DEEPLEARNING

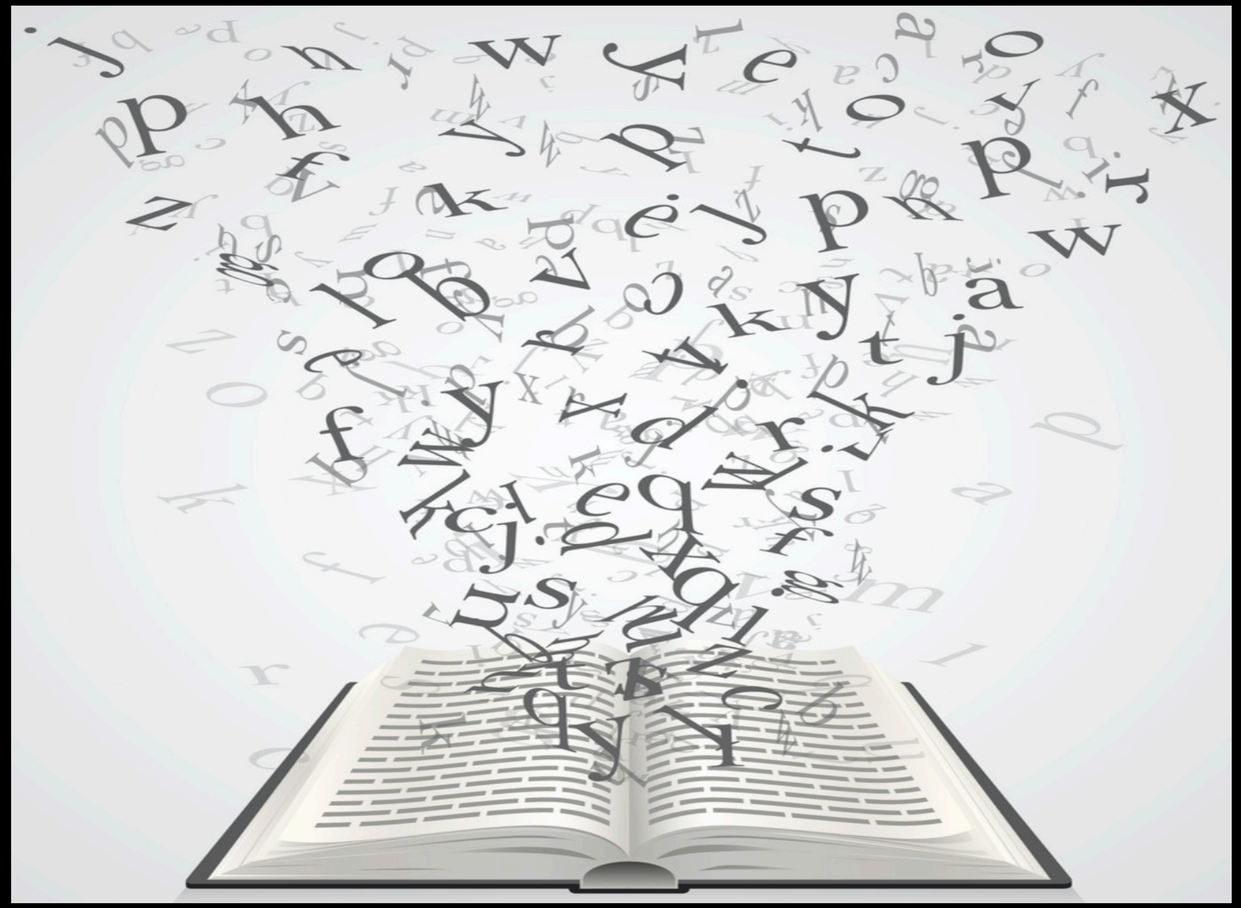
TENSORFLOW

https://www.tensorflow.org

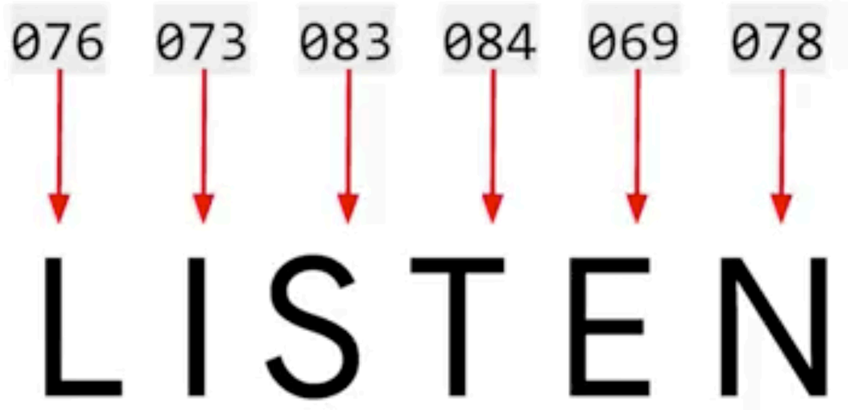
Rafael Vieira Coelho

TENSORFLOW

USO DE TEXTOS

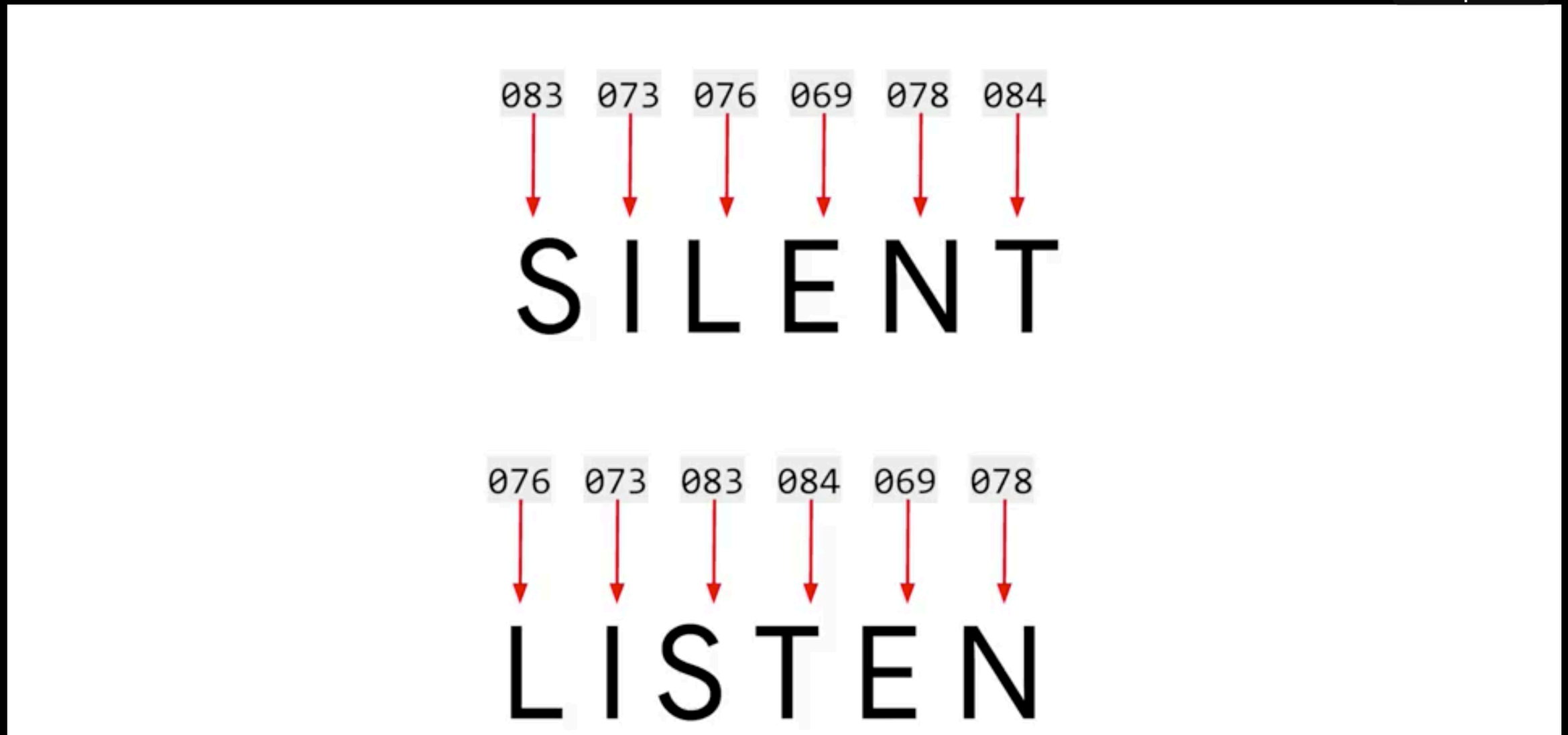


PODERÍAMOS USAR O CÓDIGO ASCII PARA REPRESENTAR AS LETRAS

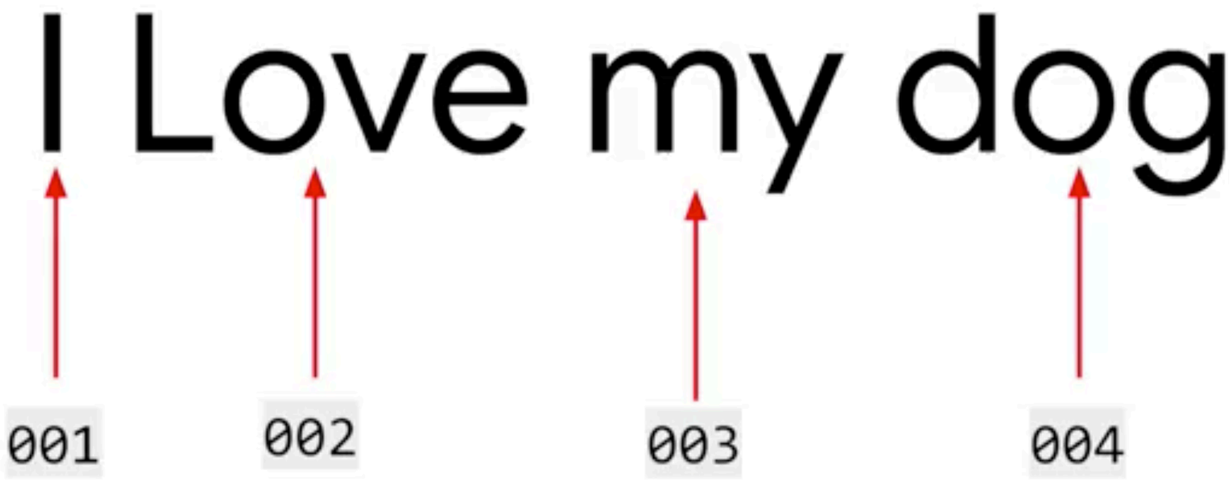


CÓDIGO ASCII

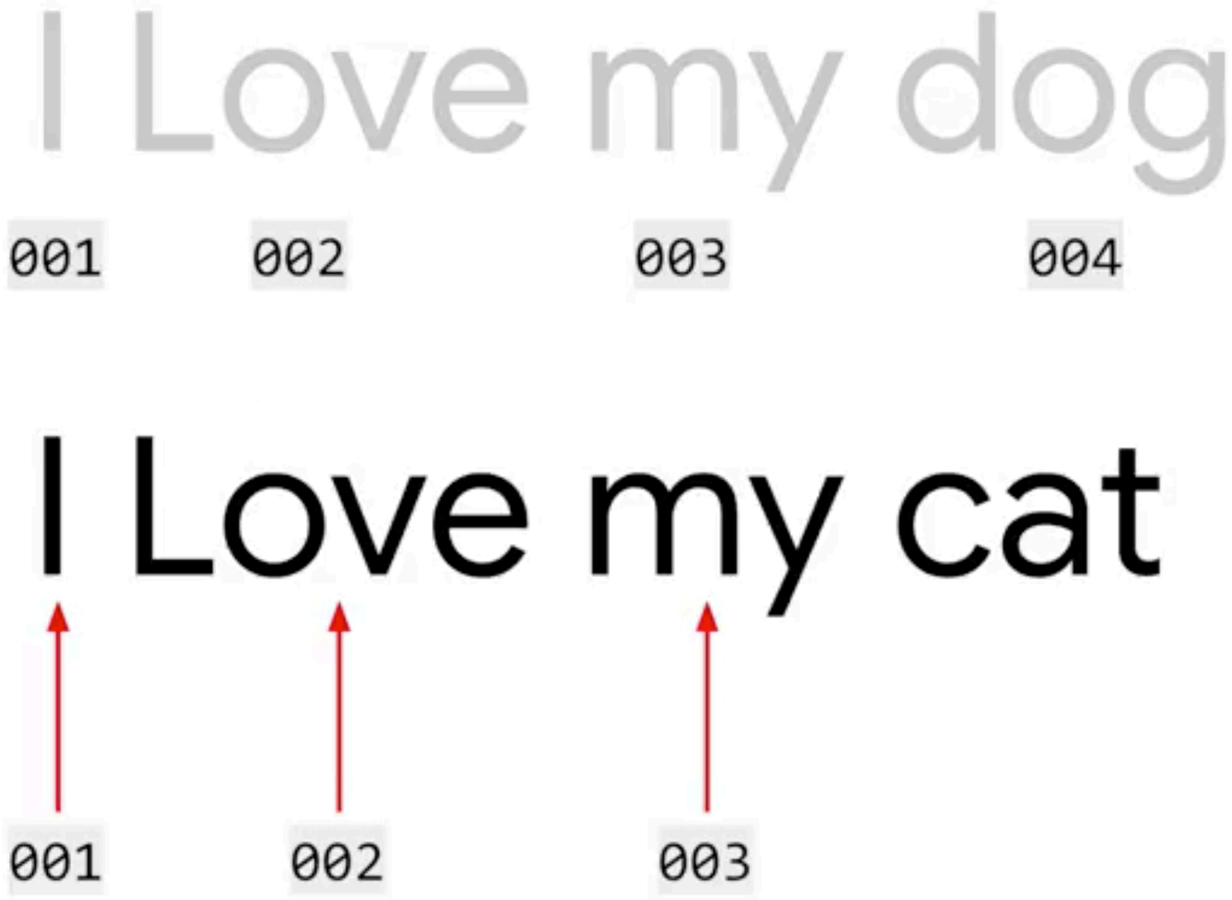
Mas podemos obter os mesmos números com outra palavra.



QUE TAL USAR UM NÚMERO PARA CADA PALAVRA?



PODEMOS ASSIM REUTILIZAR AS PALAVRAS



E USAR UM NOVO NÚMERO PARA CADA PALAVRA NOVA

I Love my dog

001 002 003 004

I Love my cat

001 002 003 005

A) USANDO AS APIS PARA MANIPULAÇÃO DE TEXTOS


