Tamkang University
淡江大學

文字探勘
(Text Mining)

處理和理解文本
(Processing and Understanding Text)

1082TM04
MBA, BDABI, TKU (E3611) (8480) (Spring 2020)
Mon, 7, 8, 9 (14:10-17:00) (B206)

Chichang Jou
周清江
Associate Professor
副教授
cjou@mail.tku.edu.tw

Min-Yuh Day
戴敏育
Associate Professor
副教授
myday@mail.tku.edu.tw

Dept. of Information Management, Tamkang University
淡江大學 資訊管理學系

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/03/02	文字探勘課程介紹 (Course Orientation on Text Mining)
2	2020/03/09	文字探勘基礎：自然語言處理 (Foundations of Text Mining: Natural Language Processing; NLP)
3	2020/03/16	Python自然語言處理 (Python for Natural Language Processing)
4	2020/03/23	處理和理解文本 (Processing and Understanding Text)
5	2020/03/30	文本表達特徵工程 (Feature Engineering for Text Representation)
6	2020/04/06	人工智慧文本分析個案研究 I (Case Study on Artificial Intelligence for Text Analytics I)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/04/13	文本分類 (Text Classification)
8	2020/04/20	文本摘要和主題模型 (Text Summarization and Topic Models)
9	2020/04/27	期中報告 (Midterm Project Report)
10	2020/05/04	文本相似度和分群 (Text Similarity and Clustering)
11	2020/05/11	語意分析和命名實體識別 (Semantic Analysis and Named Entity Recognition; NER)
12	2020/05/18	情感分析 (Sentiment Analysis)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2020/05/25	人工智慧文本分析個案研究 II (Case Study on Artificial Intelligence for Text Analytics II)
14	2020/06/01	深度學習和通用句子嵌入模型 (Deep Learning and Universal Sentence-Embedding Models)
15	2020/06/08	問答系統與對話系統 (Question Answering and Dialogue Systems)
16	2020/06/15	期末報告 I (Final Project Presentation I)
17	2020/06/22	期末報告 II (Final Project Presentation II)
18	2020/06/29	教師彈性補充教學

Outline

- Processing Text
- Understanding Text

Processing and Understanding Text

Free eBooks - Project Gutenberg

gutenberg.org

Project Gutenberg

search for books

- Browse Catalog
- Bookshelves
- Main Page
- Categories
- Contact Info

Project Gutenberg appreciates your donation!

Donate

- Why donate?

in other languages

- Português
- Deutsch
- Français

hosted by ibiblio

Free eBooks - Project Gutenberg

Book search · Book categories · Browse catalog · Mobile site · Report errors · Terms of use

Some of the Latest eBooks

Welcome

New website available for testing. Visit https://dev.gutenberg.org (or http://dev.gutenberg.org) to test the site (it may have occasional outages, as improvements are made). There is a new website page that lists some known issues, and part of the motivation for the change. If you visit the new website, please consider providing your input and suggestions via an anonymous online survey afterwards.

