#### Copyright Notice

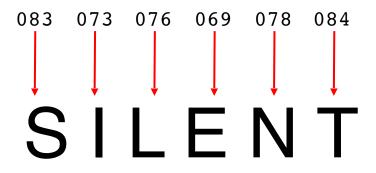
These slides are distributed under the Creative Commons License.

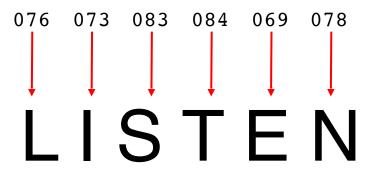
<u>DeepLearning.Al</u> makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite <u>DeepLearning.Al</u> as the source of the slides.

For the rest of the details of the license, see <a href="https://creativecommons.org/licenses/by-sa/2.0/legalcode">https://creativecommons.org/licenses/by-sa/2.0/legalcode</a>

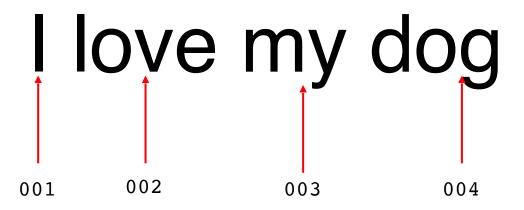
#### LISTEN

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





001



#### I love my cat

love my cat



001	002	003	004
001	002	003	005

```
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)

```
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)
```

import tensorflow as tf

from tensorflow import keras

```
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)
```

```
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print(word_index)
```

```
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)
iokenizer.iii_on_iexis(seniences)
```

word\_index = tokenizer.word\_index

print(word\_index)

```
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)

```
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
```

PHILL WOLG\_HIGGA

```
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)
```

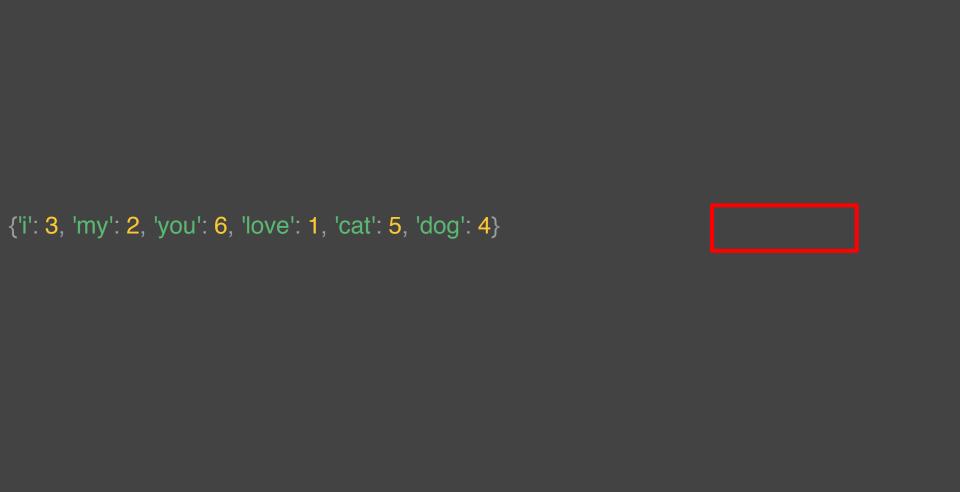
import tensorflow as tf

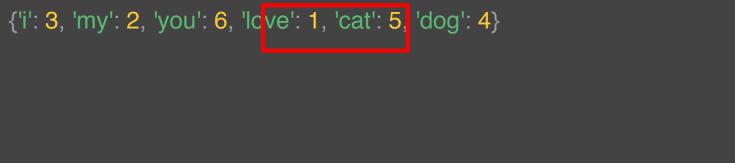
{'i': 1, 'my': 3, 'dog': 4, 'cat': 5, 'love': 2}

```
sentences = [
  'I love my dog',
  'I love my cat',
  'You love my dog!'
1
```

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

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





```
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)
```

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)

```
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)
```

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

[[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7, 5, 8, 1, 3, 9, 10]]

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

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

```
[[4, 2, 1, 3], [4, 2, 1, 6], [5, 2, 1, 3], [7, 5, 8, 1, 3, 9, 10]]
```

```
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)
WULU_ILIUGA — LUNGHIZGI.WULU_ILIUGA
sequences = tokenizer.texts_to_sequences(sentences)
print(word_index)
print(sequences)
```

```
test_data = [
    'i really love my dog',
    'my dog loves my manatee'
]
test_seq = tokenizer.texts_to_sequences(test_data)
print(test_seq)
```

```
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}

