Open in Colab

# Tokenizer Basics

In most 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 numerical representations which can be used to train a neural network. These representations are called *tokens* and Tensorflow and Keras makes it easy to generate these using its APIs. You will see how to do that in the next cells.

## Generating the vocabulary

In this notebook, you will look first at how you can provide a look up dictionary for each word. 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 fit_on_texts() method and you can get the result by looking at the `word_index` property. More frequent words have a lower index.

In [1]:

```python
from tensorflow.keras.preprocessing.text import Tokenizer

# Define input sentences
sentences = [
    'i love my dog',
    'I, love my cat'
    ]

# Initialize the Tokenizer class
tokenizer = Tokenizer(num_words = 100)

# Generate indices for each word in the corpus
tokenizer.fit_on_texts(sentences)

# Get the indices and print it
word_index = tokenizer.word_index
print(word_index)
```

```
{'i': 1, 'love': 2, 'my': 3, 'dog': 4, 'cat': 5}
```

The `num_words` parameter used in the initializer specifies the maximum number of words minus one (based on frequency) to keep when generating sequences. You will see this in a later exercise. For now, the important thing to note is it does not affect how the `word_index` dictionary is generated. You can try passing `1` instead of `100` as shown on the next cell and you will arrive at the same `word_index`.

Also notice that by default, all punctuation is ignored and words are converted to lower case. You can override these behaviors by modifying the `filters` and `lower` arguments of the `Tokenizer` class as described here. You can try modifying these in the next cell below and compare the output to the one generated above.

In [2]:

```python
# Define input sentences
sentences = [
    'i love my dog',
    'I, love my cat',
    'You love my dog!'
]

# Initialize the Tokenizer class
tokenizer = Tokenizer(num_words = 1)

# Generate indices for each word in the corpus
tokenizer.fit_on_texts(sentences)

# Get the indices and print it
word_index = tokenizer.word_index
print(word_index)  # As we see only unique words are indexed
```

```
{'love': 1, 'my': 2, 'i': 3, 'dog': 4, 'cat': 5, 'you': 6}
```

That concludes this short exercise on tokenizing input texts!