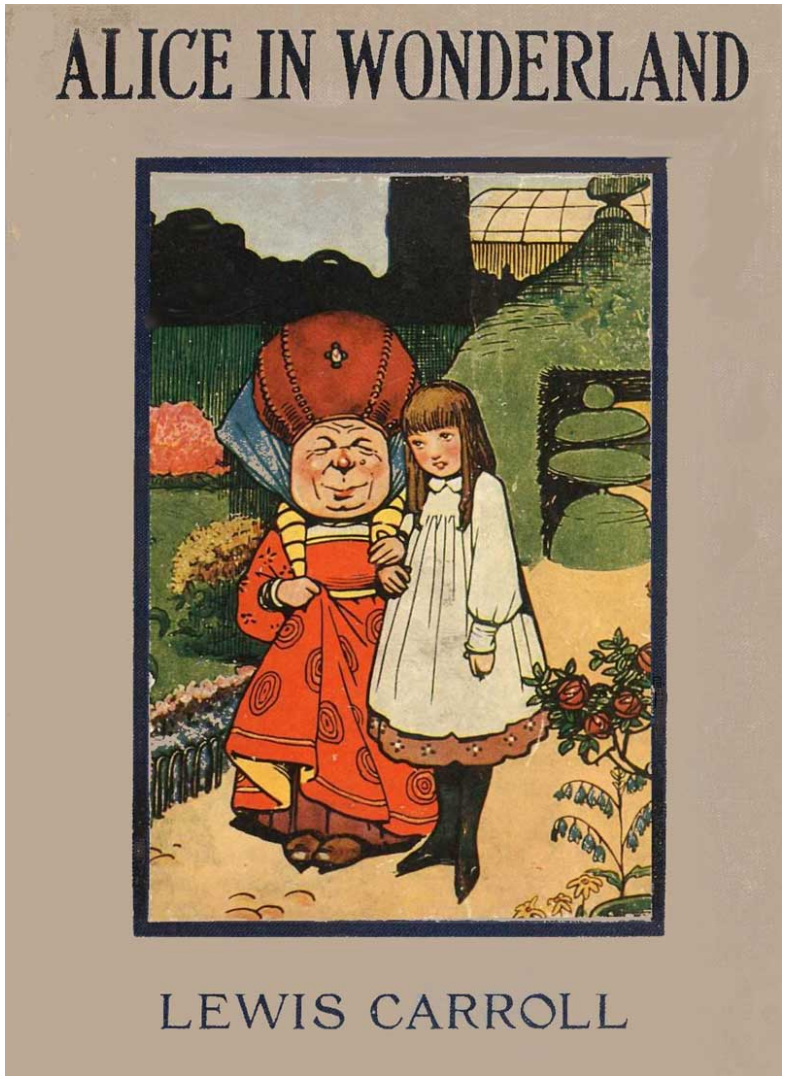
Project Gutenberg is a library of over 60,000 free eBooks. Choose among free epub and Kindle eBooks, download them or read them online. You will find the world's great literature here, with focus on older works for which U.S. copyright has expired. Thousands of volunteers digitized and diligently proofread the eBooks, for enjoyment and education.