```
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}

```
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}

```
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_seg = tokenizer.texts_to_seguences(test_data)
print(test_seq)
```

```
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_seg = tokenizer.texts_to_seguences(test_data)
print(test_seq)
```

[[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}
```

```
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)
```

```
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)
```

```
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)
MILLIAMOLA ILIAOA
print(sequences)
print(padded)
```

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

[[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]]

[[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 5 2 4]
[8 6 9 2 4 10 11]]

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

{'do': 8, 'you': 6, 'love': 3, 'i': 5, 'amazing': 11, 'my': 2, 'is': 10, 'think': 9, 'dog': 4, '<OOV>': 1, 'cat': 7} [[5, 3, 2, 4], [5, 3, 2, 7], [6, 3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]  $[[0\ 0\ 0\ 5\ 3\ 2\ 4]$ [0005327] [0006324] [8 6 9 2 4 10 11]]

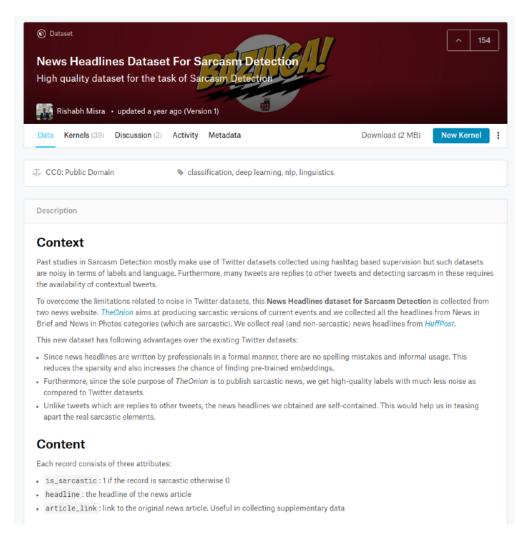
padded = pad\_sequences(sequences, padding='post')

padded = pad\_sequences(sequences, padding='post', maxlen=5)



Sarcasm in News Headlines Dataset by Rishabh Misra

<u> https://rishabhmisra.github.io/publications/</u>



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

not-come-up-with-alt-1819574302", "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-1819576697", "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-

```
{"article link": "https://politics.theonion.com/boehner-just-wants-wife-to-listen-
not-come-up-with-alt-1819574302", "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-1819576697", "headline": "mom starting to fear son's web series
closest thing she will have to grandchild", "is sarcastic": 1}
```

```
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'])
```

```
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'])
```

```
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'])
```

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

sentences = []
labels = []
urls = []

tor 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'])
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, padding='post')
print(padded[0])
print(padded.shape)
```

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, padding='post')
print(padded[0])
```

print(padded.shape)

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, padding='post')
print(padded[0])
```

print(padded.shape)

{'underwood': 24127, 'skillingsbolle': 23055, 'grabs': 12293, 'mobility': 8909, "'assassin's": 12648, 'visualize': 23973, 'hurting': 4992, 'orphaned': 9173, "'agreed''': 24365, 'narration': 28470

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, padding='post')
```

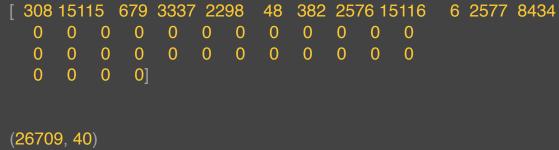
print(padded[0])

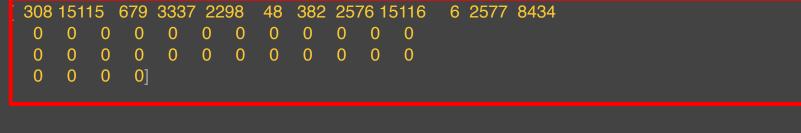
print(padded.shape)

```
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
tokenizer = Tokenizer(oov_token="<OOV>")
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index

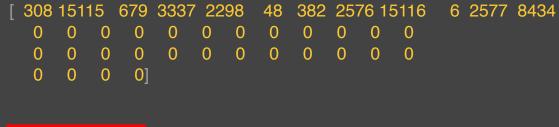
sequences = tokenizer.texts_to_sequences(sentences)
padded = pad_sequences(sequences, padding='post')
print(padded[0])
```

print(padded.shape)





(26709, 40)



(26709, 40)