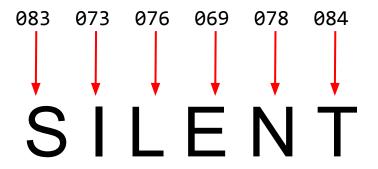
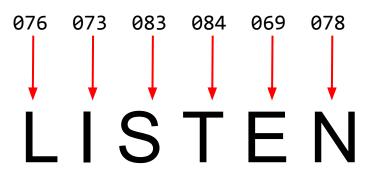
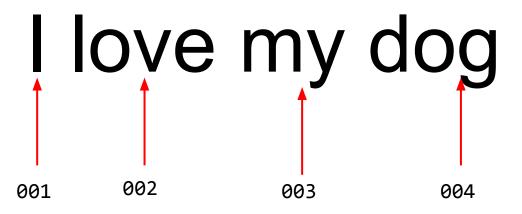
#### LISTEN

# 076 073 083 084 069 078 L I S T E N





001



001 002 003 004

#### I love my cat

love my cat



001	002	003	004
001	002	003	005

```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
print(vocabulary)
```



```
import tensorflow as tf
sentences =
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
print(vocabulary)
```

```
import tensorflow as tf
sentences =
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
print(vocabulary)
```



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
print(vocabulary)
```



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
print(vocabulary)
```



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False)
print(vocabulary)
```



['my', 'love', 'i', 'dog', 'cat']



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

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

['my', 'love', 'i', 'dog', 'you', 'cat']



['my', 'love', 'i', <mark>'dog'</mark>, 'you', 'cat']



['my', 'love', 'i', 'dog', 'you', 'cat']



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary(include_special_tokens=False
print(vocabulary)
```



```
import tensorflow as tf
sentences = [
   'I love my dog',
    'I love my cat'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
print(vocabulary)
```

```
['', '[UNK]', 'my', 'love', 'i', 'dog', 'you', 'cat']
```



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
for index, word in enumerate(vocabulary):
    print(index, word)
```



```
import tensorflow as tf
sentences =
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
for index, word in enumerate(vocabulary):
    print(index, word)
```

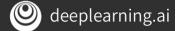


```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
for index, word in enumerate(vocabulary):
    print(index, word)
```



```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
```

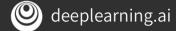
```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sequence = vectorize_layer('I love my dog')
for index, word in enumerate(vocabulary):
    print(index, word)
print(sequence)
```



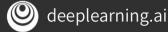
```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(sequence)
tf.Tensor([6 3 2 4], shape=(4,), dtype=int64)
```



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sequences = vectorize_layer(sentences)
for index, word in enumerate(vocabulary):
    print(index, word)
print(sequences)
```



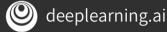
```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(sequences)
tf.Tensor(
[[6 3 2 4 0 0 0]
 [63210000]
 [5 \ 3 \ 2 \ 4 \ 0 \ 0 \ 0]
 [ 9 5 7 2 4 8 11]], shape=(4, 7), dtype=int64)
```



```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(sequences)
tf.Tensor(
                 0 0]
     3 2 10
              0 0 0]
          4
  5
                 0 0]
     3 2
              0
     5 7 2 4 8 11]], shape=(4, 7), dtype=int64)
```



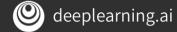
```
for index, word in enumerate(vocabulary):
   print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(sequences)
tf.Tensor(
[[6324000
 [63210000]
 5 3 2 4
             0 0 0]
 [ 9 5 7 2 4 8 11]], shape=(4, 7), dtype=int64)
```



```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(sequences)
tf.Tensor(
[[6 3 2 4 0 0 0]
  6 3 2 10
           2 4 8 11 ], shape=(4, 7), dtype=int64)
```