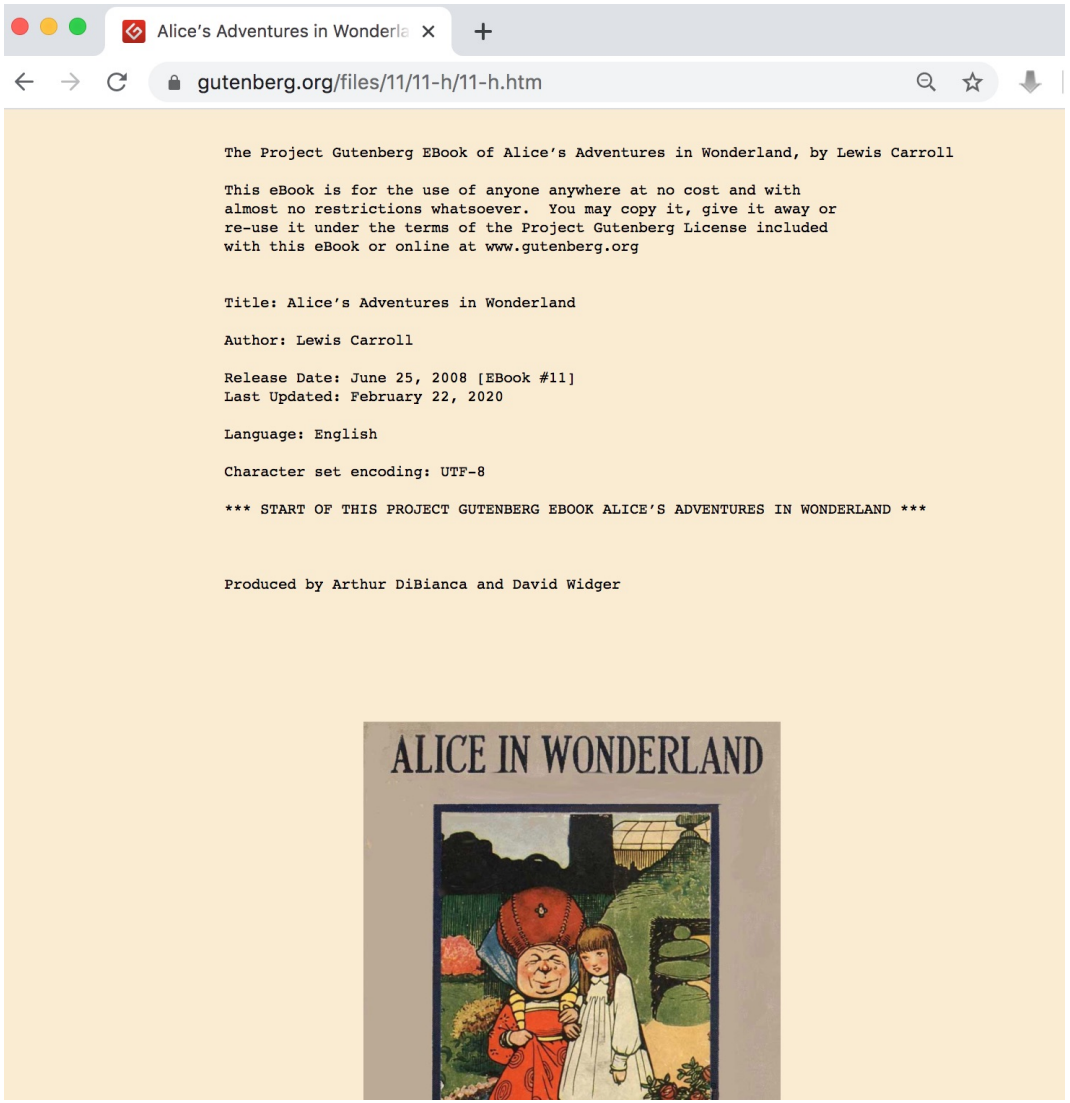
Project Gutenberg Mobile Site

No fee or registration! Everything from Project Gutenberg is gratis, libre, and completely without cost to readers. If you find Project Gutenberg useful, please consider a small donation, to help Project Gutenberg digitize more books, maintain our online presence, and improve Project Gutenberg programs and offerings. Other ways to help include digitizing, proofreading and formatting, recording audio books, or reporting errors.

