

Copyright Notice

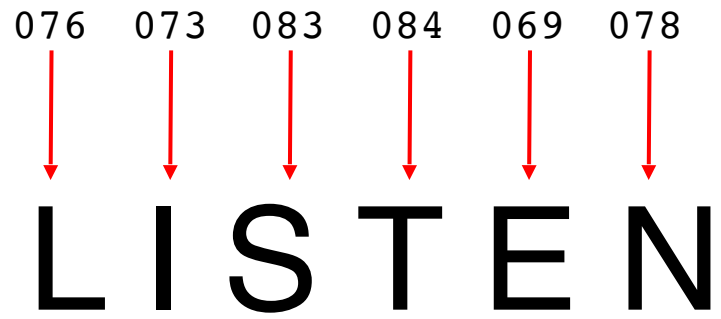
These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) 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 [DeepLearning.AI](#) as the source of the slides.

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

LISTEN

076 073 083 084 069 078



A diagram illustrating the word "LISTEN" with numerical values above each letter. The numbers are 076, 073, 083, 084, 069, and 078, corresponding to the letters L, I, S, T, E, and N respectively. Red arrows point from each number down to its corresponding letter.

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

LISTEN

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

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

I love my dog

I love my dog



001

I love my dog

001



002



003



004



I love my dog

001

002

003

004

I love my cat

I love my dog

001

002

003

004

I love my cat

001

002

003

I love my dog

001

002

003

004

I love my **cat**

001

002

003

005



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)
```

```
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
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
```

```
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
print(word_index)
```



```
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!'  
]
```




```
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,  
  '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,  
'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)
```

```
word_index = tokenizer.word_index
```

```
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_seq = tokenizer.texts_to_sequences(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_seq = tokenizer.texts_to_sequences(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)
```

```
print(word_index)
```

```
print(sequences)
```

```
print(padded)
```



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

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




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




Sarcasm in News Headlines Dataset by Rishabh Misra

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


 Dataset

News Headlines Dataset For Sarcasm Detection

High quality dataset for the task of Sarcasm Detection

 Rishabh Misra · updated a year ago (Version 1)

BAZINGA!



Data

Kernels (39)


Discussion (2)


Activity

Metadata

Download (2 MB)

New Kernel

 CC0: Public Domain

 classification, deep learning, nlp, linguistics

Description

Context

Past studies in Sarcasm Detection mostly make use of Twitter datasets collected using hashtag based supervision but such datasets are noisy in terms of labels and language. Furthermore, many tweets are replies to other tweets and detecting sarcasm in these requires the availability of contextual tweets.

To overcome the limitations related to noise in Twitter datasets, this **News Headlines dataset for Sarcasm Detection** is collected from two news website. [TheOnion](#) aims at producing sarcastic versions of current events and we collected all the headlines from News in Brief and News in Photos categories (which are sarcastic). We collect real (and non-sarcastic) news headlines from [HuffPost](#).

This new dataset has following advantages over the existing Twitter datasets:

- Since news headlines are written by professionals in a formal manner, there are no spelling mistakes and informal usage. This reduces the sparsity and also increases the chance of finding pre-trained embeddings.
- Furthermore, since the sole purpose of *TheOnion* is to publish sarcastic news, we get high-quality labels with much less noise as compared to Twitter datasets.
- Unlike tweets which are replies to other tweets, the news headlines we obtained are self-contained. This would help us in teasing apart the real sarcastic elements.

Content

Each record consists of three attributes:

- `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

`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

```
{"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}
```

[

{"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}

]

```
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'])
```



```
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)
```

```
[ 308 15115  679 3337 2298  48 382 2576 15116  6 2577 8434
  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0]
```

(26709, 40)

308	15115	679	3337	2298	48	382	2576	15116	6	2577	8434
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0]								

(26709, 40)


```
[ 308 15115 679 3337 2298 48 382 2576 15116 6 2577 8434
  0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0 0 0 0 0 0 0 0
  0 0 0 0]
```

(26709, 40)