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
```

We need to load the IMDB dataset. We are constraining the dataset to the top 5,000 words. We also split the dataset into train (50%) and test (50%) sets.

```
# load the dataset but only keep the top n words, zero the rest
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
```

Next, we need to truncate and pad the input sequences so that they are all the same length for modeling. The model will learn the zero values carry no information so indeed the sequences are not the same length in terms of content, but same length vectors is required to perform the computation in Keras.

```
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
```

We can now define, compile and fit our LSTM model.

The first layer is the Embedded layer that uses 32 length vectors to represent each word. The next layer is the LSTM layer with 100 memory units (smart neurons). Finally, because this is a classification problem we use a Dense output layer with a single neuron and a sigmoid activation function to make 0 or 1 predictions for the two classes (good and bad) in the problem.

Because it is a binary classification problem, log loss is used as the loss function (**binary_crossentropy** in Keras). The efficient ADAM optimization algorithm is used. The model is fit for only 2 epochs because it quickly overfits the problem. A large batch size of 64 reviews is used to space out weight updates.

```
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=3, batch_size=64)
```

Once fit, we estimate the performance of the model on unseen reviews.

```
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

For completeness, here is the full code listing for this LSTM network on the IMDB dataset.

```
# LSTM for sequence classification in the IMDB dataset
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
```

```
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
# load the dataset but only keep the top n words, zero the rest
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(nb_words=top_words)
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, nb_epoch=3, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Running this example produces the following output.

Note, if you are using a TensorFlow backend, you may see some warning messages related to "PoolAllocator", that you can ignore for now.

```
Epoch 1/3
16750/16750 [==============================] - 107s - loss: 0.5570 - acc: 0.7149
Epoch 2/3
16750/16750 [==============================] - 107s - loss: 0.3530 - acc: 0.8577
Epoch 3/3
16750/16750 [==============================] - 107s - loss: 0.2559 - acc: 0.9019
Accuracy: 86.79%
```

You can see that this simple LSTM with little tuning achieves near state-of-the-art results on the IMDB problem. Importantly, this is a template that you can use to apply LSTM networks to your own sequence classification problems.

Now, let's look at some extensions of this simple model that you may also want to bring to your own problems.

LSTM For Sequence Classification With Dropout

Recurrent Neural networks like LSTM generally have the problem of overfitting.

Dropout can be applied between layers using the Dropout Keras layer. We can do this easily by adding new Dropout layers between the Embedding and LSTM layers and the LSTM and Dense output layers. For example:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
```

```
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
```

The full code listing example above with the addition of Dropout layers is as follows:

```
# LSTM with Dropout for sequence classification in the IMDB dataset
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
# load the dataset but only keep the top n words, zero the rest
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(Dropout(0.2))
model.add(LSTM(100))
model.add(Dropout(0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, epochs=3, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Running this example provides the following output.

```
Epoch 1/3
16750/16750 [==============================] - 108s - loss: 0.5802 - acc: 0.6898
Epoch 2/3
16750/16750 [==============================] - 108s - loss: 0.4112 - acc: 0.8232
Epoch 3/3
16750/16750 [==============================] - 108s - loss: 0.3825 - acc: 0.8365
Accuracy: 85.56%
```

We can see dropout having the desired impact on training with a slightly slower trend in convergence and in this case a lower final accuracy. The model could probably use a few more epochs of training and may achieve a higher skill (try it an see).

Alternately, dropout can be applied to the input and recurrent connections of the memory units with the LSTM precisely and separately.

Keras provides this capability with parameters on the LSTM layer, the **dropout** for configuring the input dropout and **recurrent_dropout** for configuring the recurrent dropout. For example, we can modify the first example to add dropout to the input and recurrent connections as follows:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
```

The full code listing with more precise LSTM dropout is listed below for completeness.

```
# LSTM with dropout for sequence classification in the IMDB dataset
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
# load the dataset but only keep the top n words, zero the rest
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, epochs=3, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Running this example provides the following output.

```
Epoch 1/3
16750/16750 [==============================] - 112s - loss: 0.6623 - acc: 0.5935
Epoch 2/3
16750/16750 [==============================] - 113s - loss: 0.5159 - acc: 0.7484
Epoch 3/3
16750/16750 [==============================] - 113s - loss: 0.4502 - acc: 0.7981
Accuracy: 82.82%
```

We can see that the LSTM specific dropout has a more pronounced effect on the convergence of the network than the layer-wise dropout. As above, the number of epochs was kept constant and could be increased to see if the skill of the model can be further lifted.

Dropout is a powerful technique for combating overfitting in your LSTM models and it is a good idea to try

both methods, but you may bet better results with the gate-specific dropout provided in Keras.

LSTM and Convolutional Neural Network For Sequence Classification

Convolutional neural networks excel at learning the spatial structure in input data.

The IMDB review data does have a one-dimensional spatial structure in the sequence of words in reviews and the CNN may be able to pick out invariant features for good and bad sentiment. This learned spatial features may then be learned as sequences by an LSTM layer.

We can easily add a one-dimensional CNN and max pooling layers after the Embedding layer which then feed the consolidated features to the LSTM. We can use a smallish set of 32 features with a small filter length of 3. The pooling layer can use the standard length of 2 to halve the feature map size.

For example, we would create the model as follows:

```
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100))
model.add(Dense(1, activation='sigmoid'))
```

The full code listing with a CNN and LSTM layers is listed below for completeness.

```
# LSTM and CNN for sequence classification in the IMDB dataset
import numpy
from keras.datasets import imdb
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.convolutional import Conv1D
from keras.layers.convolutional import MaxPooling1D
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
# fix random seed for reproducibility
numpy.random.seed(7)
# load the dataset but only keep the top n words, zero the rest
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=top_words)
# truncate and pad input sequences
max_review_length = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_review_length)
X_test = sequence.pad_sequences(X_test, maxlen=max_review_length)
# create the model
embedding_vecor_length = 32
model = Sequential()
model.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model.add(Conv1D(filters=32, kernel_size=3, padding='same', activation='relu'))
model.add(MaxPooling1D(pool_size=2))
model.add(LSTM(100))
```

```
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model.summary())
model.fit(X_train, y_train, epochs=3, batch_size=64)
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Running this example provides the following output.

```
Epoch 1/3
16750/16750 [==============================] - 58s - loss: 0.5186 - acc: 0.7263
Epoch 2/3
16750/16750 [==============================] - 58s - loss: 0.2946 - acc: 0.8825
Epoch 3/3
16750/16750 [==============================] - 58s - loss: 0.2291 - acc: 0.9126
Accuracy: 86.36%
```

We can see that we achieve similar results to the first example although with less weights and faster training time.