```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat'
]

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

1) IMPORTAR O TOKENIZER

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat'
]

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

2) CRIAR O ARRAYS DE SENTENÇAS

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat'
]

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

3) INSTANCIAR UM OBJETO TOKENIZER (MÁX. 100 PALAVRAS)

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat'
]

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

4) CODIFICAR AS PALAVRAS DAS SENTENÇAS EM NÚMEROS

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat'
]

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

5) OBTER O DICIONÁRIO (PALAVRA<->NÚMERO)

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing.text import Tokenizer

sentences = [
    'I love my dog',
    'I love my cat'
]

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

DICIONÁRIO DE CODIFICAÇÃO

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

ADICIONANDO UMA NOVA FRASE...

```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!'
]
```

TEMOS UMA NOVA PALAVRA POR CAUSA DO ‘!'?!

```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!'
]
```

NÃO!!

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

E TEMOS UMA NOVA PALAVRA!

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

B) CRIANDO SEQUENCIAS DE SENTENÇAS

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

sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
]

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

sequences = tokenizer.texts_to_sequences(sentences)

print(word_index)
print(sequences)
```

```
{'my': 1, 'love': 2, 'dog': 3, 'i': 4, 'you': 5, 'cat': 6, 'do': 7, 'think': 8, 'is': 9, 'amazing': 10}
[[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7, 5, 8, 1, 3, 9, 10]]
```

E O QUE VOCÊS ACHAM QUE ACONTECE NESTE EXEMPLO?

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]

test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

ACABAMOS PERDENDO INFORMAÇÃO NESTE CASO...