https://www.gutenberg.org/

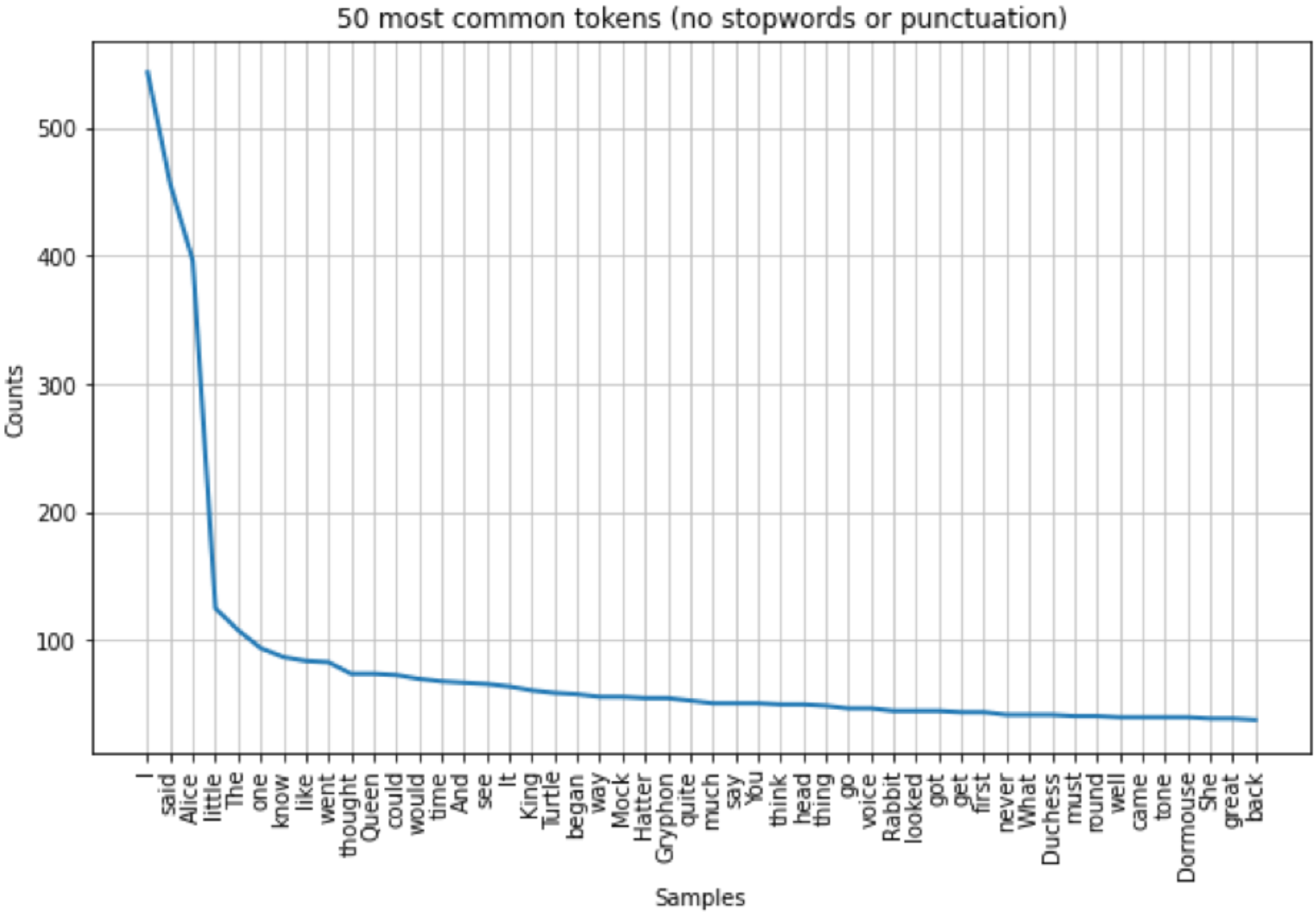
Free eBooks - Project Gutenberg
Alice in Wonderland





