How to Predict Sentiment from Movie Reviews Using Deep Learning (Text Classification)

by Jason Brownlee on August 7, 2022 in Deep Learning 173

Sentiment analysis is a natural language processing problem where text is understood, and the underlying intent is predicted.

In this post, you will discover how you can predict the sentiment of movie reviews as either positive or negative in Python using the Keras deep learning library.

After reading this post, you will know:

- About the IMDB sentiment analysis problem for natural language processing and how to load it in Keras
- How to use word embedding in Keras for natural language problems
- How to develop and evaluate a multi-layer perception model for the

IMDB problem

- How to develop a one-dimensional convolutional neural network model for the IMDB problem

Kick-start your project with my new book Deep Learning With Python, including *step-by-step tutorials* and the *Python source code* files for all examples.

Let's get started.

- **Jul/2016**: First published
- **Update Oct/2016**: Updated for Keras 1.1.0 and TensorFlow 0.10.0
- **Update Mar/2017**: Updated for Keras 2.0.2, TensorFlow 1.0.1 and Theano 0.9.0
- **Update Jul/2019**: If you are using Keras 2.2.4 and NumPy 1.16.2+ and get "*ValueError: Object arrays cannot be loaded when allow_pickle=False*", then try updating NumPy to 1.16.1, update Keras to the version from github, or use the fix described here
- **Update Sep/2019**: Updated for Keras 2.2.5
- **Update Jul/2022**: Updated for TensorFlow 2.x API

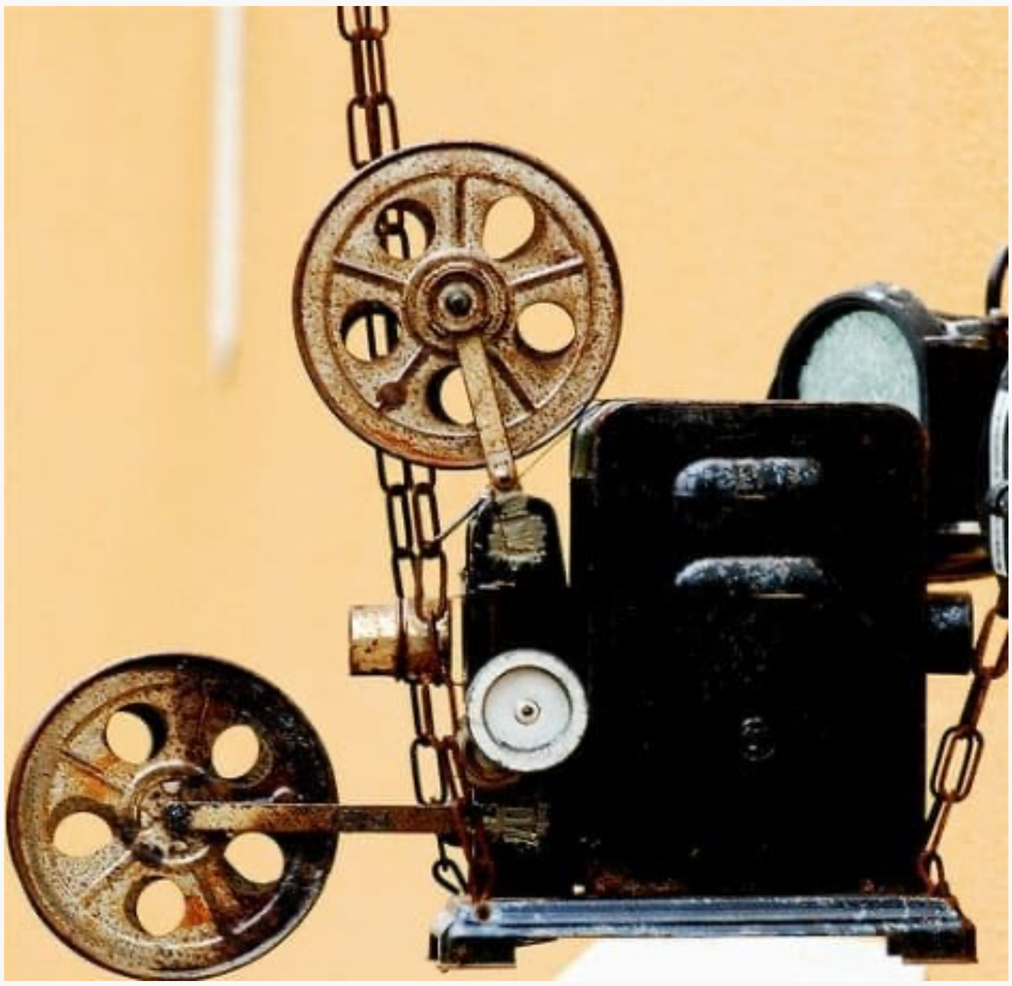
Predict sentiment from movie reviews using deep learning

Photo by SparkCBC, some rights reserved.

IMDB Movie Review Sentiment Problem Description

The dataset is the Large Movie Review Dataset, often referred to as the IMDB dataset.

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 used in a 2011 paper [PDF] where a split of 50/50 of the data was used for training and test. An accuracy of 88.89% was achieved.

The data was also used as the basis for a Kaggle competition titled “Bag of Words Meets Bags of Popcorn” from late 2014 to early 2015. Accuracy was achieved above 97%, with winners achieving 99%.

Load the IMDB Dataset with Keras

Keras provides built-in access to the IMDB dataset.

The keras.datasets.imdb.load_data() allows you to load the dataset in a format that is ready for use in neural networks and deep learning models.

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

Calling imdb.load_data() the first time will download the IMDB dataset to your computer and store it in your home directory under

~/.keras/datasets/imdb.pkl as a 32-megabyte file.

Usefully, the imdb.load_data() provides additional arguments, including the number of top words to load (where words with a lower integer are marked as zero in the returned data), the number of top words to skip (to avoid the repeated use of "the"), and the maximum length of reviews to support.

Let's load the dataset and calculate some properties of it. You will start by loading some libraries and the entire IMDB dataset as a training dataset.

```
import numpy as np
from tensorflow.keras.datasets import imdb
import matplotlib.pyplot as plt
# load the dataset
(X_train, y_train), (X_test, y_test) = imdb.load_data(
X = np.concatenate((X_train, X_test), axis=0)
y = np.concatenate((y_train, y_test), axis=0)
...
```

Next, you can display the shape of the training dataset.

```
...
# summarize size
print("Training data: ")
print(X.shape)
print(y.shape)
```

Running this snippet, you can see that there are 50,000 records.

```
Training data:
(50000,)
(50000,)
```

You can also print the unique class values.

```
...
# Summarize number of classes
print("Classes: ")
print(np.unique(y))
```

You can see it is a binary classification problem for good and bad sentiment in the review.

```
Classes:
[0 1]
```

Next, you can get an idea of the total number of unique words in the dataset.

```
...
# Summarize number of words
print("Number of words: ")
print(len(np.unique(np.hstack(X))))
```

Interestingly, you can see that there are just under 100,000 words across the entire dataset.

```
Number of words:
88585
```

Finally, you can get an idea of the average review length.

```
...
# Summarize review length
print("Review length: ")
result = [len(x) for x in X]
print("Mean %.2f words (%f)" % (np.mean(result), np.std
# plot review length
plt.boxplot(result)
plt.show()
```

You can see that the average review has just under 300 words with a standard deviation of just over 200 words.

```
Review length:
Mean 234.76 words (172.911495)
```

Looking at a box and whisker plot for the review lengths in words, you can see an exponential distribution that you can probably cover the mass of the distribution with a clipped length of 400 to 500 words.