I would expect that even better results could be achieved if this example was further extended to use dropout.

Resources

Below are some resources if you are interested in diving deeper into sequence prediction or this specific example.

- Theano tutorial for LSTMs applied to the IMDB dataset
- Keras code example for using an LSTM and CNN with LSTM on the IMDB dataset.
- Supervised Sequence Labelling with Recurrent Neural Networks, 2012 book by Alex Graves (and PDF preprint).

Summary

In this post you discovered how to develop LSTM network models for sequence classification predictive modeling problems.

Specifically, you learned:

- How to develop a simple single layer LSTM model for the IMDB movie review sentiment classification problem.
- How to extend your LSTM model with layer-wise and LSTM-specific dropout to reduce overfitting.
- How to combine the spatial structure learning properties of a Convolutional Neural Network with the sequence learning of an LSTM.

Do you have any questions about sequence classification with LSTMs or about this post? Ask your

questions in the comments and I will do my best to answer.

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional developer and a machine learning practitioner. He is dedicated to helping developers get started and get good at applied machine learning. Learn more.

View all posts by Jason Brownlee →

‹ How To Use Classification Machine Learning Algorithms in Weka

How to Use Ensemble Machine Learning Algorithms in Weka ›

135 Responses to *Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras*

Atlant July 29, 2016 at 7:15 pm #

REPLY

It's geat!

Jason Brownlee August 15, 2016 at 12:30 pm #

REPLY

Thanks Atlant.

Sahil July 30, 2016 at 9:34 pm #

REPLY

Hey Jason,

Congrats brother, for continuous great and easy to adapt/understanding lessons. I am just curious to know unsupervised and reinforced neural nets, any tutorials you have?

Regards,
Sahil

Jason Brownlee July 31, 2016 at 7:09 am #

REPLY

Thanks Sahil.

Sorry, no tutorials on unsupervised learning or reinforcement learning with neural nets just yet. Soon though.

Søren Pallesen August 1, 2016 at 1:43 am #

REPLY

Hi, great stuff you are publishing here thanks.

Would this network architecture work for predicting profitability of a stock based time series data of the stock price.

For example with data samples of daily stock prices and trading volumes with 5 minute intervals from 9.30am to 1pm paired with YES or NO to the stockprice increasing by more than 0.5% the rest of the trading day?

Each trading day is one sample and th3 entire data set woule for example the last 1000 trading days.

If this network architecture is not suitable what other would you suggest testing our?

Again thanks for this super resdource.

Jason Brownlee August 1, 2016 at 6:24 am #

REPLY

Thanks Søren.

Sure, it would be worth trying, but I am not an expert on the stock market.

Naufal August 12, 2016 at 2:50 pm #

REPLY

So, the end result of this tutorial is a model. Could you give me an example how to use this model to predict a new review, especially using new vocabularies that don't present in training data? Many thanks..

Jason Brownlee August 15, 2016 at 12:31 pm #

REPLY

I don't have an example Naufal, but the new example would have to encode words using the same integers and embed the integers into the same word mapping.

Faraaz Mohammed March 13, 2017 at 7:21 pm #

REPLY

Thanks Jason for excellent article.
to predict i did below things, please correct i am did wrong. you said to embed..i didnt get that. how to do that.

text = numpy.array(['this is excellent sentence'])
#print(text.shape)
tk = keras.preprocessing.text.Tokenizer(nb_words=2000, lower=True,split=" ")
tk.fit_on_texts(text)
prediction = model.predict(numpy.array(tk.texts_to_sequences(text)))
print(prediction)

Faraaz Mohammed March 13, 2017 at 9:43 pm #

REPLY

Thanks Jason for excellent article.
to predict i did below things, please correct i am did wrong. you said to embed..i didnt get that. how to do that.

text = numpy.array(['this is excellent sentence'])
#print(text.shape)
tk = keras.preprocessing.text.Tokenizer(nb_words=2000, lower=True,split=" ")

```
tk.fit_on_texts(text)
prediction =
model.predict(sequence.pad_sequences(tk.texts_to_sequences(text),maxlen=max_review_lengt
h))
print(prediction)
```

Jason Brownlee March 14, 2017 at 8:14 am #

REPLY

Embed refers to the word embedding layer:
https://keras.io/layers/embeddings/

Joey August 24, 2016 at 6:45 am #

REPLY

Hello Jason! Great tutorials!

When I attempt this tutorial, I get the error message from imdb.load_data :

TypeError: load_data() got an unexpected keyword argument 'test_split'

I tried copying and pasting the entire source code but this line still had the same error.

Can you think of any underlying reason that this is not executing for me?

Jason Brownlee August 24, 2016 at 8:33 am #

REPLY

Sorry to hear that Joey. It looks like a change with Keras v1.0.7.

I get the same error if I run with version 1.0.7. I can see the API doco still refers to the test_split argument here: https://keras.io/datasets/#imdb-movie-reviews-sentiment-classification

I can see that the argument was removed from the function here:
https://github.com/fchollet/keras/blob/master/keras/datasets/imdb.py

Option 1) You can remove the argument from the function to use the default test 50/50 split.

Option 2) You can downgrade Keras to version 1.0.6:

```
sudo pip install --upgrade --no-deps keras==1.0.6
```

Remember you can check your Keras version on the command line with:

```
python -c "import keras; print keras.__version__"
```

I will look at updating the example to be compatible with the latest Keras.

Joey August 25, 2016 at 4:27 am #

REPLY

I got it working! Thanks so much for all of the help Jason!

Jason Brownlee August 25, 2016 at 5:07 am #

REPLY

Glad to hear it Joey.

Jason Brownlee October 7, 2016 at 2:22 pm #

I have updated the examples in the post to match Keras 1.1.0 and TensorFlow 0.10.0.

Chong Wang August 29, 2016 at 11:13 am #

REPLY

Hi, Jason.

A quick question:
Based on my understanding, padding zero in front is like labeling ‘START’. Otherwise it is like labeling ‘END’.
How should I decide ‘pre’ padding or ‘post’ padding? Does it matter?

Thanks.

Jason Brownlee August 30, 2016 at 8:24 am #

REPLY

I don’t think I understand the question, sorry Chong.