https://www.gutenberg.org/files/11/11-h/11-h.htm

Alice Top 50 Tokens

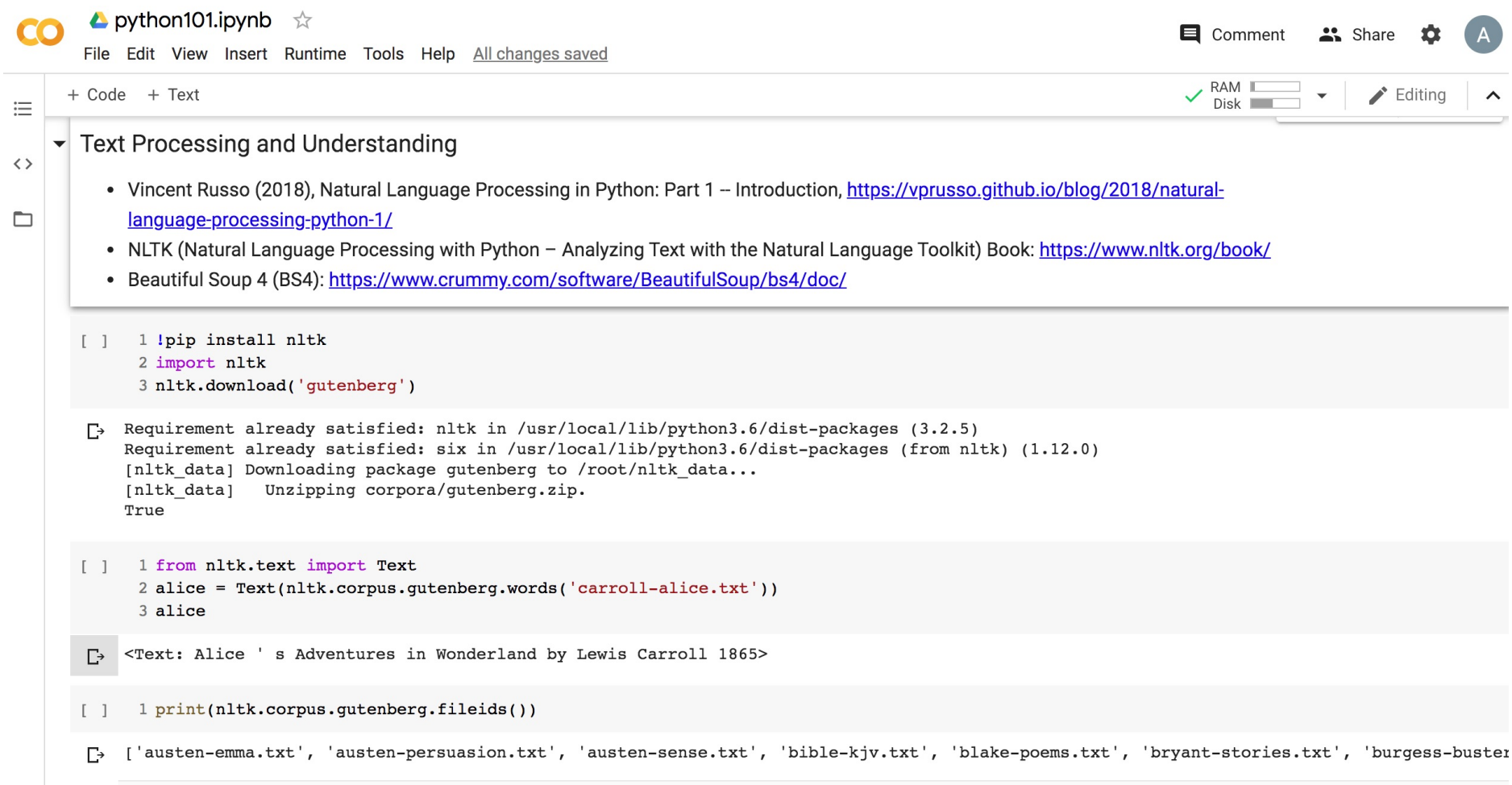


https://tinyurl.com/imtkupython101

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```
nltk.download('gutenberg')
