1. **Semana 1**

- **Processamento de linguagem natural com tensorflow**
- Há diversas formas, como usar tokens numerar as imagens e capturar os valores delas, como com a biblioteca tokenizer, que coloca cada palavra em um dicionário. Tokenizer já considera coisas como maiúsculas/minusculas e pontuação no final das palavras

```
from tensorflow.keras.preprocessing.text import Tokeni

# 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)
```

-
-

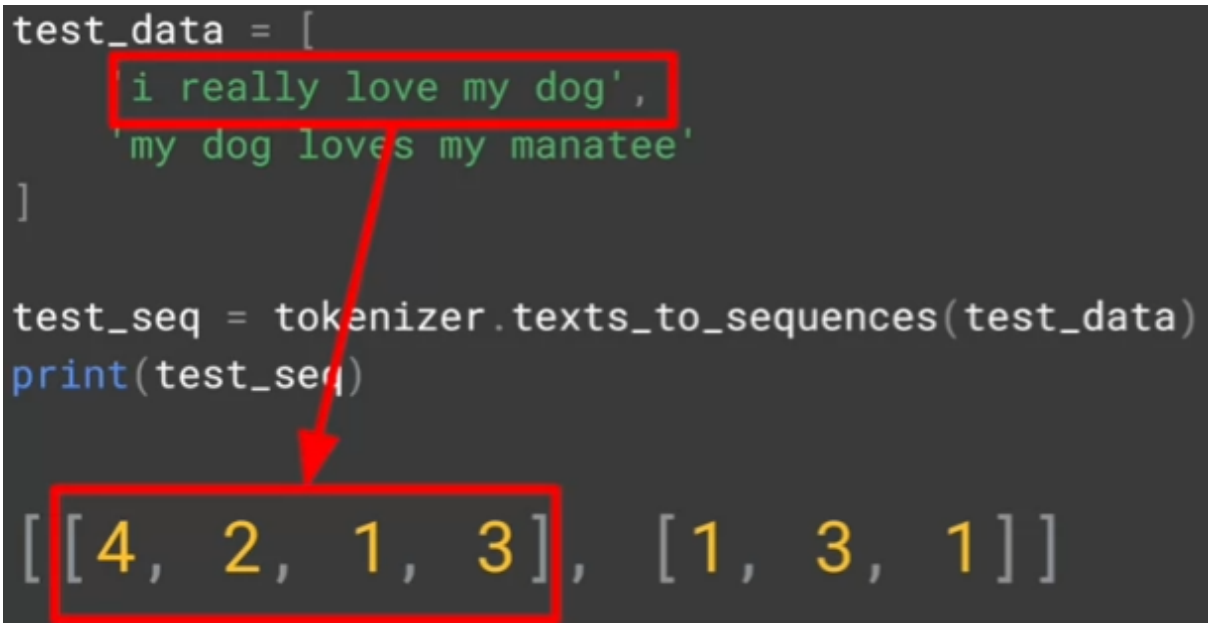
```
{'i': 1, 'love': 2, 'my': 3, 'dog': 4, 'cat': 5}
```

-

```
tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)
```

- Sequences transforma a string em uma lista de ints

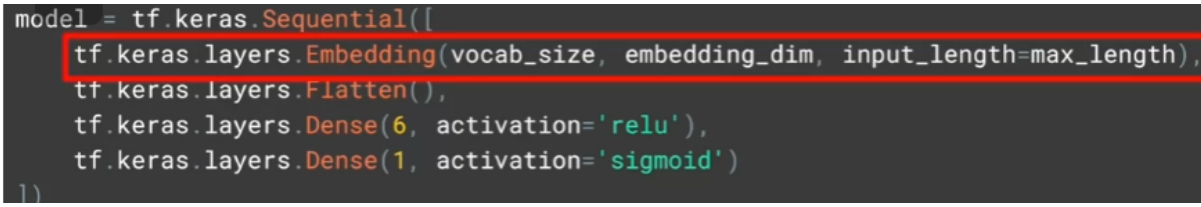


- **Padding**: transforma a lista de sequências em uma matriz em que cada linha tem o da tamanho, o tamanho da maior (ou um valor arbitrário) e coloca 0 quando não tem a palavra

-

2. Semana 2

- **Embeddings:** palavras similares, com o mesmo significado são colocadas em um cluster multidimensional de modo que apontam ao mesmo local, por exemplo, cão e canino. Isso começa a dar sentimento e sentido as palavras.



- É colocado dentro do modelo.

3. Semana 3

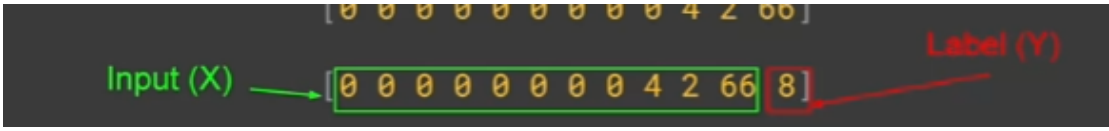
- Sequências para o entendimento de linguagens naturais é fundamental, ajuda a entender principalmente o contexto da onde a palavra vem e o significado. LSTM (long short termn memory) é uma

arquitetura que tme uma memória e ajuda a lembrar o que foi dito antes em uma sequência

```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(tokenizer.vocab_size, 64),
    tf.keras.layers.Bidirectional(tf.keras.layers.LSTM(64)),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

- Além disso, pode se usar redes convolucionais, tudo depende do contexto e do overfitting que isso ira gerar

4. **<u>Semana 4</u>**
    - Text generation: e se tratessemos como um problema de predição? em que o modelo tenta-se adivinhar a a pr[oxima palavra de um texto, por exemplo



    -
    -