## [Natural Language Processing in TensorFlow](https://www.coursera.org/learn/natural-language-processing-tensorflow/home/welcome)

**Week 1 - Sentiment in text**

This week they talked about word representations and the transformation of sentences in vectors (tokenization).

Below there is a snippet of getting some texts from JSON files and printing the tokenization.

!wget --no-check-certificate \ https://storage.googleapis.com/laurencemoroney-blog.appspot.com/sarcasm.json \

-O /tmp/sarcasm.json

import json

with open("/tmp/sarcasm.json", 'r') as f:

datastore = json.load(f)

sentences = []

labels = []

urls = []

for item in datastore:

sentences.append(item['headline'])

labels.append(item['is\_sarcastic'])

urls.append(item['article\_link'])

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

tokenizer = Tokenizer(oov\_token="<OOV>")

tokenizer.fit\_on\_texts(sentences)

word\_index = tokenizer.word\_index

print(len(word\_index))

29657

print(word\_index)

{'<OOV>': 1, 'to': 2, 'of': 3, 'the': 4, 'in': 5, 'for': 6, 'a': 7, 'on': 8, 'and': 9, 'with': 10, 'is': 11, 'new': 12,

sequences = tokenizer.texts\_to\_sequences(sentences)

padded = pad\_sequences(sequences, padding='post')

print(padded[0])

[ 0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0 0 0

0 0 0 0 308 15115 679 3337 2298 48 382 2576

15116 6 2577 8434]

print(padded.shape)

(26709, 40)

**Week 2 - Word Embeddings**

This week they talked about tokenizing sentences (and sentences to sequences) and also about word embedding, what it is and how they are created using neural networks.

Below there are some snippets on how to tokenize sentences.

vocab\_size = 1000

oov\_tok = '<OOV>'

max\_length = 120

trunc\_type = 'post'

tokenizer = Tokenizer(num\_words=vocab\_size, oov\_token=oov\_tok)

tokenizer.fit\_on\_texts(train\_sentences)

train\_sequences = tokenizer.texts\_to\_sequences(train\_sentences)

train\_padded = pad\_sequences(train\_sequences, maxlen=max\_length, truncating=trunc\_type)

Below there are some snippets on how to train the neural networks to obtain the word embeddings.

embedding\_dim = 16

model = tf.keras.Sequential([

tf.keras.layers.Embedding(vocab\_size, embedding\_dim, input\_length=max\_length),

tf.keras.layers.GlobalAveragePooling1D(),

tf.keras.layers.Dense(24, activation='relu'),

tf.keras.layers.Dense(6, activation='softmax')

])

model.compile(loss='sparse\_categorical\_crossentropy',optimizer='adam',metrics=['accuracy'])

num\_epochs = 30

history = model.fit(

Train\_padded,

training\_label\_seq,

epochs=num\_epochs,

validation\_data=(validation\_padded, validation\_label\_seq),

verbose=2

)

**Week 3 - Sequence models**

This week they talked about sequence models (LSTM, e.g.) and how those types of models can appropriately balance the importance of words in different parts of the sentence for it's meaning.

Below there is a snippet of a NN using dropout (to mitigate overfitting), convolution, pooling and LSTM.

model = tf.keras.Sequential([

tf.keras.layers.Embedding(

vocab\_size+1,

embedding\_dim,

input\_length=max\_length,

weights=[embeddings\_matrix],

trainable=False),

tf.keras.layers.Dropout(0.2),

tf.keras.layers.Conv1D(64, 5, activation='relu'),

tf.keras.layers.MaxPooling1D(pool\_size=4),

tf.keras.layers.LSTM(64),

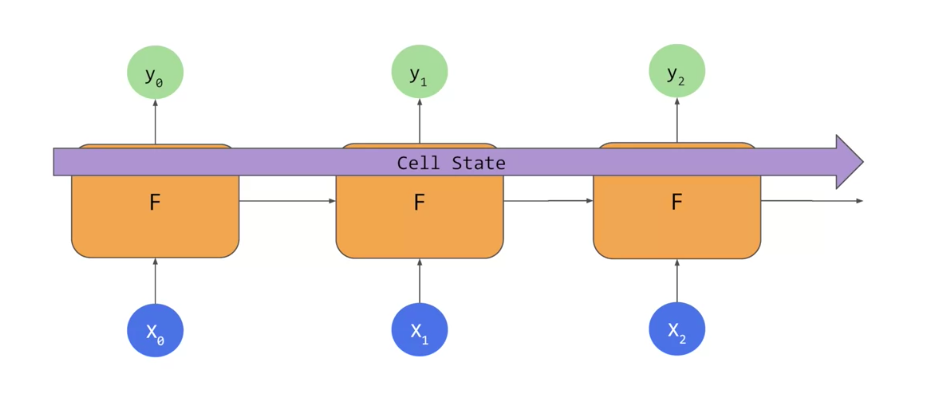
tf.keras.layers.Dense(1, activation='sigmoid')

])

model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy'])

model.summary()

The logic of context passing over cells in an LSTM neural network.



**Week 4 - Sequence models and literature**

This week they talked about sequence models and predicting next words in a given sentence.

In the snippet below there is the preprocessing of the data that will be the input of the model (xs and ys).

input\_sequences = []

for line in corpus:

token\_list = tokenizer.texts\_to\_sequences([line])[0]

for i in range(1, len(token\_list)):

n\_gram\_sequence = token\_list[:i+1]

input\_sequences.append(n\_gram\_sequence)

# pad sequences

max\_sequence\_len = max([len(x) for x in input\_sequences])

input\_sequences = np.array(pad\_sequences(input\_sequences, maxlen=max\_sequence\_len, padding='pre'))

# create predictors and label

xs, labels = input\_sequences[:,:-1],input\_sequences[:,-1]

ys = tf.keras.utils.to\_categorical(labels, num\_classes=total\_words)

The xs are all the words in the line but the last one, and, the ys are all the last words of each line.

In the snippet below there is the model structure and training.

model = Sequential()

model.add(Embedding(total\_words, 64, input\_length=max\_sequence\_len-1))

model.add(Bidirectional(LSTM(20)))

model.add(Dense(total\_words, activation='softmax'))

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

history = model.fit(xs, ys, epochs=500, verbose=1)

Note here that max\_sequence\_len is defined as the maximum length taking into account all lines/sentences inputted.

Lastly, in the snippet below there is the concatenation of the prediction, word by word, in the initial sentence.

seed\_text = "Laurence went to dublin"

next\_words = 100

for \_ in range(next\_words):

token\_list = tokenizer.texts\_to\_sequences([seed\_text])[0]

token\_list = pad\_sequences([token\_list], maxlen=max\_sequence\_len-1, padding='pre')

predicted = model.predict\_classes(token\_list, verbose=0)

output\_word = ""

for word, index in tokenizer.word\_index.items():

if index == predicted:

output\_word = word

break

seed\_text += " " + output\_word

print(seed\_text)

Laurence went to dublin dancing round merry the plenty as water water red rose together (AND SO ON)