Review length in words for IMDB dataset

Word Embeddings

A recent breakthrough in the field of natural language processing is called word embedding.

This technique is 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.

Discrete words are mapped to vectors of continuous numbers. This is useful when working with natural language problems with neural networks and deep learning models as they require numbers as input.

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

The layer takes arguments that define the mapping, including the maximum number of expected words, also called the vocabulary size (e.g., the largest integer value that will be seen as an integer). The layer also allows you to specify the dimensionality for each word vector, called the output dimension.

You want to use a word embedding representation for the IMDB dataset.

Let's say that you are only interested in the first 5,000 most used words in the dataset. Therefore, your vocabulary size will be 5,000. You can choose to use a 32-dimension vector to represent each word. Finally, you may choose to cap the maximum review length at 500 words, truncating reviews longer than that and padding reviews shorter than that with 0 values.

You will load the IMDB dataset as follows:

```
...
imdb.load_data(nb_words=5000)
```

You will then use the Keras utility to truncate or pad the dataset to a length of 500 for each observation using the sequence.pad_sequences() function.

```
...
```

```
X_train = sequence.pad_sequences(X_train, maxlen=500)
X_test = sequence.pad_sequences(X_test, maxlen=500)
```

Finally, later on, the first layer of your model would be a word embedding layer created using the Embedding class as follows:

```
...
Embedding(5000, 32, input_length=500)
```

The output of this first layer would be a matrix with the size 32×500 for a given review training or test pattern in integer format.

Now that you know how to load the IMDB dataset in Keras and how to use a word embedding representation for it, let's develop and evaluate some models.

Simple Multi-Layer Perceptron Model for the IMDB Dataset

You can start by developing a simple multi-layer perceptron model with a single hidden layer.

The word embedding representation is a true innovation, and you will demonstrate what would have been considered world-class results in 2011 with a relatively simple neural network.

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

```
# MLP for the IMDB problem
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing import sequence

...
```

Next, you will load the IMDB dataset. You will simplify the dataset as discussed during the section on word embeddings—only the top 5,000 words will be loaded.

You will also use a 50/50 split of the dataset into training and test sets. This is a good standard split methodology.

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

You will bound reviews at 500 words, truncating longer reviews and zero-padding shorter ones.

```
...
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_wo
X_test = sequence.pad_sequences(X_test, maxlen=max_word
```

Now, you can create your model. You will use an Embedding layer as the input layer, setting the vocabulary to 5,000, the word vector size to 32 dimensions, and the input_length to 500. The output of this first layer will be a 32×500-sized matrix, as discussed in the previous section.

You will flatten the Embedded layers' output to one dimension, then use one dense hidden layer of 250 units with a rectifier activation function. The output layer has one neuron and will use a sigmoid activation to output values of 0 and 1 as predictions.

The model uses logarithmic loss and is optimized using the efficient ADAM optimization procedure.

```
...
# create the model
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_wor
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='ac
model.summary()
```

You can fit the model and use the test set as validation while training. This model overfits very quickly, so you will use very few training epochs, in this case, just 2.

There is a lot of data, so you will use a batch size of 128. After the model is trained, you will evaluate its accuracy on the test dataset.

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

Tying all of this together, the complete code listing is provided below.

```
# MLP for the IMDB problem
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing import sequence
# load the dataset but only keep the top n words, zero
top_words = 5000
```

```
(X_train, y_train), (X_test, y_test) = imdb.load_data(
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_w
X_test = sequence.pad_sequences(X_test, maxlen=max_wor
# create the model
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_wo
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='a
model.summary()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Running this example fits the model and summarizes the estimated performance.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

You can see that this very simple model achieves a score of 87%, which is in the neighborhood of the original paper, with minimal effort.

```
Epoch 1/2
196/196 - 4s - loss: 0.5579 - accuracy: 0.6664 - val_lo
Epoch 2/2
196/196 - 4s - loss: 0.2108 - accuracy: 0.9165 - val_lo
Accuracy: 87.31%
```

You can likely do better if you trained this network, perhaps using a larger embedding and adding more hidden layers.

Let’s try a different network type.

One-Dimensional Convolutional Neural Network Model for the IMDB Dataset

Convolutional neural networks were designed to honor the spatial structure in image data while being robust to the position and orientation of learned objects in the scene.

This same principle can be used on sequences, such as the one-dimensional sequence of words in a movie review. The same properties that make the CNN model attractive for learning to recognize objects in images can help to learn structure in paragraphs of words, namely the techniques invariance to the specific position of features.

Keras supports one-dimensional convolutions and pooling by the Conv1D and MaxPooling1D classes, respectively.

Again, let’s import the classes and functions needed for this example and initialize your random number generator to a constant value so that you can easily reproduce the results.

```
# CNN for the IMDB problem
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Conv1D
from tensorflow.keras.layers import MaxPooling1D
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing import sequence
```

You can also load and prepare the IMDB dataset as you did before.

```
...
# load the dataset but only keep the top n words, zero
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(n
# pad dataset to a maximum review length in words
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_wo
X_test = sequence.pad_sequences(X_test, maxlen=max_word
```

You can now define your convolutional neural network model. This time, after the Embedding input layer, insert a Conv1D layer. This convolutional layer has 32 feature maps and reads embedded word representations' three vector elements of the word embedding at a time.

The convolutional layer is followed by a 1D max pooling layer with a length and stride of 2 that halves the size of the feature maps from the convolutional layer. The rest of the network is the same as the neural network above.

```
...
# create the model
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_wo
model.add(Conv1D(filters=32, kernel_size=3, padding='s
model.add(MaxPooling1D(pool_size=2))
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='a
model.summary()
```

You will also fit the network the same as before.

```
...
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y_
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
```

```
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Tying all of this together, the complete code listing is provided below.

```
# CNN for the IMDB problem
from tensorflow.keras.datasets import imdb
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Conv1D
from tensorflow.keras.layers import MaxPooling1D
from tensorflow.keras.layers import Embedding
from tensorflow.keras.preprocessing import sequence
# load the dataset but only keep the top n words, zerc
top_words = 5000
(X_train, y_train), (X_test, y_test) = imdb.load_data(
# pad dataset to a maximum review length in words
max_words = 500
X_train = sequence.pad_sequences(X_train, maxlen=max_w
X_test = sequence.pad_sequences(X_test, maxlen=max_wor
# create the model
model = Sequential()
model.add(Embedding(top_words, 32, input_length=max_wo
model.add(Conv1D(32, 3, padding='same', activation='re
model.add(MaxPooling1D())
model.add(Flatten())
model.add(Dense(250, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer='c
model.summary()
# Fit the model
model.fit(X_train, y_train, validation_data=(X_test, y
# Final evaluation of the model
scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

Running the example, you are first presented with a summary of the network structure. You can see your convolutional layer preserves the dimensionality of your Embedding input layer of 32-dimensional input with a maximum of 500 words. The pooling layer compresses this representation by halving it.

Note: Your results may vary given the stochastic nature of the algorithm or evaluation procedure, or differences in numerical precision. Consider running the example a few times and compare the average outcome.

Running the example offers a slight but welcome improvement over the neural network model above with an accuracy of 87%.