alice = Text(nltk.corpus.gutenberg.words('carroll-alice.txt'))
```



https://tinyurl.com/imtkupython101

alice.concordance("Alice")

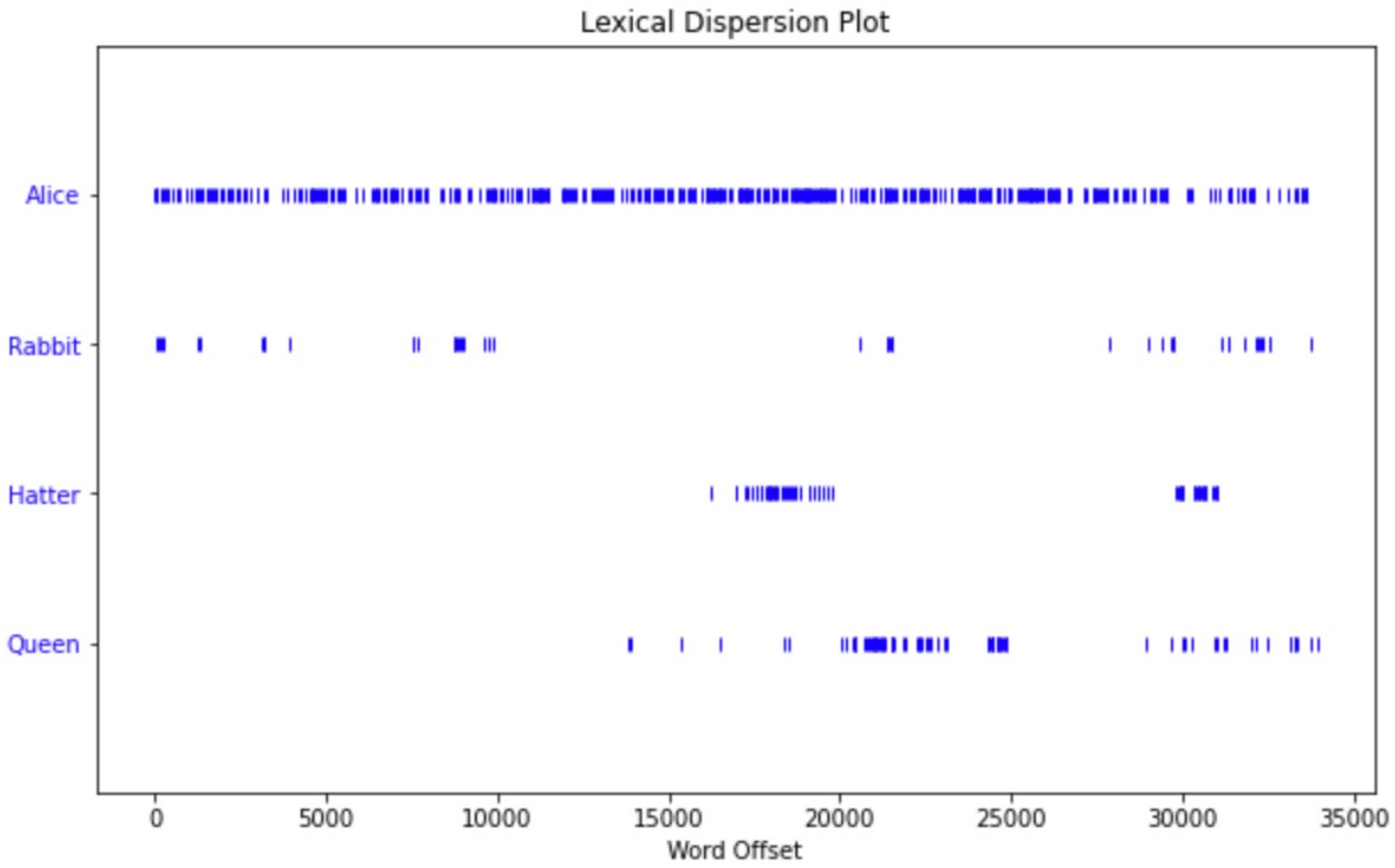
```
 1 alice.concordance("Alice")
```