Consider trying both padding approaches on your problem and see what works best.

Chong Wang October 6, 2016 at 7:49 am #

REPLY

Hi, Jason.

Thanks for your reply.

I have another quick question in section “LSTM For Sequence Classification With Dropout”.

model.add(Embedding(top_words, embedding_vector_length, input_length=max_review_length, dropout=0.2))
model.add(Dropout(0.2))
…

Here I see two dropout layers. The second one is easy to understand: For each time step, It just randomly deactivates 20% numbers in the output embedding vector.

The first one confuses me: Does it do dropout on the input? For each time step, the input of the embedding layers should be only one index of the top words. In other words, the input is one single number. How can we dropout it? (Or do you mean drop the input indices of 20% time steps?)

Jason Brownlee October 6, 2016 at 9:50 am #

REPLY

Great question, I believe it drops out weights from the input nodes from the embedded layer to the hidden layer.

You can learn more about dropout here:
http://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

Kuow January 12, 2017 at 4:10 pm #

Can the dropout applied in the Embedding layer be thought of as randomly removing a word in a sentence and forcing the classification not to rely on any word?

Jason Brownlee January 13, 2017 at 9:09 am #

I don’t see why not – off the cuff.

Harish August 30, 2016 at 8:10 pm #

REPLY

Hi Jason

Thanks for providing such easy explanations for these complex topics.

In this tutorial, Embedding layer is used as the input layer as the data is a sequence of words.

I am working on a problem where I have a sequence of images as an example and a particular label is assigned to each example. The number of images in the sequence will vary from example to example. I have the following questions:

1) Can I use a LSTM layer as an input layer?

2) If the input layer is a LSTM layer, is there still a need to specify the max_len (which is constraint mentioning the maximum number of images an example can have)

Thanks in advance.

Jason Brownlee August 31, 2016 at 9:28 am #

REPLY

Interesting problem Harish.

I would caution you to consider a suite of different ways of representing this problem, then try a few to see what works.

My gut suggests using CNNs on the front end for the image data and then an LSTM in the middle and some dense layers on the backend for transforming the representation into a prediction.

I hope that helps.

Harish August 31, 2016 at 3:31 pm #

REPLY

Thanks you very much Jason.

Can you please let me know how to deal with sequences of different length without padding in this problem. If padding is required, how to choose the max. length for padding the sequence of images.

Jason Brownlee September 1, 2016 at 7:56 am #

REPLY

Padding is required for sequences of variable length.

Choose a max length based on all the data you have available to evaluate.

Harish September 1, 2016 at 5:12 pm #

Thank you for your time and suggestion Jason.

Can you please explain what masking the input layer means and how can it be used to handle padding in keras.

Sreekar Reddy September 5, 2016 at 10:37 pm #

REPLY

Hi Harish,

I am working on a similar problem and would like to know if you continued on this problem? What worked and what did not?

Thanks in advance

Gciniwe September 1, 2016 at 6:26 am #

REPLY

Hi Jason,

Thanks for this tutorial. It's so helpful! I would like to adapt this to my own problem. I'm working on a problem where I have a sequence of acoustic samples. The sequences vary in length, and I know the identity of the individual/entity producing the signal in each sequence. Since these sequences have a temporal element to them, (each sequence is a series in time and sequences belonging to the same individual are also linked temporally), I thought LSTM would be the way to go.
According to my understanding, the Embedding layer in this tutorial works to add an extra dimension to the dataset since the LSTM layer takes in 3D input data.

My question is is it advisable to use LSTM layer as a first layer in my problem, seeing that Embedding wouldn't work with my non-integer acoustic samples? I know that in order to use LSTM as my first layer, I have to somehow reshape my data in a meaningful way so that it meets the requirements of the inputs of LSTM layer. I've already padded my sequences so my dataset is currently a 2D tensor. Padding with zeros however was not ideal because some of the original acoustic sample values are zero, representing a zero-pressure level. So I've manually padded using a different number.

I'm planning to use a stack of LSTM layers and a Dense layer at the end of my Sequential model.

P.s. I'm new to Keras. I'd appreciate any advice you can give.

Thank you

Jason Brownlee September 1, 2016 at 8:03 am #

REPLY

I'm glad it was useful Gciniwe.

Great question and hard to answer. I would caution you to review some literature for audio-based applications of LSTMs and CNNs and see what representations were used. The examples I've seen have been (sadly) trivial.

Try LSTM as the first layer, but also experiment with CNN (1D) then LSTM for additional opportunities to pull out structure. Perhaps also try Dense then LSTM. I would use one or more Dense on the output layers.

Good luck, I'm very interested to hear what you come up with.

Harish September 1, 2016 at 4:07 pm #

REPLY

Hi Gciniwe

Its interesting to see that I am also working on a similar problem. I work on speech and image processing. I have a small doubt. Please may I know how did you choose the padding values. Because in images also, we will have zeros and unable to understand how to do padding.

Thanks in advance

nick September 20, 2016 at 2:16 am #

REPLY

When i run the above code , i am getting the following error
:MemoryError: alloc failed
Apply node that caused the error: Alloc(TensorConstant{(1L, 1L, 1L) of 0.0}, TensorConstant{24}, Elemwise{Composite{((i0 * i1) // i2)}}[(0, 0)].0, TensorConstant{280})
Toposort index: 145
Inputs types: [TensorType(float32, (True, True, True)), TensorType(int64, scalar), TensorType(int64, scalar), TensorType(int64, scalar)]
Inputs shapes: [(1L, 1L, 1L), (), (), ()]
Inputs strides: [(4L, 4L, 4L), (), (), ()]
Inputs values: [array([[[0.]]], dtype=float32), array(24L, dtype=int64), array(-450L, dtype=int64), array(280L, dtype=int64)]
Outputs clients: [[IncSubtensor{Inc;:int64:}(Alloc.0, Subtensor{::int64}.0, Constant{24}), IncSubtensor{InplaceInc;int64::}(Alloc.0, IncSubtensor{Inc;:int64:}.0, Constant{0}), forall_inplace,cpu,grad_of_scan_fn}(TensorConstant{24}, Elemwise{tanh}.0, Subtensor{int64:int64:int64}.0, Alloc.0, Elemwise{Composite{(i0 – sqr(i1))}}.0, Subtensor{int64:int64:int64}.0, Subtensor{int64:int64:int64}.0, any idea why? i am using theano 0.8.2 and keras 1.0.8

Jason Brownlee September 20, 2016 at 8:34 am #

REPLY

I'm sorry to hear that Nick, I've not seen this error.