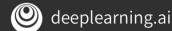


```
sentences = [
   'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sequences = vectorize_layer(sentences)
print(sequences)
tf.Tensor(
  6 3 2 4 0 0 0]
  6 3 2 10 0 0 0]
     5 7 2 4 8 11]], shape=(4, 7), dtype=int64)
```



```
sentences = [
   'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
```

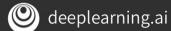
```
sentences = [
   'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
```



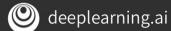
```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
print(sequences)
<_MapDataset element_spec=TensorSpec(shape=(None,), dtype=tf.int64, name=None)>
```



```
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
print(sequences)
<_MapDataset element_spec=TensorSpec(shape=(None,), dtype=tf.int64, name=None)>
```



```
sentences = [
   'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
for sentence, sequence in zip(sentences, sequences):
    print(f'{sentence} ---> {sequence}')
```



```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
for sentence, sequence in zip(sentences, sequences):
    print(f'{sentence} ---> {sequence}')
I love my dog ---> [6 3 2 4]
I love my cat ---> [6 3 2 10]
You love my dog! ---> [5 3 2 4]
Do you think my dog is amazing? ---> [9 5 7 2 4 8 11]
```

```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
for sentence, sequence in zip(sentences, sequences):
    print(f'{sentence} ---> {sequence}')
I love my dog ---> [6 3 2 4]
I love my cat ---> [6 3 2 10]
You love my dog! ---> [5 3 2 4]
Do you think my dog is amazing? ---> [9 5 7 2 4 8 11]
```

```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
sequences_pre = tf.keras.utils.pad_sequences(sequences, padding='pre')
print(sequences_pre)
[[ 0 0 0 6 3 2 4]
[ 0 0 0 6 3 2 10]
[ 0 0 0 5 3 2 4]
[ 9 5 7 2 4 8 11]]
```

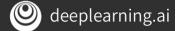


```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
sequences_pre = tf.keras.utils.pad_sequences(sequences, padding='pre')
print(sequences_pre)
  0 0 0 6 3 2 10]
0 0 0 5 3 2 4]
  <u>9 5</u> 7 2 4 8 11]]
```



```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
sentences_dataset = tf.data.Dataset.from_tensor_slices(sentences)
sequences = sentences_dataset.map(vectorize_layer)
sequences_pre = tf.keras.utils.pad_sequences(sequences, padding='pre')
print(sequences_pre)
[[0006324]
[ 0 0 0 6 3 2 10]
[ 0 0 0 5 3 2 4]
[ 9 5 7 2 4 8 11]]
```

```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization(ragged=True)
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
ragged_sequences = vectorize_layer(sentences)
for index, word in enumerate(vocabulary):
    print(index, word)
print(ragged_sequences)
```



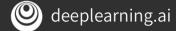
```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(ragged_sequences)
<tf.RaggedTensor [[6, 3, 2, 4], [6, 3, 2, 10], [5, 3, 2, 4], [9, 5, 7, 2, 4, 8, 11]]>
```



```
for index, word in enumerate(vocabulary):
    print(index, word)
0
1 [UNK]
2 my
3 love
4 dog
5 you
6 i
7 think
8 is
9 do
10 cat
11 amazing
print(ragged_sequences)
<tf.RaggedTensor [[6, 3, 2, 4] [6, 3, 2, 10], [5, 3, 2, 4], [9, 5, 7, 2, 4, 8, 11] >
```



```
import tensorflow as tf
sentences = [
    'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization(ragged=True)
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
ragged_sequences = vectorize_layer(sentences)
pre_padded_sequences = tf.keras.utils.pad_sequences(ragged_sequences.numpy())
print(pre_padded_sequences)
```



```
print(pre_padded_sequences)

[[ 0  0  0  6  3  2  4]
  [ 0  0  0  6  3  2  10]
  [ 0  0  0  5  3  2  4]
  [ 9  5  7  2  4  8  11]]
```