```
Displaying 25 of 398 matches:
                                      Alice ' s Adventures in Wonderland by Lewi
] CHAPTER I . Down the Rabbit - Hole Alice was beginning to get very tired of s
what is the use of a book ,' thought Alice ' without pictures or conversation ?
so VERY remarkable in that ; nor did Alice think it so VERY much out of the way
looked at it , and then hurried on , Alice started to her feet , for it flashed
 hedge . In another moment down went Alice after it , never once considering ho
ped suddenly down , so suddenly that Alice had not a moment to think about stop
she fell past it . ' Well !' thought Alice to herself , ' after such a fall as
down , I think --' ( for , you see , Alice had learnt several things of this so
tude or Longitude I ' ve got to ?' ( Alice had no idea what Latitude was , or L
 . There was nothing else to do , so Alice soon began talking again . ' Dinah '
cats eat bats , I wonder ?' And here Alice began to get rather sleepy , and wen
dry leaves , and the fall was over . Alice was not a bit hurt , and she jumped
 not a moment to be lost : away went Alice like the wind , and was just in time
 but they were all locked ; and when Alice had been all the way down one side a
on it except a tiny golden key , and Alice ' s first thought was that it might
and to her great delight it fitted ! Alice opened the door and found that it le
ead would go through ,' thought poor Alice , ' it would be of very little use w
ay things had happened lately , that Alice had begun to think that very few thi
