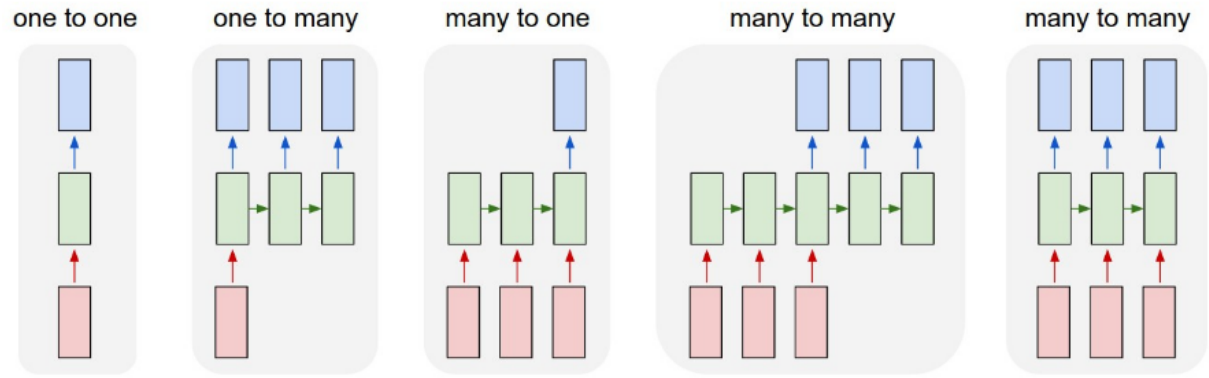
How to structure an LSTM neural network for classification

Asked 5 years, 10 months ago Modified 5 years, 6 months ago Viewed 2k times

5

I have data that has various conversations between two people. Each sentence has some type of classification. I am attempting to use an NLP net to classify each sentence of the conversation. I tried a convolution net and get decent results (not ground breaking tho). I figured that since this a back and forth conversation, and LSTM net may produce better results, because what was previously said may have a large impact on what follows.



If I follow the structure above, I would assume that I am doing a many-to-many. My data looks like.

```
X_train = [[sentence 1],
           [sentence 2],
           [sentence 3]]
Y_train = [[0],
           [1],
           [0]]
```

Data has been processed using word2vec. I then design my network as follows..

```
model = Sequential()
model.add(Embedding(len(vocabulary),embedding_dim,
          input_length=X_train.shape[1]))
model.add(LSTM(88))
model.add(Dense(1,activation='sigmoid'))
model.compile(optimizer='rmsprop',loss='binary_crossentropy',
              metrics['accuracy'])
model.fit(X_train,Y_train,verbose=2,nb_epoch=3,batch_size=15)
```

I assume that this setup will feed one batch of sentences in at a time. However, if in model.fit, shuffle is not equal to false its receiving shuffled batches, so why is an LSTM net even useful in this case? From research on the subject, to achieve a many-to-many structure one would need to change the LSTM layer too

```
model.add(LSTM(88,return_sequence=True))
```

and the output layer would need to be...

```
model.add(TimeDistributed(Dense(1,activation='sigmoid')))
```

When switching to this structure I get an error on the input size. I'm unsure of how to reformat the data to meet this requirement, and also how to edit the embedding layer to receive the new data format.

Any input would be greatly appreciated. Or if you have any suggestions on a better method, I am more than happy to hear them!

python neural-network nlp keras lstm

Share Improve this question Follow

edited Feb 20, 2017 at 1:06

asked Feb 19, 2017 at 23:45

DJK
8,626 4 23 39

1 Answer

Sorted by: Highest score (default)

4

Your first attempt was good. The shuffling takes place between sentences, the only shuffle the training samples between them so that they don't always come in in the same order. The words inside sentences are not shuffled.

Or maybe I didn't understand the question correctly?

EDIT :

After a better understanding of the question, here is my proposition.

Data preparation : You slice your corpus in blocks of `n` sentences (they can overlap). You should then have a shape like `(number_blocks_of_sentences, n, number_of_words_per_sentence)` so basically a list of 2D arrays which contain blocks of `n` sentences. `n` shouldn't be too big because LSTM can't handle huge number of elements in the sequence when training (vanishing gradient). Your targets should be an array of shape `(number_blocks_of_sentences, n, 1)` so also a list of 2D arrays containing the class of each sentence in your block of sentences.

Model :

```
n_sentences = X_train.shape[1]  # number of sentences in a sample (n)
n_words = X_train.shape[2]      # number of words in a sentence

model = Sequential()
# Reshape the input because Embedding only accepts shape (batch_size, input_length) so
```

```
we just transform list of sentences in huge list of words
model.add(Reshape((n_sentences * n_words,),input_shape = (n_sentences, n_words)))
# Embedding layer - output shape will be (batch_size, n_sentences * n_words,
embedding_dim) so each sample in the batch is a big 2D array of words embedded
model.add(Embedding(len(vocabaulary), embedding_dim, input_length = n_sentences *
n_words ))
# Recreate the sentence shaped array
model.add(Reshape((n_sentences, n_words, embedding_dim)))
# Encode each sentence - output shape is (batch_size, n_sentences, 88)
model.add(TimeDistributed(LSTM(88)))
# Go over lines and output hidden layer which contains info about previous sentences -
output shape is (batch_size, n_sentences, hidden_dim)
model.add(LSTM(hidden_dim, return_sequence=True))
# Predict output binary class - output shape is (batch_size, n_sentences, 1)
model.add(TimeDistributed(Dense(1,activation='sigmoid')))
...
```

This should be a good start.

I hope this helps

Share Improve this answer Follow edited Feb 20, 2017 at 8:11

answered Feb 20, 2017 at 6:55

Nassim Ben

11.2k 1 33 50

So are you saying the LSTM layer will get fed one word at a time? So even though the sentence's are being shuffled, each word in a sentence is passed to the LSTM separately to learn the overall context between a sentence as a whole? – DJK Feb 20, 2017 at 7:02

If i did not phrase my question correctly I'm sorry. Since the data is a conversation, what was said in the previous sentence has weight on the following sentence. So I'm trying to setup the network to learn the conversation flow and categorize each sentence. That was why I was attempting to use the return_sequence, so the network would hold information about the previous sentence, while classifying the current sentence. – DJK Feb 20, 2017 at 7:07

An LSTM is fed a sequence of vectors. In your case it is a sequence of words embeddings. It will return a vector of length 88 for each sentence in your case, which you reduce to 1output with the dense layer. So it only cares about one sentence at a time. That is what you currently do. Is that what you wanted to do? – Nassim Ben Feb 20, 2017 at 7:07

Ok thats a whole new story. Then you should reshape your inout data and send it as blocs of multiple sentences. Then encode each lines with LSTM used time distributed on each sentence. You will get a sequence of sentences all represented by a 88 length vector. Then use a second LSTM, with return sequence = True, which will output a vector for each sentence again but taking into account the previous sentences. Then time distributed like you do now on the last layer – Nassim Ben Feb 20, 2017 at 7:14

I understand the overarching concepts your explaining, however I'm unsure how to implement it. This is my first time experimenting with the LSTM. I tried reshaping the arrays, as I have seen others suggest, however I get errors from this, especially on the embedding layer which seems to only be able to accept a 2D input. If you could add a code example, I would really appreciate it. – DJK Feb 20, 2017 at 7:23