Perhaps try the Theano backend and see if that makes any difference?

Shristi Baral November 9, 2016 at 9:57 pm #

REPLY

I got the same problem and I have no clue how to solve it..

Deepak October 3, 2016 at 2:41 am #

REPLY

Hi Jason,

I have one question. Can I use RNN LSTM for Time Series Sales Analysis. I have only one input every day sales of last one year. so total data points is around 278 and I want to predict for next 6 months. Will this much data points is sufficient for using RNN techniques.. and also can you please explain what is difference between LSTM and GRU and where to USE LSTM or GRU

Jason Brownlee October 3, 2016 at 5:21 am #

REPLY

Hi Deepak, My advice would be to try LSTM on your problem and see.

You may be better served using simpler statistical methods to forecast 60 months of sales data.

Corne Prinsloo October 13, 2016 at 5:59 pm #

REPLY

Jason, this is great. Thanks!

I would also love to see some unsupervised learning to know how it works and what the applications are.

Jason Brownlee October 14, 2016 at 8:59 am #

REPLY

Hi Corne,

I tend not to write tutorials on unsupervised techniques (other than feature selection) as I do not find methods like clustering useful in practice on predictive modeling problems.

Jeff Wu October 14, 2016 at 5:49 am #

REPLY

Thanks for writing this tutorial. It's very helpful. Why do LSTMs not require normalization of their features' values?

Jason Brownlee October 14, 2016 at 9:09 am #

REPLY

Hi Jeff, great question.

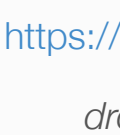
Often you can get better performance with neural networks when the data is scaled to the range of the transfer function. In this case we use a sigmoid within the LSTMs so we find we get better performance by normalizing input data to the range 0-1.

I hope that helps.

Lau MingFei October 19, 2016 at 10:21 pm #

REPLY

Hi, Jason! Your tutorial is very helpful. But I still have a question about using dropouts in the LSTM cells. What is the difference of the actual effects of droupout_W and dropout_U? Should I just set them the same value in most cases? Could you recommend any paper related to this topic? Thank you very much!



Jason Brownlee October 20, 2016 at 8:38 am #

REPLY

I would refer you to the API Lau:

https://keras.io/layers/recurrent/#lstm

> *dropout_W: float between 0 and 1. Fraction of the input units to drop for input gates.*
> *dropout_U: float between 0 and 1. Fraction of the input units to drop for recurrent connections.*

Generally, I recommend testing different values and see what works. In practice setting them to the same values might be a good starting point.

Jeff October 24, 2016 at 10:16 pm #

REPLY

Hello,
thanks for the nice article. I have a question about the data encoding: “The words have been replaced by integers that indicate the ordered frequency of each word in the dataset”.

What exactly does ordered frequency mean? For instance, is the most frequent word encoded as 0 or 4999 in the end?

Jason Brownlee October 25, 2016 at 8:23 am #

REPLY

Great question Jeff.

I believe the most frequent word is 1.

I believe 0 was left for use as padding or when we want to trip low frequency words.

Mazen October 25, 2016 at 12:27 am #

REPLY

Thank you for your very useful posts.
I have a question.
In the last example (CNN&LSTM), It's clear that we gained a faster training time, but how can we know that CNN is suitable here for this problem as a prior layer to LSTM. What does the spatial structure here mean?
So, If I understand how to decide whether a dataset X has a spatial structure, then will this be a suitable clue to suggest a prior CNN to LSTM layer in a sequence-based problem?

Thanks,
Mazen

Jason Brownlee October 25, 2016 at 8:28 am #

REPLY

Hi Mazen,

The spatial structure is the order of words. To the CNN, they are just a sequence of numbers, but we know that that sequence has structure – the words (numbers used to represent words) and their order matter.

Model selection is hard. Often you want to pick the model that has the mix of the best performance and lowest complexity (easy to understand, maintain, retrain, use in production).

Yes, if a problem has some spatial structure (image, text, etc.) try a method that preserves that structure, like a CNN.

Eduardo November 8, 2016 at 3:31 am #

REPLY

Hi Jason, great post!

I have been trying to use your experiment to classify text that come from several blogs for gender classification. However, I am getting a low accuracy close to 50%. Do you have any suggestions in terms of how I could pre-process my data to fit in the model? Each blog text has approximately 6000 words and i am doing some research know to see what I can do in terms of pre-processing to apply to your model.

Thanks

Jason Brownlee November 8, 2016 at 9:57 am #

REPLY

Wow, cool project Eduardo.

I wonder if you can cut the problem back to just the first sentence or first paragraph of the post.

I wonder if you can use a good word embedding.

I also wonder if you can use a CNN instead od LSTM to make the classification – or at least compare CNN alone to CNN + LSTM and double done on what works best.

Generally, here is a ton of advice for improving performance on deep learning problems:
http://machinelearningmastery.com/improve-deep-learning-performance/

Emma November 11, 2016 at 4:24 pm #

REPLY

Hi Jason,

Thank you for your time for this very helpful tutorial.
I was wondering if you would have considered to randomly shuffle the data prior to each epoch of training?

Thanks

Jason Brownlee November 12, 2016 at 7:18 am #

REPLY

Hi Emma,

Great question. The data is automatically shuffled prior to each epoch by the fit() function.

See more about the shuffle argument to the fit() function here:
https://keras.io/models/sequential/

Shashank November 11, 2016 at 4:51 pm #

REPLY

Hi Jason,

Can you please show how to convert all the words to integers so that they are ready to be feed into keras models?

Here in IMDB they are directly working on integers but I have a problem where I have got many rows of text and I have to classify them(multiclass problem).

Also in LSTM+CNN i am getting an error:

ERROR (theano.gof.opt): Optimization failure due to: local_abstractconv_check
ERROR (theano.gof.opt): node: AbstractConv2d{border_mode='half', subsample=(1, 1), filter_flip=True, imshp=(None, None, None, None), kshp=(None, None, None, None)}(DimShuffle{0,2,1,x}.0, DimShuffle{3,2,0,1}.0)
ERROR (theano.gof.opt): TRACEBACK:
ERROR (theano.gof.opt): Traceback (most recent call last):
File "C:\Anaconda2\lib\site-packages\theano\gof\opt.py", line 1772, in process_node
replacements = lopt.transform(node)
File "C:\Anaconda2\lib\site-packages\theano\tensor\nnet\opt.py", line 402, in local_abstractconv_check
node.op.__class__.__name__)
AssertionError: AbstractConv2d Theano optimization failed: there is no implementation available supporting the requested options. Did you exclude both "conv_dnn" and "conv_gemm" from the optimizer? If on GPU, is cuDNN available and does the GPU support it? If on CPU, do you have a BLAS library installed Theano can link against?