```
Epoch 1/2
196/196 - 5s - loss: 0.4661 - accuracy: 0.7467 - val_lo
Epoch 2/2
196/196 - 5s - loss: 0.2195 - accuracy: 0.9144 - val_lo
Accuracy: 87.64%
```

Again, there is a lot of opportunity for further optimization, such as using deeper and/or larger convolutional layers.

One interesting idea is to set the max pooling layer to use an input length of 500. This would compress each feature map to a single 32-length vector and may boost performance.

Summary

In this post, you discovered the IMDB sentiment analysis dataset for natural language processing.

You learned how to develop deep learning models for sentiment analysis, including:

- How to load and review the IMDB dataset within Keras
- How to develop a large neural network model for sentiment analysis

- How to develop a one-dimensional convolutional neural network model for sentiment analysis

Do you have any questions about sentiment analysis or this post? Ask your questions in the comments, and I will do my best to answer.

More On This Topic

How to Prepare Movie Review Data for Sentiment…

How to Develop a Deep Learning Bag-of-Words Model…

Deep Convolutional Neural Network for Sentiment…

Predict Whether a Persons Eyes are Open or Closed…

How to Predict Room Occupancy Based on Environmental Factors

Probabilistic Forecasting Model to Predict Air…

About Jason Brownlee

Jason Brownlee, PhD is a machine learning specialist who teaches developers how to get results with modern machine learning methods via hands-on tutorials.

View all posts by Jason Brownlee →

‹ Object Classification with CNNs Using the Keras Deep Learning Library

How to Normalize and Standardize Your Machine Learning Data in Weka ›

173 Responses to *How to Predict Sentiment from Movie Reviews Using Deep Learning (Text Classification)*

Vishal September 12, 2016 at 1:29 am #

REPLY ↰

imdb.load_data(nb_words=5000, test_split=0.33)

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

The test_split argument doesn't appear to exist in Keras 1.08, perhaps I'm doing something wrong?

Jason Brownlee September 12, 2016 at 8:33 am

REPLY ↰

The API has changed, sorry. I will update the example. You can remove the "test_split" argument.

Jason Brownlee October 7, 2016 at 2:06 pm # REPLY

I have updated the example to match the API changes for Keras 1.1.0 and TensorFlow 0.10.0.

Jason June 29, 2018 at 11:48 pm # REPLY

hi , thanks for your tutorial. but I'm wondering if you have any tutorial about Aspect-based sentiment analysis

Jason Brownlee June 30, 2018 at 6:08 am REPLY

What is aspect-based sentiment analysis?

Joshua July 31, 2018 at 9:55 pm #

A multi-class classification problem where each sentence is associated to an 'aspect'. There are two forms; categorical and term-based. Datasets available:

SemEval-2014

http://alt.qcri.org/semeval2014/task4/index.php?id=data-and-tools

SemEval-2015 http://alt.qcri.org/semeval2015/task12/

Two papers worth reviewing:

Deep Learning for Aspect-Based Sentiment Analysis By Bo

Wang and Min Liu

Aspect-Based Sentiment Analysis Using a Two-Step Neural Network Architecture by S Jebbara and P Cimiano.

There are unsupervised versions of it (term extraction) but categorical is probably more desirable. This dataset would need to be labeled.

Jason Brownlee August 1, 2018 at 7:43 am #

Thanks for sharing.

Joe Williams October 3, 2016 at 12:52 am #

REPLY

Hi, Jason,

Thanks for the great tutorial! How could I modify this to perform sentiment analysis based on user input? Or from a Twitter stream?

Best wishes

Jason Brownlee October 3, 2016 at 5:19 am #

REPLY

You would have to encode the tweets using the same mapping from characters to integers.

I do not have code for you to do this at the moment.

Jie November 9, 2016 at 4:15 pm #

REPLY ↰

1. embeddings are trainable, right? I mean, embeddings are dynamic, and they are changing during the training?

2. How to save the embeddings to a file? We have to load the embeddings to predict new data in the future.

3. I have a file named predict.py. I know how to load the model and graph architecture from here: https://machinelearningmastery.com/save-load-keras-deep-learning-models/
But how to load embeddings in predict.py in order to predict new data?

Jason Brownlee November 10, 2016 at 7:38 am

REPLY ↰

Hi Jie, great questions.

An embedding is a projection, they can be prepared from data. I would not say they are learned, but perhaps you could say that.

I believe they can be prepared deterministically so we would not have to save them to file. I could be wrong about that, but that is my intuition.

Maxim December 19, 2016 at 5:23 am #

REPLY ↰

Jason, many thanks for your lessons! They're amazing!!!!

Maybe I ask you a very stupid question, but I can't understand one thing.

What is the Embedding layer? Could you show an example of it. I mean this is word vector with dimentionality of 500X32. So how it looks like?

[0,0,1,….0,0,0] X 32

.

X500

.

[0,1,0,….0,0,0] X 32

What digits in it? And why if we lift it dimensionality up to 64, the accuracy will rise up?

Thank you!

Martin December 27, 2016 at 7:27 am #

REPLY

test_split = 0.33

This is defined, but not used anywhere in the code. why is that?

Jason Brownlee December 28, 2016 at 7:01 am

REPLY

It is a typo and should be removed, thanks Martin. I will update the example soon.

Anand December 28, 2016 at 7:16 pm #

REPLY

What params would you change typically to achieve what you have mentioned towards the end of the article to improve accuracy?

To quote you: "set the max pooling layer to use an input length of 500. This would compress each feature map to a single 32 length vector"

Could you please help what params (which lines) would need this change?

Jason Brownlee December 29, 2016 at 7:15 am

REPLY

Hi Anand, use some trial and error and tune the method for the problem.

Jim January 13, 2017 at 1:38 am #

REPLY

I am testing the predicted probabilities and values using model.predict(X_test) and model.predict_classes(X_test)

I noticed that the predicted probabilities for the 0 class are all 0.5 w/ median predicted probability of 0.9673.

Am I correct in assuming the model.predict returns probability of the 1 class always, and predicts a class of 0 when that probability is < 50%?

Jason Brownlee January 13, 2017 at 9:14 am #

REPLY

Hi Jim,

I would expect that model.predict() is performing an arg max on the probabilities (selecting the class with the highest probability).

This is knowable if you wade through the code:

https://github.com/fchollet/keras/tree/master/keras

mariya March 25, 2018 at 4:20 am #

REPLY

jim can u sent me the file test that you did ?

brent January 22, 2017 at 4:59 am #

REPLY

When an element in the data is labeled as an integer, let's say 4 for example, could that represent any word that has occurred 4 times in the data or is it a representation of a unique word?

Jason Brownlee January 22, 2017 at 5:13 am #

REPLY

Hi brent, each integer represents a unique word.

GT May 24, 2017 at 7:34 pm #

REPLY

Hi Jason,

Thanks for the great tutorial, great as usual!

You mentioned that "each integer represents a unique word", why?

My assumption is that we have mapped each word to its frequency in the whole corpus. If my assumption is true, so two words could come up with the same frequency. For example “Dog” and “Cat” both could repeat 10 times in the corpus.

Could you please if my assumption is wrong?

Thanks,

Jason Brownlee June 2, 2017 at 11:33 am REPLY

Words were ordered by frequency then assigned integers based on that frequency.

Agustin February 3, 2017 at 9:15 am # REPLY

Hello Jason, recently i became acquainted with the basics in machine learning and deep learning, in part thanks to the information provided in this site, which I found most insightful.
However, lately I came upon a problem of generation simple and short questions automatically from a text. Due to my lack of knowledge and expertise i cant asses if it is possible to solve this problem with Deep Learning or any other method. Currently I have a database of several thousand questions based on around a hundred corpora that could be used as training data. Do you think I could get any successful results, and if so what approach will be the best? (Consider that even if it makes gibberish 50% of the time, it will still save a lot of work)

Jason Brownlee February 3, 2017 at 10:19 am # REPLY

It does sound like a good deep learning problem Agustin.

I'd recommend reading some papers on the topic to spark some ideas on methods and ways to represent the problem.

Try searching on google scholar and on arvix.

Chris Shumaker February 9, 2017 at 7:15 am # REPLY

Hi, thank you for the example! Do you know the NumPy and matplotlib versions you were using in this example? I am having trouble with several methods like mean, std and box plot.

Jason Brownlee February 9, 2017 at 7:28 am # REPLY

Hi Chris,

This might be a Python 2 vs Python 3 issue, I used Python 2.

Chris Shumaker February 9, 2017 at 7:34 am # REPLY

Actually, I am thinking that it is the call to map(). What version of Python are you using?

Chris Shumaker February 9, 2017 at 7:36 am #

Sorry for post-spam. This works in Python 3:

REPLY

