# **Natural Language Processing**

# **Week-1 (Words based encodings using apis)**

Text

Description automatically generated

A picture containing text, device, gauge, meter

Description automatically generated

# Text to sequence

Text

Description automatically generated

Graphical user interface, text, application, chat or text message

Description automatically generated

Graphical user interface, text

Description automatically generated

00V (Out of Vocabulary)

Graphical user interface, text

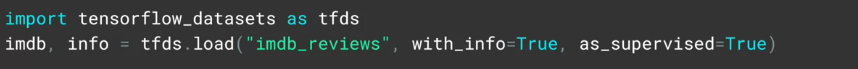
Description automatically generated

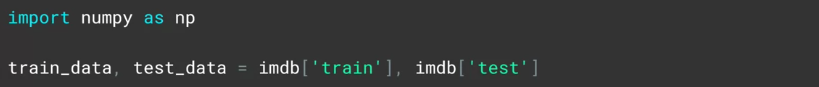
Graphical user interface, text

Description automatically generated

# **Week-2 (Word Embeddings)**

**Tensorflow datasets**



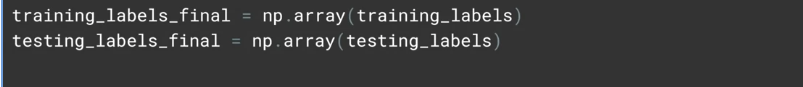


The values for S and I are tensors, so by calling their NumPy method, I'll actually extract their value.

Graphical user interface, text

Description automatically generated

When training, my labels are expected to be NumPy arrays. So I'll turn the list of labels that I've just created into NumPy arrays with this code.



Text

Description automatically generated

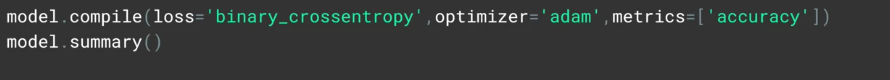
Graphical user interface, text

Description automatically generated

Alternatively, you can use a Global Average Pooling 1D like below, which averages across the vector to flatten it out. Over 10 epochs with global average pooling, I got an accuracy of 0.9664 on training and 0.8187 on test, taking about 6.2 seconds per epoch. With flatten, my accuracy was 1.0 and my validation about 0.83 taking about 6.5 seconds per epoch. So it was a little slower, but a bit more accurate.

Text

Description automatically generated



Text

Description automatically generated

Demonstrate the embeddings

Text

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Now it's time to write the vectors and their metadata auto files. The TensorFlow Projector reads this file type and uses it to plot the vectors in 3D space so we can visualize them. To the vectors file, we simply write out the value of each of the items in the array of embeddings, i.e, the co-efficient of each dimension on the vector for this word. To the metadata array, we just write out the words.

Text

Description automatically generated

To now render the results, go to the TensorFlow Embedding Projector on projector.tensorflow.org, press the ''Load data'' button on the left. You'll see a dialog asking you to load data from your computer. Use vector.TSV for the first one, and meta.TSV for the second. Once they're loaded, you should see something like this. Click this ''sphereize data'' checkbox on the top left, and you'll see the binary clustering of the data.

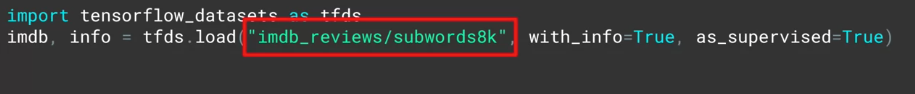
Graphical user interface, website, scatter chart

Description automatically generated

# Pre-tokenized datasets

We'll take a look at a version of the IMDb dataset that has been pre-tokenized for you, but the tokenization is done on sub words. We'll use that to demonstrate how text classification can have some unique issues, namely that the sequence of words can be just as important as their existence.

<https://github.com/tensorflow/datasets/blob/master/docs/catalog/imdb_reviews.md>





Graphical user interface, text

Description automatically generated

A screenshot of a computer

Description automatically generated with medium confidence

Text

Description automatically generated

Text

Description automatically generated

One thing to take into account, is the shape of the vectors coming from the tokenizer through the embedding, and it's not easily flattened. So we'll use Global Average Pooling 1D instead. Trying to flatten them, will cause a TensorFlow crash.

Graphical user interface, text

Description automatically generated

Text

Description automatically generated

Chart, line chart

Description automatically generated

While losses decreasing, it's decreasing in a very small way. So why do you think that might be? Well, the keys in the fact that we're using sub-words and not for-words, sub-word meanings are often nonsensical and it's only when we put them together in sequences that they have meaningful semantics. Thus, some way from learning from sequences would be a great way forward, and that's exactly what you're going to do with recurrent neural networks.

# **Week-3 (LSTMs)**

Diagram

Description automatically generated

Diagram

Description automatically generated

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

You can also stack LSTMs like any other keras layer by using code like below. But when you feed an LSTM into another one, you do have to put the return sequences equal true parameter into the first one. This ensures that the outputs of the LSTM match the desired inputs of the next one.

Text

Description automatically generated

Chart, line chart

Description automatically generated

Graphical user interface

Description automatically generated

# Using a convolutional network

Text

Description automatically generated

Chart, line chart

Description automatically generated

Text

Description automatically generated

IMDB Dataset –

Text

Description automatically generated

Graphical user interface, diagram

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generated with medium confidence

Graphical user interface, text

Description automatically generated

Graphical user interface, line chart

Description automatically generated

Text

Description automatically generated

Chart

Description automatically generated

# **Week-4 (Text Generation)**

Graphical user interface, text, application

Description automatically generated

Text

Description automatically generated

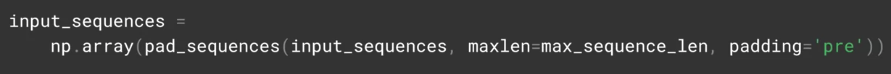
A picture containing text

Description automatically generated

Text

Description automatically generated with low confidence





Text

Description automatically generated with low confidence

Text

Description automatically generated

Text

Description automatically generated



A picture containing table

Description automatically generated

Text

Description automatically generated

We subtract one because we cropped off the last word of each sequence to get the label, so our sequences will be one less than the maximum sequence length.

Text

Description automatically generated

Let's take a look at what happens if we change the code to be bidirectional. By adding this line simply defining the LSTM is bidirectional, and then retraining, I can see that I do converge a bit quicker as you'll see in this chart.

# Predicting a word

Text

Description automatically generated

**Updated the model make it work better with a larger corpus of work**

Three things that you can experiment with. First, is the dimensionality of the embedding, 100 is purely arbitrary. Similarly, I increased the number of LSTN units to 150. Again, you can try different values or you can see how it behaves if you remove the bidirectional.

Text

Description automatically generated