I am running keras in windows with Theano backend and CPU only.

Thanks

Thang Le November 14, 2016 at 4:16 am #

REPLY

Hi Jason,

Can you tell me how the IMDB database contains its data please? Text or vector?

Thanks.

Jason Brownlee November 14, 2016 at 7:45 am #

REPLY

Hi Thang Le, the IMDB dataset was originally text.

The words were converted to integers (one int for each word), and we model the data as fixed-length vectors of integers. Because we work with fixed-length vectors, we must truncate and/or pad the data to this fixed length.

Le Thang November 14, 2016 at 2:03 pm #

REPLY

Thank you Jason!

So when we call (X_train, y_train), (X_test, y_test) = imdb.load_data(), X_train[i] will be vector. And if it

is vector then how can I convert my text data to vector to use in this?

Jason Brownlee November 15, 2016 at 7:40 am #

REPLY

Hi Le Thang, great question.

You can convert each character to an integer. Then each input will be a vector of integers. You can then use an Embedding layer to convert your vectors of integers to real-valued vectors in a projected space.

Quan Xiu November 14, 2016 at 6:36 pm #

REPLY

Hi Jason,

As I understand, X_train is a variable sequence of words in movie review for input then what does Y_train stand for?

Thank you!

Jason Brownlee November 15, 2016 at 7:53 am #

REPLY

Hi Quan Xiu, Y is the output variables and Y_train are the output variables for the training dataset.

For this dataset, the output values are movie sentiment values (positive or negative sentiment).

Quan Xiu November 15, 2016 at 2:38 pm #

REPLY

Thank you Jason,

So when we take X_test as input, the output will be compared to y_test to compute the accuracy, right?

Jason Brownlee November 16, 2016 at 9:24 am #

REPLY

Yes Quan Xiu, the predictions made by the model are compared to y_test.

Herbert Kruitbosch November 22, 2016 at 7:47 pm #

REPLY

The performance of this LSTM-network is lower than TFIDF + Logistic Regression:

https://gist.github.com/prinsherbert/92313f15fc814d6eed1e36ab4df1f92d

Are you sure the hidden state's aren't just counting words in a very expensive manor?

Jason Brownlee November 23, 2016 at 8:55 am #

REPLY

It's true that this example is not tuned for optimal performance Herbert.

Herbert Kruitbosch November 23, 2016 at 8:57 pm #

REPLY

This leaves a rather important question, does it actually learn more complicated features than word-counts? And do LSTM's do so in general? Obviously there is literature out there on this topic, but I think your post is somewhat misleading w.r.t. power of LSTM's. It would be great to see an example where an LSTM outperforms a TFIDF, and give an idea about the type and size of the data that you need. (Thank you for the quick reply though 🙂)

LSTM's are only neat if they actually remember contextual things, not if they just fit simple models and take a long time to do so.

Jason Brownlee November 24, 2016 at 10:39 am #

REPLY

I agree Herbert.

LSTMs are hard to use. Initially, I wanted to share how to get up and running with the technique. I aim to come back to this example and test new configurations to get more/most from the method.

Herbert Kruitbosch December 8, 2016 at 12:29 am #

That would be great! It would also be nice to get an idea about the size of data needed for good performance (and of course, there are thousands of other open questions :))

Huy Huynh November 23, 2016 at 4:08 am #

REPLY

Many thank your post, Jason. It's helpful

I have some short questions. First, I feel nervous when chose hyperparameter for the model such as length vectors (32), a number of Embedding unit (500), a number of LSTM unit(100), most frequent words(5000). It depends on dataset, doesn't it? How can we choose parameter?

Second, I have dataset about news daily for predicting the movement of price stock market. But, each news seems more words than each comment imdb dataset. Average each news about 2000 words, can you recommend me how I can choose approximate hyperparameter.

Thank you, (P/s sorry about my English if have any mistake)

Jason Brownlee November 23, 2016 at 9:03 am #

REPLY

Hi Huy,

We have to choose something. It is good practice to grid search over each of these parameters and select for best performance and model robustness.

Perhaps you can work with the top n most common words only.

Perhaps you can use a projection or embedding of the article.

Perhaps you can use some classical NLP methods on the text first.

Huy Huynh November 24, 2016 at 3:47 am #

REPLY

Thank you for your quick response,

I am a newbie in Deep Learning, It seems really difficult to choose relevant parameters.

Huy Huynh November 23, 2016 at 4:16 am #

REPLY

According to my understanding, When training, the number of epoch often more than 100 to evaluate supervised machine learning result. But, In your example or Keras sample, It's only between 3-15 epochs. Can you explain about that?

Thanks,

Jason Brownlee November 23, 2016 at 9:03 am #

REPLY

Epochs can vary from algorithm and problem. There are no rules Huy, let results guide everything.

Huy Huynh November 24, 2016 at 3:49 am #

REPLY

So, How we can choose the relevant number of epochs?

Jason Brownlee November 24, 2016 at 10:41 am #

REPLY

Trial and error on your problem, and carefully watch the learning rate on your training and validation datasets.

Søren Pallesen November 27, 2016 at 8:08 pm #

REPLY

Im looking for benchmarks of LSTM networks on Keras with known/public datasets.

Could you share what hardware configuration the examples in this post was run on (GPU/CPU/RAM etc)?

Thx

Jason Brownlee November 28, 2016 at 8:43 am #

REPLY

I used AWS with the g2.2xlarge configuration.

Mike November 30, 2016 at 11:41 am #

REPLY

Is it possible in Keras to obtain the classifier output as each word propagates through the network?

Jason Brownlee December 1, 2016 at 7:14 am #

REPLY

Hi Mike, you can make one prediction at a time.

Not sure about seeing how the weights propagate through – I have not done this myself with Keras.

lim December 9, 2016 at 4:50 am #

REPLY

Hi,

What are some of the changes you have to make in your binary classification model to work for the multi-label classification?

lim December 9, 2016 at 11:03 am #

REPLY

also instead of a given input data such as imdb in number digit format, what steps do you take to process your raw text format dataset to make it compatible like imdb?