```
# Summarize review length
print("Review length: ")
result = list(map(len, X))
print(type(result))
print(result)
```

Python 3.5.2 :: Anaconda 4.1.1 (x86_64)

Keras (1.2.1)

tensorflow (0.12.1)

numpy (1.11.2)

matplotlib (1.5.3)

Akshit Bhatia February 12, 2017 at 6:44 am #

REPLY

Hey,

Can you give me some idea on how to implement other Deep Learning techniqeus such as recursive autoencoders(RAE) , RBM deep learning algorithm for sentiment analysis

Any help would be appreciated :).

Regards

Jason Brownlee February 13, 2017 at 9:08 am #

REPLY

Hi Akshit,

I don’t have an example of RAE or RBM.

This post has an example of sentiment analysis that you can use as a starting point.

Zhang February 12, 2017 at 8:47 am #

REPLY

Hello, thanks for the example. I really appreciate you if you suggest me why I got this error.

File “C:\Anaconda\lib\site-packages\theano-0.9.0.dev4-py2.7.egg\theano\gof\cmodule.py”, line 2236, in compile_str
raise MissingGXX(“g++ not available! We can’t compile c code.”)

MissingGXX: (‘The following error happened while compiling the node’, Shape_i{1}(embedding_2_W), ‘\n’, “g++ not available! We can’t compile c code.”, ‘[Shape_i{1}(embedding_2_W)]’)

Chri February 12, 2017 at 2:51 pm #

REPLY

@Zhang, looks like you have a beta version of Theano. If you’re just looking to get started, maybe you want to try a stable channel instead? Looks like your error is because you’re installing from source and your environment isn’t set up quite right.

Jason Brownlee February 13, 2017 at 9:11 an

REPLY

Thanks for the note Chri.

Jason Brownlee February 13, 2017 at 9:09 am # REPLY

Hi Zhang,

It looks like g++ is not available. I'm not a windows user, I'm not sure how to interpret this message.

Consider searching or posting on stack overflow.

Chris February 12, 2017 at 2:55 pm # REPLY

@Jason, thanks for your reply and thanks again for the post!

I am having trouble improving the results on this model. I have changed the pool_length (500,250,125,375,5,10,15,20), tried adding another dense layer at size 250 and 500, and changed the number of epochs (25,50).

Do you have any recommendations for tweaking the model? I tried the suggestions (deeper, larger, pool_length, and also number of epochs). Do you have any tips or reading suggestions for improving performance in general? This seems to be my last 'wall' to really being able to do ML.

Thanks!

Jason Brownlee February 13, 2017 at 9:12 am # REPLY

Nice experiments Chris.

Perhaps changes to the structure of the problem itself are required.

This post might shake loose more ideas on how to improve performance: https://machinelearningmastery.com/improve-deep-learning-performance/

Kiran March 3, 2017 at 4:34 pm #

REPLY

Hi jason, I removed the denser layer with 250 neurons and it reduced the number of parameters to be trained drastically with an increased accuracy of about 1% over 5 epochs. Any idea why you added 2 dense layers after flatten layer?

Jason Brownlee March 6, 2017 at 10:42 am #

REPLY

Well done Kiran.

I came up with the configuration after some brief trial and error. It was not optimized.

Dan March 12, 2017 at 12:02 am #

REPLY

Does it make sense to specify validation_data as X_test,y_test in the fit function if we evaluate our model in the scores function afterwards? Or could we skip specify validation_data in model.fit(…)?

Jason Brownlee March 12, 2017 at 8:28 am #

REPLY

No, you will get this for free when fitting your model. The validation data should be different from training data and is completely optional.

Harish March 23, 2017 at 5:54 pm #

REPLY

What is the accuracy of this approach? Which approach is better to get accuracy of at least 0.89?

Jason Brownlee March 24, 2017 at 7:54 am #

REPLY

Accuracy: 88.28%

Prakash April 12, 2017 at 8:23 am #

REPLY

I tried the sentiment analysis with convolutional NNs and LSTMs and find the CNNs give higher accuracy. Any insight into why?

Jason Brownlee April 12, 2017 at 9:35 am #

REPLY

Some ideas:

Perhaps the CNNs are better at capturing the spatial relationships.

Perhaps the LSTM needs to be larger and trained for longer to achieve the same skill.

Patrick April 17, 2017 at 9:20 pm #

REPLY

Please add these to the imports

from keras.preprocessing import sequence

from keras.layers.embeddings import Embedding

Jason Brownlee April 18, 2017 at 8:32 am #

REPLY

These are listed in the imports for the section titled "One-Dimensional Convolutional Neural Network Model for the IMDB Dataset"

Trang April 27, 2017 at 2:21 pm #

REPLY

Hi, Jason, i have the same question with Maxim. Can you tell me why is that.Thank you

Maybe I ask you a very stupid question, but I can't understand one thing. What is the Embedding layer? Could you show an example of it. I mean this is word vector with dimentionality of 500X32. So how it looks like?

[0,0,1,….0,0,0] X 32

.

X500

.

[0,1,0,….0,0,0] X 32

What digits in it? And why if we lift it dimensionality up to 64, the accuracy will rise up?

Thank you!

Jason Brownlee April 28, 2017 at 7:34 am #

REPLY ↰

Great question.

The embedding layer is a mapping of integers to a higher dimensional space. See here:

https://keras.io/layers/embeddings/

And here:

https://en.wikipedia.org/wiki/Word_embedding

I hope to cover this in more detail in a dedicated post.

Ahmed May 10, 2017 at 3:19 am #

REPLY ↰

Thanks So So Much For This Awesome Tutorial .

However , I'm Asking About How To Use This To Predict The Opinion

I Still Don't know , for instance i need to know the movie is good or bad , or if i used a twitter data set i need to know the public opinion summary about a specific hashtag or topic

I tried More And More But i Failed as i'm still Beginner

Thanks In Advance <3

Hamza May 24, 2017 at 5:30 am #

REPLY

Well, thank you so much for this great work.

I have a question here. I didn't understand why we use ReLU instead of tanh as the activation function. Most people use SGD or backpropagation for training. What did we use here? I do not know about ADAM. Can you please explain why did you use it for training?

Jason Brownlee June 2, 2017 at 11:28 am #

REPLY

ReLU has better properties than sigmoid or tanh and has become the new defacto standard.

We did use SGD to fit the model, we just used a more fancy version called Adam.

Adnan June 12, 2017 at 2:49 am #

REPLY

Hi, thanks a lot for this HELPFUL tutorial. I have a question, could be an odd one. what if we use pre-trained word2vec model. I mean if we just use pre-trained word2vec model and train our neural network with movie reviews data. Correct me if I am wrong!

Or best way is to train word2vec with movie reviews data, then train neural network with same movie reviews data then try.

Kindly guide me. Thanks

Jason Brownlee June 12, 2017 at 7:11 am #

REPLY

Sounds great, try it.

Adnan June 12, 2017 at 7:53 pm #

REPLY

but should I use pre-trained word2vec model (trained with wiki data) or train from scratch with by using movie reviews data or amazon product reviews data. thanks

Jason Brownlee June 13, 2017 at 8:19 am

REPLY

Try both and see what works best on your problem.

Alok June 22, 2017 at 12:42 am #

REPLY

Hi Jason,

I am trying to use tfidf matrix as input to my cnn for document classification. I don't want to use embedding layer. Can you help me how this can be achieved. I have not seen any example which shows tfidf as input to Cov1d layer in Keras. Please help

Jason Brownlee June 22, 2017 at 6:10 am #

REPLY

Sorry, I have not used tfidf as input to a CNN.

Matteo July 7, 2017 at 5:50 pm #

REPLY

Dear Jason, I do thank you for this great post. Could you give me any indications on how to extend this approach to multiclass classification?

Jason Brownlee July 9, 2017 at 10:38 am #

REPLY

Set the number of nodes in the output layer to the number of classes and change activation to softmax.

Prakash July 12, 2017 at 2:52 am #

REPLY

I see that you have not used any regularizers in the model. How is overfitting avoided?

Jason Brownlee July 12, 2017 at 9:50 am #

REPLY

Here, by an underspecified model and under-training the model.

You could also try dropout and weight regularization.

Rahul July 22, 2017 at 4:53 am #

Hey Jason great review but I was wondering how I c REPLY

created model to predict the sentiment of a new inputted te

Jason Brownlee July 22, 2017 at 8:39 am # REPLY

You can encode your test using the same method as in the problem in order to make a prediction.

I hope to have many more NLP examples soon.

mm August 23, 2017 at 5:45 pm # REPLY

5000 means ?

Jason Brownlee August 24, 2017 at 6:27 am # REPLY

That is the number of samples in the train and test sets.

Dev September 13, 2017 at 8:03 pm # REPLY

I am getting an error: name 'sequence' is not defined in the line: X_train = sequence.pad_sequences(X_train, maxlen=500)

Jason Brownlee September 15, 2017 at 12:04 pı REPLY

Ensure you copy all of the code, including this line:

```python
from keras.preprocessing import sequence
```

Hamada Zahera September 21, 2017 at 6:55 pm #

REPLY

Hi Jason,

Thanks for this nice post . I have a question about how to use the model to predict sentiment of a new text .

myText="Hello, this is a my review"

How to structure this text into model ? and use predict function.

Jason Brownlee September 22, 2017 at 5:37 am

REPLY

I have a suite of posts on this topic scheduled for the coming weeks if you can hang on?

Dimuthu de Silva September 23, 2017 at 2:54

REPLY

I am stuck with making new predictions too..

Jason Brownlee September 24, 2017 at 5:

REPLY

What is the problem exactly?

Jesse December 2, 2017 at 10:23 am #

I am having a difficult time encoding new, inputted text. Am I able to simply pass [“I hate computer bugs”] to pad_sequences()? Or is it a different approach? Any pointers would be very much appreciated!

Jason Brownlee December 3, 2017 at 5:23 am #

The words must be encoded as integers first, see this post:
https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

Parth October 5, 2017 at 1:34 am #

REPLY

Hi Jason,

Thanks for the blog and tutorial. Since you talked about IMDB standard dataset. I just wrote a blog mentioning the accuracies of state-of-the-art models for sentiment analysis on IMDB dataset. You can see it here:
http://blog.paralleldots.com/technology/nlp/breakthrough-research-papers-and-models-for-sentiment-analysis/

Jason Brownlee October 5, 2017 at 5:25 am #

REPLY

Thanks for sharing.

Asad October 7, 2017 at 4:49 pm #

REPLY

Hi jason,

you did a great work. i appreciate your effort. Actually i just want to know,in "One-Dimensional Convolutional Neural Network Model for the IMDB Dataset". how i can know the answer of these question.

1)number of neurons used in this tutorial?

2)no of hidden layers?

3)no of out put layers?

4)if no of hidden layers maximize and minimize then output will affect or not?

5)how many parameters is used?

6)how we can draw a network diagram in python using code or library?

please answer me briefly. thanks

Jason Brownlee October 8, 2017 at 8:28 am #

REPLY

We cannot know for sure, these configuration hyperparameters cannot be specified analytically. You must use experiments to discover what works best for a given problem.

You can draw a network diagram in Keras, here are the API details:
https://keras.io/visualization/

Asad October 10, 2017 at 8:36 pm #

how i can do experiments? on which bases i can do

REPLY

do you have any tutorial in which you take a dataset

any NN for sentiment analysis?

Jason Brownlee October 11, 2017 at 7:51

REPLY

Yes, I have a few posts scheduled for later this month.

Asad October 7, 2017 at 4:57 pm #

REPLY

hi jason

i want to know that how i can use this model for my own data set which is in csv file. the data set have polarity of idiom sentences.

Jason Brownlee October 8, 2017 at 8:29 am #

REPLY

I will have an example on the blog soon.

Asad October 9, 2017 at 3:30 pm #

REPLY

ok jason i am waiting.. please also send notification to me. because on your blog there is no notification or email recieved about any answer.

Yang Cheng November 1, 2017 at 2:13 pm #

REPLY

Hi Jason, great post! I have learnt a lot from your previous posts. Thanks a lot!

For sentiment analysis, we can get better result (0.8822 in this case) if we change the output layer to softmax activation.

Jason Brownlee November 1, 2017 at 4:15 pm #

REPLY

Great tip, thanks!

Sai Rohit S November 14, 2017 at 5:36 pm #

REPLY

Hi Jason.

I have used a CNN Model for Sentiment Analysis.

I have a set of my own strings to test the model on.

What is the conversion I need to make on my Strings to input them to the model?

Thanks.

Jason Brownlee November 15, 2017 at 9:49 am

REPLY

You can use a bag of words model or a word embedding.

I have posts on both, perhaps start here:

https://machinelearningmastery.com/start-here/#nlp

Sai Rohit S November 15, 2017 at 2:17 pm #

REPLY

Thank You.

I will go through it.

Jason Brownlee November 16, 2017 at 10:25 am

REPLY

Let me know how you go.

Amit Adesara November 23, 2017 at 5:44 pm #

REPLY

Hi, Does Conv2D give better accuracy. What will be the input shape in that case. Have your covered Conv2D in your book on Deep Learning for NLP.

Thanks.

Jason Brownlee November 24, 2017 at 9:36 am

REPLY

It is not a question of accuracy, rather what is appropriate for the data.

1D data like a sequence of words requires a 1D convolutional neural net.

Amit Adesara November 29, 2017 at 6:54 pm #

Hi Jason, in one of the examples in your book (chapter on text classification) the model.predict requ input arrays as we are working with 3 channels. Though model.evaluate runs fine i fail to understand where do we define the three input in predict.sentiment function.

Will be helpful if you can guide.

REPLY ↰

Jason Brownlee November 30, 2017 at 8:0

REPLY ↰

You can pass 3 inputs to the predict function as an array:

```
x1, x2, x3 = ...
yhat = model.predict([x1,x2,x3])
```

Does that help?

Abdur Rehman Nadeem December 14, 2017 at 5:4

REPLY ↰

Hi Jason,

You used pre-built dataset but if I want to run this model on my dataset (e.g. in my case i want to run this on my tweets dataset) how can I made my dataset compatible to this blog code. Please give some suggestions or reference so that I can make tweets dataset accordant to this model.

Best Regards,

Jason Brownlee December 14, 2017 at 5:42 am #

This post shows how:

https://machinelearningmastery.com/develop-word-emb

model-predicting-movie-review-sentiment/

REPLY

Ali January 28, 2018 at 4:09 pm #

REPLY

Thanks Jason,

I learned a lot from your posts.

This model performs well for binary classification but poorly on multiclass. My question is, in case of multiclass (say 10 or more classes), what changes in layers or their parameters will improve the accuracy?

Jason Brownlee January 29, 2018 at 8:15 am #

REPLY

I would recommend trying a suite of configurations to see what works best on your specific data.

shweta khairnar February 21, 2018 at 7:56 am #

REPLY

hey jason, hi great tutorial but i don;t know why i am getting an error in pad.sequences line it is saying name sequnce is not defined i do not understand why i am getting this error? can you help?

Jason Brownlee February 22, 2018 at 11:10 am

REPLY

Sorry to hear that. Are you able to check that you copied all of the code?

bhavik February 23, 2018 at 5:15 pm #

REPLY

Hey thanks for the great work, just wanted to know that how to print out the results of this prediction as what are the positive and negative comments?

Jason Brownlee February 24, 2018 at 9:10 am #

REPLY

What problem are you having exactly?

lana February 28, 2018 at 10:02 pm #

REPLY

i also have this mistake: NameError Traceback (most recent call last)

in ()

—-> 1 X_train = sequence.pad_sequences(X_train, maxlen=500)

2 X_test = sequence.pad_sequences(X_test, maxlen=500)

NameError: name 'sequence' is not defined

Jason Brownlee March 1, 2018 at 6:13 am #

REPLY

Ensure you have copied all code required.

Svetlana Bondareva March 5, 2018 at 8:15 am #

REPLY

hey Jason, how can I use “predict” with the model? for example, I have a text and I want to see the outcome based on the model. TIA

Jason Brownlee March 6, 2018 at 6:07 am #

REPLY

You can use model.predict() with new data as an argument.

Note that new data will need to be prepared in the same way as the training data.

aima othman March 5, 2018 at 6:44 pm #

REPLY

can you share how to do a prediction with the model?

Jason Brownlee March 6, 2018 at 6:11 am #

REPLY

Sure:

```
newX = ...
yhat = model.predict(newX)
```

Uchenna Iheanacho March 10, 2018 at 6:56 pm #

REPLY

Hey Jason,

How do I classify the user as angry or not angry based on the type of words in the review sent?

Jason Brownlee March 11, 2018 at 6:22 am # REPLY

Start by collecting examples of angry and not angry movie reviews.

mariya March 25, 2018 at 3:51 am # REPLY

hi Mr jason … can u tell howa how can i do the file test of this model …

and thanks !

Jason Brownlee March 25, 2018 at 6:34 am # REPLY

What do you mean by “file test”?

PForet March 28, 2018 at 4:27 am # REPLY

Very nice article! Clear and precise. I just have some reserves concerning the use of deep learning for sentiment classification. Sure, it seems to be the future somehow, but for now, we just get better results with Bayesian methods. For instance, I worked with the same dataset (see http://www.ml-hack.com/bayesian-features-machines/) and get 91.6% accuracy with Bayesian learning. Do you think deep learning performs less because the task is too simple? Or because the dataset is too small?

I’d love to hear your thoughts about that

Jason Brownlee March 28, 2018 at 6:30 am #

REPLY

It sees CNNs are doing very well also:

https://machinelearningmastery.com/best-practices-document-classification-deep-learning/

Jane_in May 30, 2018 at 11:36 pm #

REPLY

Hye Jason. I would like to tell that as my first attempt I tried multi-layerd perceptron and i have this issue that is it okay to use the same data for validating as well a testing? if we will validate our training on test data then the result will be biased??
And whenever I am increasing no. of epochs loss is increasing simultaneously and accuracy is just around 86% always. Please guide me.

Jason Brownlee May 31, 2018 at 6:18 am #

REPLY

It is a good idea to evaluate the skill of a model on data not used to train it, learn more here:

https://machinelearningmastery.com/faq/single-faq/how-do-i-evaluate-a-machine-learning-algorithm

Ram June 4, 2018 at 4:13 pm #

REPLY

Hi Jason!!!

what will be x_train,x_test,y_train and y_test in case of twitter sentiments where labels are floating point values?

I am not getting what are these lists?

can you clear it?

Jason Brownlee June 5, 2018 at 6:34 am #

REPLY

Inputs will be text, output will be a sentiment class label.

Train and test will be some split of the data for fitting and evaluating the model respectively.

Jane_in June 4, 2018 at 8:29 pm #

REPLY

I want to know the reason for increasing validation loss after 2nd epoch..?? is there any mistake i am doing?

Jason Brownlee June 5, 2018 at 6:38 am #

REPLY

Perhaps the model is overfitting?

Jane_in June 6, 2018 at 11:24 pm #

REPLY

It is the MLP code in your example what should i do to increase the accuracy as well as remove overfitting.. will dropout layer works??

Jason Brownlee June 7, 2018 at 6:29 am # REPLY

It may. Here are more ideas:

https://machinelearningmastery.com/improve-deep-learning-performance/

Bharath June 8, 2018 at 7:50 pm # REPLY

Thank you Jason for this!

I am really new to deep learning and NLP.

Here are few naive questions that I have:
– Why did you select 32 as the parameter?
– Can we use this learned model to now predict any other text data? say for eg, i wanted to evaluate feedback data of customers, can i use the same model to do so? If so, how?
– How are you taking care of stop words and other irrelevant terms in this text?

Thanks in advance 🙂

Jason Brownlee June 9, 2018 at 6:50 am # REPLY

I used experimental testing to configure the model. You can learn more here:

https://machinelearningmastery.com/faq/single-faq/how-many-layers-and-nodes-do-i-need-in-my-neural-network

Yes, within reason (e.g. from the same domain).

I often remove stop words. Here is a fuller example:

https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/

Kaushal June 11, 2018 at 4:33 pm #

REPLY

Hi Jason,

What is the current benchmark accuracy for imdb movie review classification?

Jason Brownlee June 12, 2018 at 6:35 am #

REPLY

I'm not sure, accuracy above 88% is excellent.

Bharath Nair June 12, 2018 at 12:45 am #

REPLY

thank you Jason for the prompt response 🙂 Really appreciate it.

Jason Brownlee June 12, 2018 at 6:44 am #

REPLY

No problem.

Ritwik June 12, 2018 at 5:22 am #

REPLY

Hey Jason!

excellent stuff. How to use your code to predict the same corpus and model to predict employee feedback information? I do not understand how to use this on some other data. How to use your code as starting point?

Jason Brownlee June 12, 2018 at 6:50 am #

REPLY

This process will help you work through a new predictive modeling problem:

https://machinelearningmastery.com/start-here/#process

SOORAJ June 21, 2018 at 10:28 pm #

REPLY

Your tutorials are very helpful as a beginner like me. I have a doubt if we are using a dataset having only two labels (class & text) and then how many input neurons should I create. Is it 1 or more than one..??

Jason Brownlee June 22, 2018 at 6:08 am #

REPLY

This is a common question that I answer here:

https://machinelearningmastery.com/faq/single-faq/how-many-layers-

and-nodes-do-i-need-in-my-neural-network

SOORAJ June 25, 2018 at 5:25 pm #

REPLY

What is the use of Flatten()?

Jason Brownlee June 26, 2018 at 6:34 am #

REPLY

In some models, the network will have a 2d or 3d internal shape to the data. Flatten squashes this down to 1D as the fully-connected layers expect.

SOORAJ June 26, 2018 at 10:05 pm #

REPLY

thank you…..

Ishay Telavivi August 14, 2018 at 6:04 am #

REPLY

Hi Jason,

Thanks so much. Great post!

I have the followng questions:
1. You explained well why maxlen was set to 500. Is there a way to examine other lengths is an easy way (like we grid search over the model hyperparameters), and not manually? Is it important?

2. On the “One Dimensional CNN” section, you used pool_size=2 within the maxpool function. What is reason/benefit for that?

3. Is the flatten layer right after it is because the pool_size is 2? I mean – I won’t be needing it if I just take the default?

Jason Brownlee August 14, 2018 at 6:26 am #

REPLY

You can experiment with other lengths.

Max pool of 2 reduces he size of the filter maps to 1/4 their size. It is a commonly used configuration.

Flatten is to reduce the filter map structure down to a vector that the dense layer can take as input.

Ishay Telavivi August 15, 2018 at 12:28 am #

REPLY

OK Thank you

Jason Brownlee August 15, 2018 at 6:04 am #

REPLY

You’re welcome.

Bahara October 1, 2018 at 8:31 pm #

REPLY

Hi.

I have a question, assuming that i want to inject some hand-crafted features into CNN layers for sentiment analysis.

At first i want to know is it possible?

And then how can i use fully connected layer to do this?? I dont want to use any methods like svm or … just want to use combinitation of deep and hand-crafted features directly in cnn. Thank you

Jason Brownlee October 2, 2018 at 6:24 am #

REPLY

Yes, you could have a multiple-input model, one input is the text, the other are the new features.

I have many examples of this type of model on the blog, perhaps start here:

https://machinelearningmastery.com/keras-functional-api-deep-learning/

PIYUSH KILLA November 25, 2018 at 12:43 am #

REPLY

how can I predict the result for my new comment?

ex = “this movies is fantastic”

Jason Brownlee November 25, 2018 at 6:58 am

REPLY

You must prepare the data in the same way and call model.predict().

Perhaps try this tutorial:

https://machinelearningmastery.com/develop-word-embedding-model-predicting-movie-review-sentiment/

Belgacem BRAHIMI February 25, 2019 at 4:41 am # REPLY

Hello

thanks for this tutorial

I have a question about how to use cross- validation ?

Jason Brownlee February 25, 2019 at 6:47 am # REPLY

This post shows you how:

https://machinelearningmastery.com/k-fold-cross-validation/

Mithu March 18, 2019 at 12:01 am # REPLY

Hi, how do you modify the learning rate?

Jason Brownlee March 18, 2019 at 6:05 am # REPLY

Good question, this tutorial explains how:

https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/

Chan Kim March 22, 2019 at 1:26 pm #

REPLY

Hi, Jason, thank you again for these kind tutorials. I'm learning many things here.
In the CNN case, I see from the model summary that the number of parameters in the first Embedding layer is 160000 (5000*32) and I can understand this. But why is the number of parameters of the first Conv1D layer 3104? I guess the 500 input words are transformed to 500 * 32 output values in the Embedding layer and the convolution is done for this 500*32 inputs values with kernel size 3. Is my understanding correct?
And I found the only way I can make 3104 seems to be 3*32*32+32 but I cannot think of any way the 1D conv parameters are applied for the input values. Could you elaborate on 1-D convolution in this case? (I read https://blog.goodaudience.com/introduction-to-1d-convolutional-neural-networks-in-keras-for-time-sequences-3a7ff801a2cf but the explanation is confusing to me..)

Jason Brownlee March 22, 2019 at 2:36 pm #

REPLY

A good starting point is to summarize the model and look at the output shape of each layer.

Chan Kim March 22, 2019 at 1:35 pm #

REPLY

Hi, Jason, this is a follow-up question to my previous question.

I thought 3104 = (32+1) * (3*32) so there are 3 kernel params for each 32 values for the inputs, and these kernel values are each different on each 32 inputs(3*32) and all the 32 inputs values are used to make each one of 32 output values with bias(thus * (32+1)). Hope you could understand what I mean.. 🙂

Royal June 29, 2019 at 8:17 pm #

REPLY

Hi Jason,

I'm getting an error with the command (X_train, y_train), (X_test, y_test) = imdb.load_data()

using Python 3.6.8 [Anaconda], win32 version on Windows10:

Traceback (most recent call last):

File "", line 2, in

File "C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_3.6\lib\site-packages\keras\datasets\imdb.py", line 59, in load_data

x_train, labels_train = f['x_train'], f['y_train']

File "C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_3.6\lib\site-packages\numpy\lib\npyio.py", line 262, in __getitem__

pickle_kwargs=self.pickle_kwargs)

File

"C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\numpy\lib\format.py”, line 692, in read_array

raise ValueError(“Object arrays cannot be loaded when ”

ValueError: Object arrays cannot be loaded when allow_pickle=False

Jason Brownlee June 30, 2019 at 9:39 am #

REPLY

I’m sorry to hear that, perhaps try updating NumPy?

Royal June 30, 2019 at 5:04 pm #

REPLY

Ok, I upgraded from ‘1.16.3’ –> ‘1.16.4’ and obtained the same error.

Does (X_train, y_train), (X_test, y_test) = imdb.load_data() still work for you?

This is an interesting tutorial, would be a pity if it could no longer be used1

Jason Brownlee July 1, 2019 at 6:32 am #

REPLY

I have a fix, add these lines to the start of the code example:

```
import numpy as np
np_load_old = np.load
np.load = lambda *a,**k: np_load_old(*a, allow_pic
```

Based on:

https://stackoverflow.com/questions/55890813/how-to-fix-object-arrays-cannot-be-loaded-when-allow-pickle-false-for-imdb-loa

Royal July 4, 2019 at 7:44 pm #

REPLY

Brilliant, thanks, that brings us forward! Before uncorking the champagne bottle, two more incompatibilities show up next:

1)

print("Mean %.2f words (%f)" % (numpy.mean(result), numpy.std(result)))

Traceback (most recent call last):

File "", line 1, in

File "C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_3.6\lib\site-packages\numpy\core\fromnumeric.py", line 3118, in mean out=out, **kwargs)

File "C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_3.6\lib\site-packages\numpy\core_methods.py", line 87, in _mean ret = ret / rcount

TypeError: unsupported operand type(s) for /: 'map' and 'int'

2)

pyplot.boxplot(result)

pyplot.hist(result)

Traceback (most recent call last):

File "C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_3.6\lib\site-packages\matplotlib\units.py", line 168, in get_converter

```
if not np.all(xravel.mask):

AttributeError: ‘numpy.ndarray’ object has no attribute ‘mask’

During handling of the above exception, another exception occurred:

Traceback (most recent call last):

File “”, line 1, in

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\pyplot.py”, line 2659, in hist

**({“data”: data} if data is not None else {}), **kwargs)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\__init__.py”, line 1810, in inner

return func(ax, *args, **kwargs)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\axes\_axes.py”, line 6534, in hist

self._process_unit_info(xdata=x[0], kwargs=kwargs)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\axes\_base.py”, line 2135, in

