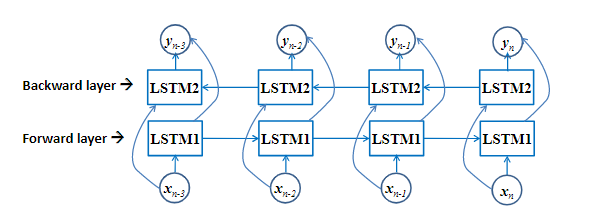
Bi-directional LSTM.



When a problem requires a reverse flow of information then it’s better to use this model.

Example : POS tagging (Part of speech) .

Example:

model = Sequential()  
model.add(Bidirectional(LSTM(10, return\_sequences=True), input\_shape=(5, 10)))  
model.add(Bidirectional(LSTM(10)))  
model.add(Dense(5))  
model.add(Activation('softmax'))  
model.compile(loss='categorical\_crossentropy', optimizer='rmsprop')  
  
 # With custom backward layer  
 model = Sequential()  
 forward\_layer = LSTM(10, return\_sequences=True)  
 backward\_layer = LSTM(10, activation='relu', return\_sequences=True,  
                       go\_backwards=True)  
 model.add(Bidirectional(forward\_layer, backward\_layer=backward\_layer,  
                         input\_shape=(5, 10)))  
 model.add(Dense(5))  
 model.add(Activation('softmax'))  
 model.compile(loss='categorical\_crossentropy', optimizer='rmsprop')

Example 2:

from keras.models import Sequential

from keras.layers import Activation, LSTM, Merge, TimeDistributedDense

from keras.optimizers import SGD

def fork (model, n=2):

forks = []

for i in range(n):

f = Sequential()

f.add (model)

forks.append(f)

return forks

# First bidirectional LSTM layer

forward = Sequential()

forward.add(LSTM(output\_dim=512, input\_shape=(50, 43), return\_sequences=True))

backward = Sequential()

backward.add(LSTM(output\_dim=512, input\_shape=(50, 43), return\_sequences=True, go\_backwards=True))

model = Sequential()

model.add(Merge([forward, backward], mode='concat'))

# Second bidirectionl LSTM layer

forward\_2, backward\_2 = fork(model)

forward\_2.add(LSTM(output\_dim=512, input\_shape=(50, 512), return\_sequences=True))

backward\_2.add(LSTM(output\_dim=512, input\_shape=(50, 512), return\_sequences=True, go\_backwards=True))

model = Sequential()

model.add(Merge([forward\_2, backward\_2], mode='concat'))

# Softmax decision layer

model.add(TimeDistributedDense(output\_dim=5))

model.add(Activation('softmax'))

# Optimizer function

sgd = SGD(lr=0.1, decay=1e-5, momentum=0.9, nesterov=True)

model.compile(loss='categorical\_crossentropy', optimizer=sgd)

print("Train...")

model.fit([X\_train, X\_train], Y\_train, batch\_size=1, nb\_epoch=nb\_epoches, validation\_data=([X\_test, X\_test], Y\_test), verbose=1, show\_accuracy=True)

Reference Link:

1: <https://www.tensorflow.org/api_docs/python/tf/keras/layers/Bidirectional>

2: <https://github.com/keras-team/keras/issues/1629>