Hossein December 9, 2016 at 9:19 am #

REPLY

Great Job Jason.

I liked it very much…
I would really appreciate it if you tell me how we can do Sequence Clustering with LSTM Recurrent Neural Networks (Unsupervised learning task).

Jason Brownlee December 10, 2016 at 8:01 am #

REPLY

Sorry, I have not used LSTMs for clustering. I don't have good advice for you.

ryan December 10, 2016 at 8:56 pm #

REPLY

Hi Jason,

Your book is really helpful for me. I have a question about time sequence classifier. Let's say, I have 8 classes of time sequence data, each class has 200 training data and 50 validation data, how can I estimate the classification accuracy based on all the 50 validation data per class (sth. like log-maximum likelihood) using scikit-learn package or sth. else? It would be very appreciated that you could give me some advice. Thanks a lot in advance.

Best regards,
Ryan

Jason Brownlee December 11, 2016 at 5:26 am #

REPLY

Hi Ryan, this list of classification measures supported by sklearn might help as a start:
http://scikit-learn.org/stable/modules/classes.html#classification-metrics

Logloss is a very useful measure for evaluating the performance of learning algorithms on multi-class classification problems:
http://scikit-learn.org/stable/modules/generated/sklearn.metrics.log_loss.html#sklearn.metrics.log_loss

I hope that helps as a start.

ryan December 11, 2016 at 8:25 am #

REPLY

Hi Jason, Thank you so much. I will try this logloss.

Jason Brownlee December 12, 2016 at 6:45 am #

REPLY

Let me know how you go.

Shashank December 12, 2016 at 5:09 pm #

REPLY

Hi Jason,

Which approach is better Bags of words or word embedding for converting text to integer for correct and better classification?

I am a little confused in this.

Thanks in advance

Jason Brownlee December 13, 2016 at 8:05 am #

REPLY

Hi Shashank, embeddings are popular at the moment. I would suggest both and see what representation works best for you.

Mango December 19, 2016 at 1:34 am #

REPLY

Hi Jason, thank you for your tutorials, I find them very clear and useful, but I have a little question

when I try to use it to another problem setting..

as is pointed out in your post, words are embedding as vectors, and we feed a sequence of vectors to the model, to do classification.. as you mentioned cnn to deal with the implicit spatial relation inside the word vector(hope I got it right), so I have two questions related to this operation:

1. Is the Embedding layer specific to word, that said, keras has its own vocabulary and similarity definition to treat our feeded word sequence?

2. What if I have a sequence of 2d matrix, something like an image, how should I transform them to meet the required input shape to the CNN layer or directly the LSTM layer? For example, combined with your tutorial for the time series data, I got an trainX of size (5000, 5, 14, 13), where 5000 is the length of my samples, and 5 is the look_back (or time_step), while I have a matrix instead of a single value here, but I think I should use my specific Embedding technique here so I could pass a matrix instead of a vector before an CNN or a LSTM layer….

Sorry if my question is not described well, but my intention is really to get the temporal-spatial connection lie in my data… so I want to feed into my model with a sequence of matrix as one sample.. and the output will be one matrix..

thank you for your patience!!

Banbhrani December 19, 2016 at 7:04 pm #

REPLY

33202176/33213513 [==============================>.] – ETA: 0s 19800064/33213513 [================>……] – ETA: 207s – ETA: 194s

```
_____
Layer (type) Output Shape Param # Connected to
==========================================================================
========================
embedding_1 (Embedding) (None, 500, 32) 160000 embedding_input_1[0][0]
_____
_______
lstm_1 (LSTM) (None, 100) 53200 embedding_1[0][0]
_____
_______
dense_1 (Dense) (None, 1) 101 lstm_1[0][0]
==========================================================================
========================
Total params: 213301
_____
_______
None
```

Epoch 1/3

Kernel died, restarting

Eka January 10, 2017 at 12:49 pm #

REPLY

Hi Jason,

Thanks for the nice article. Because IMDb data is very large I tried to replace it with spam dataset. What kind of changes should I make in the original code to run it. I have asked this question in stack-overflow but sofar no answer. http://stackoverflow.com/questions/41322243/how-to-use-keras-rnn-for-text-classification-in-a-dataset ?

Any help?

Jason Brownlee January 11, 2017 at 9:25 am #

REPLY

Great idea!

I would suggest you encode each word as a unique integer. Then you can start using it as an input for the Embedding layer.

AKSHAY January 11, 2017 at 6:55 am #

REPLY

Hi Jason,

Thanks for the post. It is really helpful. Do I need to configure for the tensorflow to make use of GPU when I run this code or does it automatically select GPU if its available?

Jason Brownlee January 11, 2017 at 9:31 am #

REPLY

These examples are small and run fast on the CPU, no GPU is required.

AKSHAY January 11, 2017 at 12:49 pm #

REPLY

I tried it on CPU and it worked fine. I plan to replicate the process and expand your method for a different use case. Its high dimensional compared to this. Do you have a tutorial on making use of GPU as well? Can I implement the same code in gpu or is the format all different?

Jason Brownlee January 12, 2017 at 9:24 am #

REPLY

Same code, use of the backend is controlled by the Theano or TensorFlow backend that you're using.

Stan January 12, 2017 at 4:12 am #

REPLY

Jason,

Thanks for the interesting tutorial! Do you have any thoughts on how the LSTM trained to classify sequences could then be turned around to generate new ones? I.e. now that it "knows" what a positive review sounds like, could it be used to generate new and novel positive reviews? (ignore possible nefarious uses for such a setup 🙂)

There are several interesting examples of LSTMs being trained to learn sequences to generate new ones… however, they have no concept of classification, or understanding what a "good" vs "bad" sequence is, like yours does. So, I'm essentially interested in merging the two approaches — train an LSTM with a number of "good" and "bad" sequences, and then have it generate new "good" ones.

Any thoughts or pointers would be very welcome!

Jason Brownlee January 12, 2017 at 9:37 am #

REPLY

I have not explored this myself. I don't have any offhand quips, it requires careful thought I think.

This post might help with the other side of the coin, the generation of text:
http://machinelearningmastery.com/text-generation-lstm-recurrent-neural-networks-python-keras/

I would love to hear how you get on.

