**4.1 Introduction**

We've seen classification of text over the last few lessons. But what about if we want to generate new text. Now this might sound like new unbroken ground, but when you think about it, you've actually covered everything that you need to do this already. Instead of generating new text, how about thinking about it as **a prediction problem**. Remember when for example you had a bunch of pixels for a picture, and you trained a neural network to classify what those pixels were, and it would predict the contents of the image, like maybe a fashion item, or a piece of handwriting. Well, text prediction is very similar. We can get a body of texts, extract the full vocabulary from it, and then create datasets from that, where we make it phrase the Xs and the next word in that phrase to be the Ys. For example, consider the phrase, Twinkle, Twinkle, Little, Star. What if we were to create training data where the Xs are Twinkle, Twinkle, Little, and the Y is star. Then, whenever neural network sees the words Twinkle, Twinkle, Little, the predicted next word would be star. Thus given enough words in a corpus with a neural network trained on each of the phrases in that corpus, and the predicted next word, we can come up with some pretty sophisticated text generation and this week, you'll look at coding that.

**4.2 Looking into the code**

图片包含 文字, 屏幕截图

描述已自动生成

So let's start with a simple example. I've taken a traditional Irish song and here's the first few words of it, and here's the beginning of the code to process it.

图片包含 屏幕截图

描述已自动生成

In this case to keep things simple, I put the entire song into a single string. You can see that string here and I've denoted line breaks with \n. Then, by calling the split function on \n, I can create a Python list of sentences from the data and I'll convert all of that to lowercase. Using the tokenizer, I can then call fit\_on\_texts to this corpus of work and it will create the dictionary of words and the overall corpus. This is a key value pair with the key being the word and the value being the token for that word. We can find the total number of words in the corpus, by getting the length of its word index. We'll add one to this, to consider outer vocabulary words.

**4.3 Training the data**

图片包含 屏幕截图

描述已自动生成

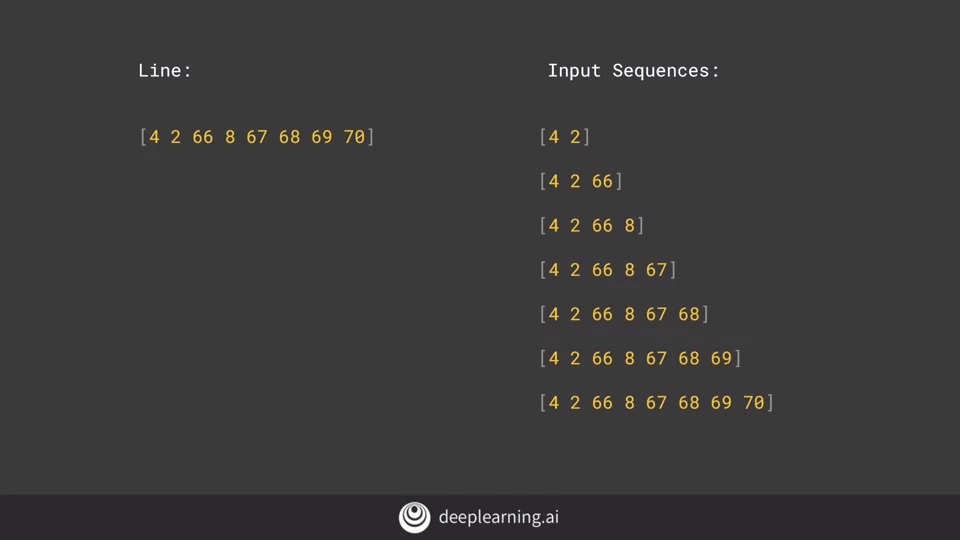
So now, let's look at the code to take this corpus and turn it into training data. Here's the beginning, I will unpack this line by line.

First of all, our training x's will be called input sequences, and this will be a Python list.

图片包含 屏幕截图

描述已自动生成

Then for each line in the corpus, we'll generate a token list using the tokenizers, texts to sequences method. This will convert a line of text like, "In the he town of Athy one Jeremy Lanigan," into a list of the tokens representing the words.



Then we'll iterate over this list of tokens and create a number of n-grams sequences, namely the first two words in the sentence or one sequence, then the first three are another sequence etc. The result of this will be, for the first line in the song, the following input sequences that will be generated. The same process will happen for each line, but as you can see, the input sequences are simply the sentences being broken down into phrases, the first two words, the first three words, etc.

图片包含 动物

描述已自动生成

We next need to find the length of the longest sentence in the corpus. To do this, we'll iterate over all of the sequences and find the longest one with code like this.

图片包含 动物

描述已自动生成

Once we have our longest sequence length, the next thing to do is pad all of the sequences so that they are the same length. We will pre-pad with zeros to make it easier to extract the label, you'll see that in a few moments.

图片包含 屏幕截图

描述已自动生成

So now, our line will be represented by a set of padded input sequences that looks like this. Now, that we have our sequences, the next thing we need to do is turn them into x's and y's, our input values and their labels.

图片包含 屏幕截图

描述已自动生成

When you think about it, now that the sentences are represented in this way, all we have to do is take all but the last character as the x and then use the last character as the y on our label. We do that like this, where for the first sequence, everything up to the four is our input and the two is our label. Similarly, here for the second sequence where the input is two words and the label is the third word, tokenized to 66. Here, the input is three words and the label is eight, which was the fourth word in the sentence. By this point, it should be clear why we did pre-padding, because it makes it much easier for us to get the label simply by grabbing the last token.



So now, we have to split our sequences into our x's and our y's. To do this, let's grab the first n tokens, and make them our x's. We'll then get the last token and make it our label. Before the label becomes a y, there's one more step, and you'll see that shortly.

Python makes this really easy to do with it's less syntax. So to get my x's, I just get all of the input sequences sliced to remove the last token. To get the labels, I get all of the input sequence sliced to keep the last token.

Now, I should one-hot encode my labels as this really is a classification problem. Where given a sequence of words, I can classify from the corpus, what the next word would likely be. So to one-hot encode, I can use the contrast utility to convert a list to a categorical. I simply give it the list of labels and the number of classes which is my number of words, and it will create a one-hot encoding of the labels.

图片包含 电子产品

描述已自动生成

So for example, if we consider this list of tokens as a sentence, then the x is the list up to the last value, and the label is the last value which in this case is 70. The y is a one-hot encoded array whether length is the size of the corpus of words and the value that is set to one is the one at the index of the label which in this case is the 70th element. Okay. You now have all of the data ready to train a network for prediction. Hopefully, this was useful for you. You'll see the neural network in the next video. But first, let's see your screen cast of processing the data, using the methods that you saw in this lesson.

4.4 Notebook for lesson 1

4.5 Finding what the next word should be

4.6 Example

4.7 Predicting a word

4.8 Poetryl

4.9 Looking into the code

4.10 Laurence the port

4.11 You next task