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]

test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

```
[[4, 2, 1, 3], [1, 3, 1]]
```

```
{'think': 8, 'amazing': 10, 'my': 1, 'love': 2, 'dog': 3, 'is': 9, 'you': 5, 'do': 7,
'cat': 6, 'i': 4}
```

A PRIMEIRA SENTENÇA ESTÁ OK!

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]

test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

```
[[4, 2, 1, 3], [1, 3, 1]]
```

```
{'think': 8, 'amazing': 10, 'my': 1, 'love': 2, 'dog': 3, 'is': 9, 'you': 5, 'do': 7,
'cat': 6, 'i': 4}
```

JÁ A SEGUNDA...

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]

test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

```
[[4, 2, 1, 3], [1, 3, 1]]
```

```
{'think': 8, 'amazing': 10, 'my': 1, 'love': 2, 'dog': 3, 'is': 9, 'you': 5, 'do': 7,
'cat': 6, 'i': 4}
```

C) USANDO OOV (OUT OF VOCABULARY)

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

sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
]

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)

test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]

test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

really loves manatee

```
[[5, 1, 3, 2, 4], [2, 4, 1, 2, 1]]
```

```
{'think': 9, 'amazing': 11, 'dog': 4, 'do': 8, 'i': 5, 'cat': 7,
 'you': 6, 'love': 3, '<OOV>': 1, 'my': 2, 'is': 10}
```

D) NORMALIZAR O TAMANHO DAS SENTENÇAS (PADDING)

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
]

tokenizer = Tokenizer(num_words = 100, oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)

padded = pad_sequences(sequences)
print(word_index)
print(sequences)
print(padded)
```

frase maior

```
{'OOV': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}
[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]
[[ 0  0  0  5  3  2  4]
 [ 0  0  0  5  3  2  7]
 [ 0  0  0  6  3  2  4]
 [ 8  6  9  2  4 10 11]]
```

padding

frase maior

PODEMOS COLOCAR O QUE IREMOS PERDER PARA O FINAL DA SENTENÇA

```
padded = pad_sequences(sequences, padding='post')
```

```
{'OOV': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}
[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]
[[ 5  3  2  4  0  0  0]
 [ 5  3  2  7  0  0  0]
 [ 6  3  2  4  0  0  0]
 [ 8  6  9  2  4 10 11]]
```

jogando o padding para o final (post)

PODEMOS LIMITAR O NÚMERO MÁXIMO DE PALAVRAS POR SENTENÇA

No exemplo, restringimos a no máximo 5 palavras.

```
padded = pad_sequences(sequences, padding='post', maxlen=5)
```

```
{'OOV': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}
[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]
[[ 5  3  2  4  0]
 [ 5  3  2  7  0]
 [ 6  3  2  4  0]
 [ 9  2  4 10 11]]
```

restringindo o tamanho máximo, diminuímos o padding

mas perdemos informação em frases maiores que 5

no máximo 5

E PODEMOS DEFINIR ONDE QUEREMOS PERDER NA HORA DE RESTRINGIR UMA SENTENÇA

Neste caso, estamos truncando no final da sentença. Olhem a última sentença:

“Do[8] you[6] think[9] my[2] dog[4] is[10] amazing[11]?”

```
padded = pad_sequences(sequences, padding='post',
                       truncating='post', maxlen=5)
```

```
{'OOV': 1, 'my': 2, 'love': 3, 'dog': 4, 'i': 5, 'you': 6, 'cat': 7, 'do': 8, 'think': 9, 'is': 10, 'amazing': 11}
[[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]
[[5 3 2 4 0]
 [5 3 2 7 0]
 [6 3 2 4 0]
 [8 6 9 2 4]]
```

truncamento no final da sentença (antes era [9 2 4 10 11])

C) USANDO UMA BASE DE DADOS REAL COM TEXTO

Usaremos esta base de dados para a detecção de sarcasmo no texto.

Sarcasm in News Headlines Dataset by Rishabh Misra

https://rishabhmisra.github.io/publications/

https://www.kaggle.com/rmisra/news-headlines-dataset-for-sarcasm-detection/home

AS INFORMAÇÕES CONTIDAS NO ARQUIVO

is_sarcastic: 1 if the record is sarcastic otherwise 0

headline: the headline of the news article

article_link: link to the original news article. Useful in collecting supplementary data

ALTERAREMOS O ARQUIVO JSON PARA O FORMATO QUE A API JSON CONSIGA LER O DOCUMENTO