Stan January 13, 2017 at 1:31 am #

REPLY

Thanks, if you do come up with any crazy ideas, please let me know :).

One pedestrian approach I'm thinking off is having the classifier used to simply "weed out" the undesired inputs, and then feed only desired ones into a new LSTM which can then be used to generate more sequences like those, using the approach like the one in your other post.

That doesn't seem ideal, as it feels like I'm throwing away some of the knowledge about what

makes an undesired sequence undesired… But, on the other hand, I have more freedom in selecting the classifier algorithm.

Albert January 27, 2017 at 9:03 am #

REPLY

Thank you for this tutorial.

Regarding the variable length problem, though other people have asked about it, I have a further question.

If I have a dataset with high deviation of length, say, some text has 10 words, some has 100000 words. Therefore, if I just choose 1000 as my maxlen, I lost a lot of information.

If I choose 100000 as the maxlen, I consume too much computational power.

Is there a another way of dealing with that? (Without padding or truncating)

Also, can you write a tutorial about how to use word2vec pretrained embedding with RNN?

Not word2vec itself, but how to use the result of word2vec.

The counting based word representation lost too much semantic information.

Jason Brownlee January 27, 2017 at 12:28 pm #

REPLY

Great questions Albert.

I don't have a good off-the-cuff answer for you re long sequences. It requires further research.

Keen to tackle the suggested tutorial using word2vc representations.

Charles January 29, 2017 at 4:33 am #

REPLY

I only have biology background, but I can reproduced the results. Great.

Jason Brownlee February 1, 2017 at 10:11 am #

REPLY

Glad to hear it Charles.

Jax February 1, 2017 at 6:27 am #

REPLY

Hi Jason, i noted you mentioned updated examples for Tensorflow 0.10.0. I can only see Keras codes, am i missing something?

Thanks.

Jason Brownlee February 1, 2017 at 10:54 am #

REPLY

Hi Jax,

Keras runs on top of Theano and TensorFlow. One or the other are required to use Keras.

I was leaving a note that the example was tested on an updated version of Keras using an updated version of the TensorFlow backend.

Kakaio February 13, 2017 at 8:30 am #

REPLY

I am not sure I understand how recurrence and sequence work here.
I would expect you'd feed a sequence of one-hot vectors for each review, where each one-hot vector represents one word. This way, you would not need a maximum length for the review (nor padding), and I could see how you'd use recurrence one word at a time.
But I understand you're feeding the whole review in one go, so it looks like e feedforward.
Can you explain that?

Jason Brownlee February 13, 2017 at 9:16 am #

REPLY

Hi Kakaio,

Yes, indeed we are feeding one review at a time. It is the input structured we'd use for a MLP.

Internally, consider the LSTM network as building up state on the sequence of words in the review and from that sequence learning the appropriate sentiment.

Kakaop February 13, 2017 at 9:42 am #

REPLY

how is the LSTM building up state one the sequence of words leveraging recurrence? you're feeding the LSTM all the sequence at the same time, there're no time steps.

Jason Brownlee February 13, 2017 at 9:53 am #

REPLY

Hi Kakaop, quite right. The example does not leverage recurrence.

Sweta March 1, 2017 at 8:15 pm #

REPLY

From this tutorial how can I predict the test values and how to write to a file? Are these predicted values generate in the encoded format?

Bruce Ho March 2, 2017 at 9:29 am #

REPLY

Guys, this is a very clear and useful article, and thanks for the Keras code. But I can't seem to find any sample code for running the trained model to make a prediction. It is not in imdb.py, that just does the evaluation. Does any one have some sample code for prediction to show?

Jason Brownlee March 3, 2017 at 7:39 am #

REPLY

Hi Bruce,

You can fit the model on all of the training data, than forecast for new inputs using:

```
1 y = model.predict(X)
```

Does that help?

Bruce Ho March 3, 2017 at 4:47 pm #

REPLY

That's not the hard part. However, I may have figured out what I need to know. That is take the result returned by model.predict and take the last item in the array as the classifications. Any one disagrees?

JUNETAE KIM March 17, 2017 at 10:40 pm #

REPLY

Hi, it's the awesome tutorial.

I have a question regarding your model.

I am new to RNN, so the question would be stupid.

Inputting word embedding layer is crucial in your setting – sequence classification rather than prediction of the next word??

Jason Brownlee March 18, 2017 at 7:48 am #

REPLY

Generally, a word embedding (or similar projection) is a good representation for NLP problems.

DanielHa March 24, 2017 at 2:41 am #

REPLY

Hi Jason,
great tutorial. Really helped me alot.

I've noticed that in the first part you called fit() on the model with "validation_data=(X_test, y_test)". This isn't in the final code summary. So I wondered if that's just a mistake or if you forgot it later on.

But then again it seems wrong to me to use the test data set for validation. What are your thoughts on this?

Jason Brownlee March 24, 2017 at 8:00 am #

REPLY

The model does not use the test data at this point, it is just evaluated on it. It helps to get an idea of how well the model is doing.

Liam March 24, 2017 at 6:25 pm #

REPLY

What happen if the code uses LSTM with 100 units and sentence length is 200. Does that mean only the first 100 words in the sentence act as inputs, and the last 100 words will be ignored?

Jason Brownlee March 25, 2017 at 7:34 am #

REPLY

No, the number of units in the hidden layer and the length of sequences are different configuration parameters.

You can have 1 unit with 2K sequence length if you like, the model just won't learn it.

I hope that helps.

Danielha March 28, 2017 at 7:29 pm #

REPLY

Hi Jason,
in the last part the LSTM layer returns a sequence, right? And after that the dense layer only takes one parameter. How does the dense layer know that it should take the last parameter? Or does it even take the last parameter?

Jason Brownlee March 29, 2017 at 9:06 am #

REPLY

No, in this case each LSTM unit is not returning a sequence, just a single value.

Prashanth R March 28, 2017 at 9:25 pm #

REPLY

Hi Jason,
Very interesting and useful article. Thank you for writing such useful articles. I have had the privilege of going through your other articles which are very useful.

Just wanted to ask, how do we encode a new test data to make same format as required for the program. There is no dictionary involved i guess for the conversion. So how can we go about for this conversion? For instance, consider a sample sentence “Very interesting article on sequence classification”. What will be encoded numeric representation?
Thanks in advance

Jason Brownlee March 29, 2017 at 9:07 am #

REPLY

Great question.