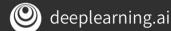
```
sentences = [
   'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
```

```
sentences = [
   'I love my dog',
    'I love my cat',
    'You love my dog!',
    'Do you think my dog is amazing?'
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
test_data =
    'i really love my dog',
    'my dog loves my manatee'
```

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

test_seq = vectorize_layer(test_data)
print(test_seq)
```

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
test_seq = vectorize_layer(test_data)
print(test_seq)
tf.Tensor(
[[6 1 3 2 4]
[2 \ 4 \ 1 \ 2 \ 1]], shape=(2, 5), dtype=int64)
```



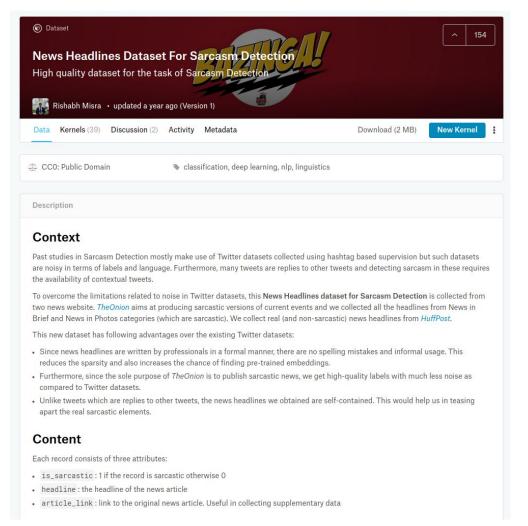
```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
test_seq = vectorize_layer(test_data)
                                                                    0
print(test_seq)
                                                                    1 [UNK]
                                                                    2 my
                                                                    3 love
                                                                    4 dog
tf.Tensor(
                                                                    5 you
[[6 1 3 2 4]
                                                                    6 i
 [2 \ 4 \ 1 \ 2 \ 1]], shape=(2, 5), dtype=int64)
                                                                    7 think
                                                                    8 is
                                                                    9 do
                                                                    10 cat
                                                                    11 amazing
```





Sarcasm in News Headlines Dataset by Rishabh Misra

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



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

```
"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": "bttps://www.buffingtonpost.com/optny/poscenne.povivel_poviou.us_Fab2a407a4b0F4d118a0426F"
```

{"article link":

grandchild", "is sarcastic": 1}

"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}

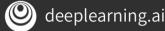
{"article\_link":

"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

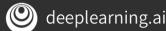
```
{"article_link":
"https://politics.theonion.com/boehner-just-wants-wife-to-listen-not-come-up-with-alt-18195
74302", "headline": "boehner just wants wife to listen, not come up with alternative
debt-reduction ideas", "is_sarcastic": 1},
{"article_link":
"https://www.huffingtonpost.com/entry/roseanne-revival-review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

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

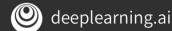
sentences = []
labels = []
urls = []
for item in datastore:
    sentences.append(item['headline'])
    labels.append(item['is_sarcastic'])
    urls.append(item['article_link'])
```



```
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'])
```



```
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'])
```



```
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'])
```



```
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'])
```



```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
post_padded_sequences = vectorize_layer(sentences)
print(f'padded sequence: {post_padded_sequences[2]}')
padded sequence: [140 825 2 813 1100 2048 571 5057 199 139 39 46 2
```

import tensorflow as tf



```
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
post_padded_sequences = vectorize_layer(sentences)
print(f'padded sequence: {post_padded_sequences[2]}')
padded sequence: [140 825 2 813 1100 2048 571 5057 199 139 39 46 2
```

import tensorflow as tf

```
import tensorflow as tf
vectorize_layer = tf.keras.layers.TextVectorization()
vectorize_layer.adapt(sentences)
vocabulary = vectorize_layer.get_vocabulary()
post_padded_sequences = vectorize_layer(sentences)
print(f'padded sequence: {post_padded_sequences[2]}')
padded sequence: [140 825 2 813 1100 2048 571 5057 199 139 39 46 2
```

