Ungraded Lab: Building a Vocabulary

In most natural language processing (NLP) tasks, the initial step in preparing your data is to extract a vocabulary of words from your corpus (i.e. input texts). You will need to define how to represent the texts into numeric features which can be used to train a neural network. Tensorflow and Keras makes it easy to generate these using its APIs. You will see how to do that in the next cells.

The code below takes a list of sentences, then takes each word in those sentences and assigns it to an integer. This is done using the TextVectorization() preprocessing layer and its adapt() method.

As mentioned in the docs above, this layer does several things including:

1. Standardizing each example. The default behavior is to lowercase and strip punctuation. See its `standardize` argument for other options.
2. Splitting each example into substrings. By default, it will split into words. See its `split` argument for other options.
3. Recombining substrings into tokens. See its `ngrams` argument for reference.
4. Indexing tokens.
5. Transforming each example using this index, either into a vector of ints or a dense float vector.

Run the cells below to see this in action.

In [1]:
```
import tensorflow as tf

# Sample inputs
sentences = [
    'i love my dog',
    'I, love my cat'
    ]

# Initialize the layer
vectorize_layer = tf.keras.layers.TextVectorization()

# Build the vocabulary
vectorize_layer.adapt(sentences)

# Get the vocabulary list. Ignore special tokens for now.
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
```

The resulting `vocabulary` will be a list where more frequently used words will have a lower index. By default, it will also reserve indices for special tokens but , for clarity, let's reserve that for later.

In [2]:
```
# Print the token index
for index, word in enumerate(vocabulary):
  print(index, word)
```

```
0 my
1 love
2 i
3 dog
4 cat
```

If you add another sentence, you'll notice new words in the vocabulary and new punctuation is still ignored as expected.

In [3]:
```
# Add another input
sentences = [
    'i love my dog',
    'I, love my cat',
    'You love my dog!'
]

# Initialize the layer
vectorize_layer = tf.keras.layers.TextVectorization()

# Build the vocabulary
vectorize_layer.adapt(sentences)

# Get the vocabulary list. Ignore special tokens for now.
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
```

In [4]:
```
# Print the token index
for index, word in enumerate(vocabulary):
  print(index, word)
```

```
0 my
1 love
2 i
3 dog
4 you
5 cat
```

Now that you see how it behaves, let's include the two special tokens. The first one at `0` is used for padding and `1` is used for out-of-vocabulary words. These are important when you use the layer to convert input texts to integer sequences. You'll see that in the next lab.

In [5]:
```
# Get the vocabulary list.
vocabulary = vectorize_layer.get_vocabulary()

# Print the token index
for index, word in enumerate(vocabulary):
  print(index, word)
```

```
0
1 [UNK]
2 my
3 love
4 i
5 dog
6 you
7 cat
```

That concludes this short exercise on building a vocabulary!