```
{"article_link":
"https://politics.theonion.com/boehner-just-wants-wife-to-listen-not-come-up-with-alt-18195
74302", "headline": "boehner just wants wife to listen, not come up with alternative
debt-reduction ideas", "is_sarcastic": 1}

{"article_link":
"https://www.huffingtonpost.com/entry/roseanne-revival-review_us_5ab3a497e4b054d118e04365",
"headline": "the 'roseanne' revival catches up to our thorny political mood, for better and
worse", "is_sarcastic": 0}

{"article_link":
"https://local.theonion.com/mom-starting-to-fear-son-s-web-series-closest-thing-she-1819576
697", "headline": "mom starting to fear son's web series closest thing she will have to
grandchild", "is_sarcastic": 1}
```



```
[

{"article_link":
"https://politics.theonion.com/boehner-just-wants-wife-to-listen-not-come-up-with-alt-18195
74302", "headline": "boehner just wants wife to listen, not come up with alternative
debt-reduction ideas", "is_sarcastic": 1},

{"article_link":
"https://www.huffingtonpost.com/entry/roseanne-revival-review_us_5ab3a497e4b054d118e04365",
"headline": "the 'roseanne' revival catches up to our thorny political mood, for better and
worse", "is_sarcastic": 0},

{"article_link":
"https://local.theonion.com/mom-starting-to-fear-son-s-web-series-closest-thing-she-1819576
697", "headline": "mom starting to fear son's web series closest thing she will have to
grandchild", "is_sarcastic": 1}

]
```

COMO CARREGAR OS DADOS VIA CÓDIGO

```python
import json

with open("sarcasm.json", 'r') as f:
    datastore = json.load(f)

sentences = []
labels = []
urls = []
for item in datastore:
    sentences.append(item['headline'])
    labels.append(item['is_sarcastic'])
    urls.append(item['article_link'])
```

Sarcasm_Headli…Dataset_v2.json ×


```
1  {"is_sarcastic": 1, "headline": "thirtysomething scientists unveil doomsday clock of hair loss", "article_link": "http
2  {"is_sarcastic": 0, "headline": "dem rep. totally nails why congress is falling short on gender, racial equality", "ar
3  {"is_sarcastic": 0, "headline": "eat your veggies: 9 deliciously different recipes", "article_link": "https://www.huff
4  {"is_sarcastic": 1, "headline": "inclement weather prevents liar from getting to work", "article_link": "https://local
5  {"is_sarcastic": 1, "headline": "mother comes pretty close to using word 'streaming' correctly", "article_link": "http
6  {"is_sarcastic": 0, "headline": "my white inheritance", "article_link": "https://www.huffingtonpost.com/entry/my-white
7  {"is_sarcastic": 0, "headline": "5 ways to file your taxes with less stress", "article_link": "https://www.huffingtonp
8  {"is_sarcastic": 1, "headline": "richard branson's global-warming donation nearly as much as cost of failed balloon tr
9  {"is_sarcastic": 1, "headline": "shadow government getting too large to meet in marriott conference room b", "article_
10 {"is_sarcastic": 0, "headline": "lots of parents know this scenario", "article_link": "https://www.huffingtonpost.comh
```

LINKS ÚTEIS

- https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%201%20-%20Lesson%201.ipynb

- https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%201%20-%20Lesson%202.ipynb#scrollTo=rX8mhOLljYeM

- https://rishabhmisra.github.io/publications/

- https://colab.research.google.com/github/lmoroney/dlaicourse/blob/master/TensorFlow%20In%20Practice/Course%203%20-%20NLP/Course%203%20-%20Week%201%20-%20Lesson%203.ipynb

EXERCÍCIO

Neste exercício, iremos explorar o arquivo da BBC News.

Acesse o link BBC text archive.

Você precisa transformar o texto obtido em tokens.

Você deve desconsiderar palavras não-relevantes. Exemplos dessas palavras estão no link here.

DÚVIDAS?

RAFAELVC2@GMAIL.COM