You can encode the chars as integers (integer encode), then encode the integers as boolean vectors (one hot encode).

trangtruong March 29, 2017 at 7:19 pm #

REPLY

I have dataset just a vector feature like [1, 0,5,1,1,2,1] -> y just 0,1 binary or category like 0,1,2,3. I want to use LSTM to classify binary or category, how can i do it guys, i just add LSTM with Dense, but LSTM need input 3 dimension but Dense just 2 dimension. I know i need time sequence, i try to find out more but can’t get nothing. Can u explain and tell me how. pls, Thank you so much

Jason Brownlee March 30, 2017 at 8:51 am #

REPLY

You may want to consider a seq2seq structure with an encoder for the input sequence and a decoder for the output sequence.

Something like:

```
# encoder
model.add(LSTM(3, ...))
# mapping
model.add(RepeatVector(2))
# decoder
model.add(LSTM(2, return_sequences=True))
model.add(Dense(2))
```

I have a tutorial on this scheduled.

I hope that helps.

trangtruong March 30, 2017 at 5:51 pm #

REPLY

thanks you, i will try to find out, then response you.

Jason Brownlee March 31, 2017 at 5:51 am #

REPLY

You're welcome.

trangtruong March 30, 2017 at 6:17 pm #

REPLY

Ay, i have 1 question in another your post about why i use function evaluate model.evaluate(x_test, y_test) to get accuracy score of model after train with train dataset , but its return result >1 in some case, i don't know why, it make me can't beleive in this function. Can you explain for me why?

Jason Brownlee March 31, 2017 at 5:52 am #

REPLY

Sorry I don't understand your question, perhaps you can rephrase it?

trangtruong March 31, 2017 at 12:34 pm #

I don't know the result return by function evaluate >1, but i thinks it should just from 0 -> 1 (model.evaluate(x_test,y_test) with model i had trained it before with train dataset)

trangtruong March 30, 2017 at 9:07 pm #

REPLY

Hi Jason, Can you explain your code step by step Jason, i have follow tutorial : https://blog.keras.io/building-autoencoders-in-keras.html but i have some confused to understand. :(.

Jason Brownlee March 31, 2017 at 5:54 am #

REPLY

If you have questions about that post, I would recommend contacting the author.

Mazhar Ali April 6, 2017 at 7:36 pm #

REPLY

Hi Dear Joson
I am new to deep learning and intends to work on keras or tensorflow for corpus analysis. May you help me or send me basic tutorials
regards
Mazhar Ali

Jason Brownlee April 9, 2017 at 2:38 pm #

REPLY

Sorry, I only have tutorials for Keras:
http://machinelearningmastery.com/start-here/#deeplearning

Ady April 7, 2017 at 12:52 am #

REPLY

Thank you for your friendly explanation.
I bought a lot of help from your books.

Are you willing to add examples of fit_generator and batch normalization to the IMDB LSTM example?

I was told to use the fit_generator function to process large amounts of data.
If there is an example, it will be very helpful to book buyers.

Jason Brownlee April 9, 2017 at 2:44 pm #

REPLY

I would like to add this kind of example in the future. Thanks for the suggestion.

Fernando López April 8, 2017 at 4:31 am #

REPLY

Hi Jason

I would like to know where I can read more about dropout and recurrent_dropout. Do you know some paper or something to explore it?

Thanks!

Jason Brownlee April 9, 2017 at 2:56 pm #

REPLY

I have a tutorial on dropout here:
http://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/

I have a post on recurrent dropout scheduled for the blog soon.

Leave a Reply

Name (required)

Email (will not be published) (required)

Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery

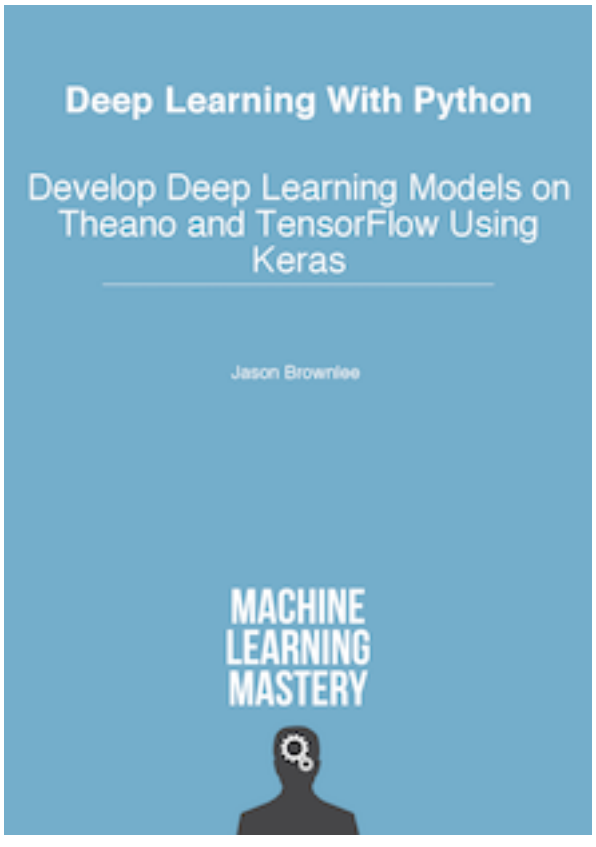
Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More

Finally Get Started With Deep Learning

Sick of the fancy math and need for super computers?
Looking for step-by-step tutorials?
Want end-to-end projects?

Get Started With Deep Learning in Python Today!



POPULAR

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras

JULY 21, 2016

Develop Your First Neural Network in Python With Keras Step-By-Step

MAY 24, 2016

Your First Machine Learning Project in Python Step-By-Step

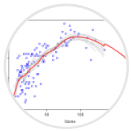
JUNE 10, 2016

How to Run Your First Classifier in Weka

FEBRUARY 17, 2014

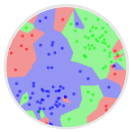
Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras

JULY 26, 2016



A Tour of Machine Learning Algorithms

NOVEMBER 25, 2013



Tutorial To Implement k-Nearest Neighbors in Python From Scratch

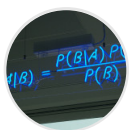
SEPTEMBER 12, 2014

Multi-Class Classification Tutorial with the Keras Deep Learning Library

JUNE 2, 2016

Regression Tutorial with the Keras Deep Learning Library in Python

JUNE 9, 2016



How To Implement Naive Bayes From Scratch in Python

DECEMBER 8, 2014