_process_unit_info

kwargs = _process_single_axis(xdata, self.xaxis, ‘xunits’, kwargs)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\axes\_base.py”, line 2118, in
```

_process_single_axis

axis.update_units(data)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\axis.py”, line 1467, in update_units

converter = munits.registry.get_converter(data)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\units.py”, line 181, in get_converter

converter = self.get_converter(next_item)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\units.py”, line 187, in get_converter

thisx = safe_first_element(x)

File

“C:\Users\XXX\AppData\Local\Continuum\anaconda3\envs\env_python_

3.6\lib\site-packages\matplotlib\cbook\__init__.py”, line 1635, in

safe_first_element

raise RuntimeError(“matplotlib does not support generators ”

RuntimeError: matplotlib does not support generators as input

---

Jason Brownlee July 5, 2019 at 8:04 am #

REPLY

Sorry to hear that.

I can confirm that the code listings work with Keras 2.2.4 and

TensorFlow 1.14.0.

Are you able to confirm that you copied all of the code and that your libraries are up to date?

Royal July 5, 2019 at 5:28 pm #

REPLY

1) It works now, many thanks. You might want to include the code in your example above so it works for others:

import numpy as np

np_load_old = np.load

np.load = lambda *a,**k: np_load_old(*a, allow_pickle=True, **k)

2) The error message resulted from the following code (under # Summarize review length):

result = map(len, X) # Bad version

result = [len(x) for x in X] # Good version which creates no errors.

Did you just correct this? Otherwise I have no explanation why copy-and-paste would have introduced "result = map(len, X)" before.

Brilliant work!

Jason Brownlee July 6, 2019 at 8:27 am #

REPLY

Thanks. I am hoping there is a new Keras release around the corner that already incldues the fix.

The code works as is with Python 3.6 run from the command line. Is it possible you were running in an IDE/Notebook or using an older version of Python?

Royal July 6, 2019 at 6:55 pm #

REPLY

I'm using python 3.6.8 in command-line mode with Windows 10.

I've never encountered anyone so dedicated to support their online learning service as you. I'm amazed.

In summary, thanks to your input, the example works fine now.

– Dr. Royal Truman

Jason Brownlee July 7, 2019 at 7:50 am #

REPLY

Thanks, I'm very happy to hear that it's now working!

Swarupa De August 2, 2019 at 8:44 am #

REPLY

Hi Jason.

I was trying to implement Conv2D for the same thing. I had kept the kernel size as (3,3) however this didn't seem to work. But as i switched to Conv1D after looking at your example i got it working. Could you let me know why you decided to use 1D and if there are any specific use cases for either.

Thanks

Swarupa

Jason Brownlee August 2, 2019 at 2:36 pm #

REPLY ↰

I don't believe that a conv2d would be appropriate for text input.

We use a conv1d because we are working with 1d sequence of words.

Liberty November 17, 2019 at 12:17 pm #

REPLY ↰

Hi Mr. Jason, could tell me how can I do a prediction with the model picking a single comment out of the dataset as the input?

Jason Brownlee November 18, 2019 at 6:41 am

REPLY ↰

Yes, you can have one sample input and call predict().

This will help:
https://machinelearningmastery.com/how-to-make-classification-and-regression-predictions-for-deep-learning-models-in-keras/

Ilias P. May 14, 2020 at 4:50 am #

REPLY ↰

Jason, thank you for your beautiful lessons. I see that in this example you are not using any sliding window, or timesteps for your input in the lstm model. However, in other lstm examples, and generally in the internet, i have seen that it’s useful to transform the dataset with timesteps, when using lstm models? Is it true? When i need to preproccess the data with timesteps? Thank you in advance

Jason Brownlee May 14, 2020 at 5:58 am #

REPLY

You’re welcome.

This is text data, not time series. If you want to see examples of LSTMs for time series, start here:
https://machinelearningmastery.com/start-here/#deep_learning_time_series

kooni June 30, 2020 at 4:34 pm #

REPLY

Thank you for this tutorial
Is it right to use lstm or cnn-lstm in this dataset?

Jason Brownlee July 1, 2020 at 5:51 am #

REPLY

You’re welcome.

A CNN or LSTM with a word embedding would probably most appropriate for this type of problem.

Alexey July 16, 2020 at 10:43 pm #

Hi Jason,

REPLY

Thanks a lot for your blog/tutorial/book, which are excellent.

I have been struggling trying to make a similar notebook (using imdb data from keras.datasets) work on a TPU(s) by using Google Colab.

Any advice on how to make the code in this tutorial work on TPU?

Thank you,

Alex

Jason Brownlee July 17, 2020 at 6:17 am #

REPLY

You're welcome.

Sorry, I don't know about colab:

https://machinelearningmastery.com/faq/single-faq/do-code-examples-run-on-google-colab

I recommend running examples on your workstation.

Johan August 13, 2020 at 5:18 pm #

REPLY

As comparison, a traditional approach with CountVectorizer and TfidfTransformer reaches 84.26% accuracy with MultinomialNB and 88.68% with LinearSVC (using top 5k words)

Jason Brownlee August 14, 2020 at 5:58 am #

REPLY

Well done! Thanks for sharing.

Pique March 6, 2024 at 7:59 pm #

REPLY

I have been trying to get the code (MLP for IMDB proble) above trying to access a locally saved archive (imdb.npz) downloaded from the source.

However everytime I get the same result (Accuracy: 50%).

The output says:

```
Train on 25000 samples, validate on 25000 samples
Epoch 1/2
– 93s – loss: 7.9342 – acc: 0.4999 – val_loss: 7.9712 – val_acc: 0.5000
Epoch 2/2
– 108s – loss: 7.9712 – acc: 0.5000 – val_loss: 7.9712 – val_acc: 0.5000
Accuracy: 50.00%
```

What am I doing wrong here.

A word on the above wuld be highly appreciated.

James Carmichael March 7, 2024 at 9:47 am #

REPLY

Hi Pique…It is possible for the performance of your model to get stuck during training.

This can happen with neural networks that achieve a specific loss, error or accuracy and no longer improve, showing the same score at the end of each subsequent epoch.

In the simplest case, if you have a fixed random number seed for the code example, then try changing the random seed, or do not specify the seed so that different random numbers are used for each run of the code.

Then try running the example a few times.

To learn more about randomness in machine learning see this post:

Embrace Randomness in Machine Learning

To learn more about pseudorandom number generators in machine learning, see this post:

Introduction to Random Number Generators for Machine Learning in Python

If the problem is not the randomness, it is possible that your model has converged. If the skill of the model is not good after convergence, this is referred to as premature convergence.

You can address premature convergence by slowing down the learning by the model. You can do this by changing different hyperparameters for the learning process, such as:

Using a smaller learning rate.
Using a larger network (nodes or layers).
Using a training dataset with more examples.
And so on…

Premature convergence is a big area of study and I recommend reading up further on the topic if you need further ideas.

Leave a Reply

Name (required)

Email (will not be published) (required)

SUBMIT COMMENT

Welcome!

I'm *Jason Brownlee* PhD

and I **help developers** get results with **machine learning.**

Read more

Never miss a tutorial:

Picked for you:

Your First Deep Learning Project in Python with Keras Step-by-Step

How to Grid Search Hyperparameters for Deep Learning Models in Python with Keras

Regression Tutorial with the Keras Deep Learning Library in Python

Multi-Class Classification Tutorial with the Keras Deep Learning Library

How to Save and Load Your Keras Deep Learning Model

>> SEE WHAT'S INSIDE