ertainly was not here before ,' said Alice ,) and round the neck of the bottle
ay ' Drink me ,' but the wise little Alice was not going to do THAT in a hurry
bottle was NOT marked ' poison ,' so Alice ventured to taste it , and finding i
* * ' What a curious feeling !' said Alice ; ' I must be shutting up like a tel
 for it might end , you know ,' said Alice to herself , ' in my going out altog
garden at once ; but , alas for poor Alice ! when she got to the door , she fou
```

https://tinyurl.com/imtkupython101

alice.dispersion_plot(["Alice", "Rabbit", "Hatter", "Queen"])

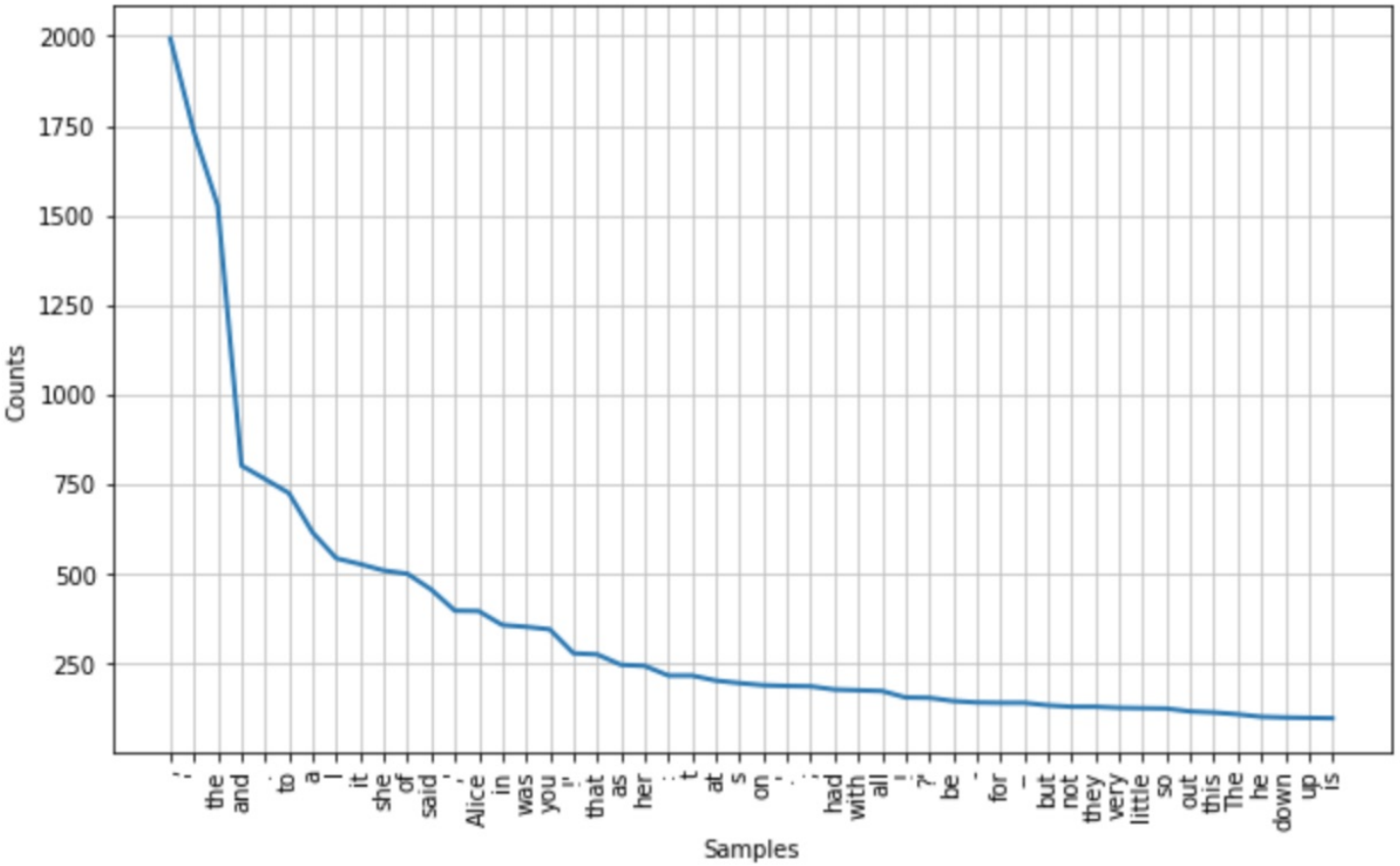
```
1 import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 alice.dispersion_plot(["Alice", "Rabbit", "Hatter", "Queen"])
```



https://tinyurl.com/imtkupython101

fdist = nltk.FreqDist(alice)
fdist.plot(50)

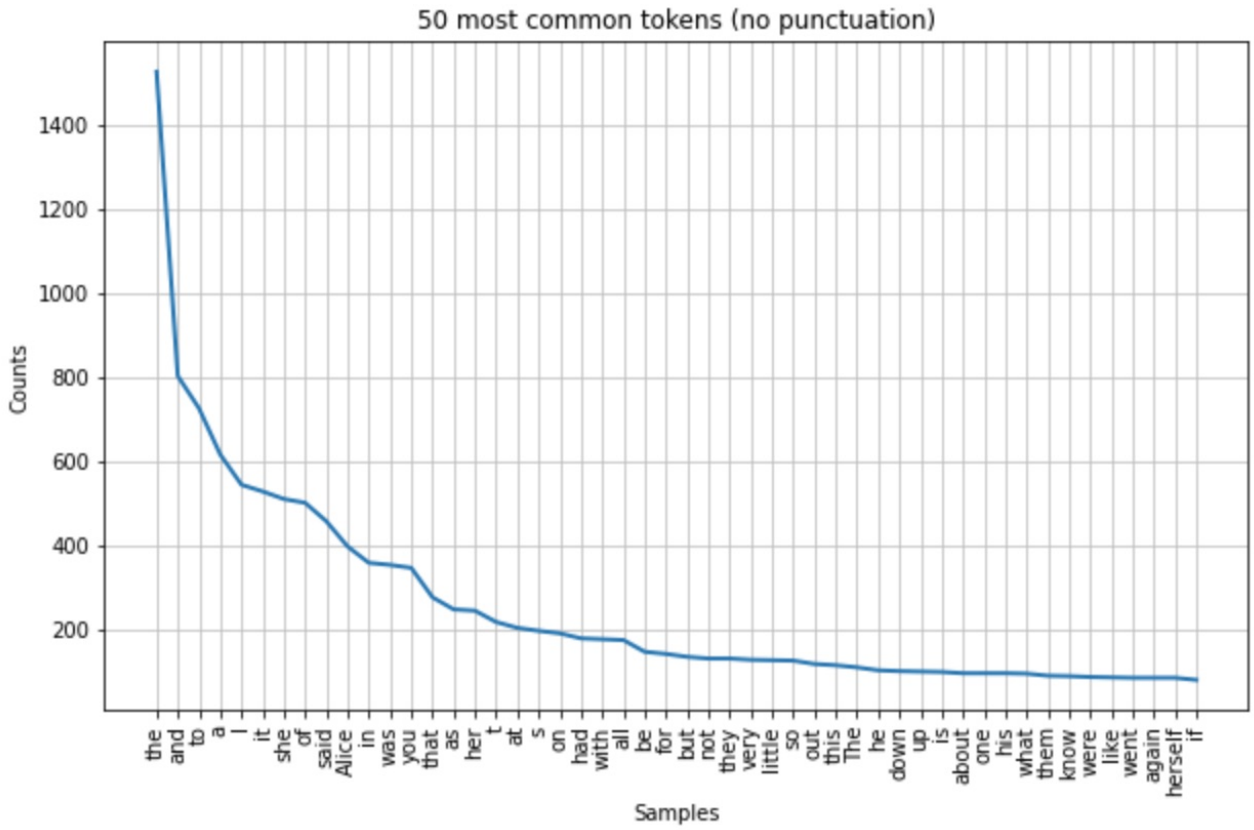
```
1 #import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 fdist = nltk.FreqDist(alice)
4 fdist.plot(50)
```



https://tinyurl.com/imtkupython101

for word, freq in fdist.items() if word.isalpha()

```
1 #import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 fdist_no_punc = nltk.FreqDist(dict((word, freq) for word, freq in fdist.items() if word.isalpha()))
4 fdist_no_punc.plot(50, cumulative=False, title="50 most common tokens (no punctuation)")
```



https://tinyurl.com/imtkupython101

```
nltk.download('stopwords')
stopwords = nltk.corpus.stopwords.words('english')
```

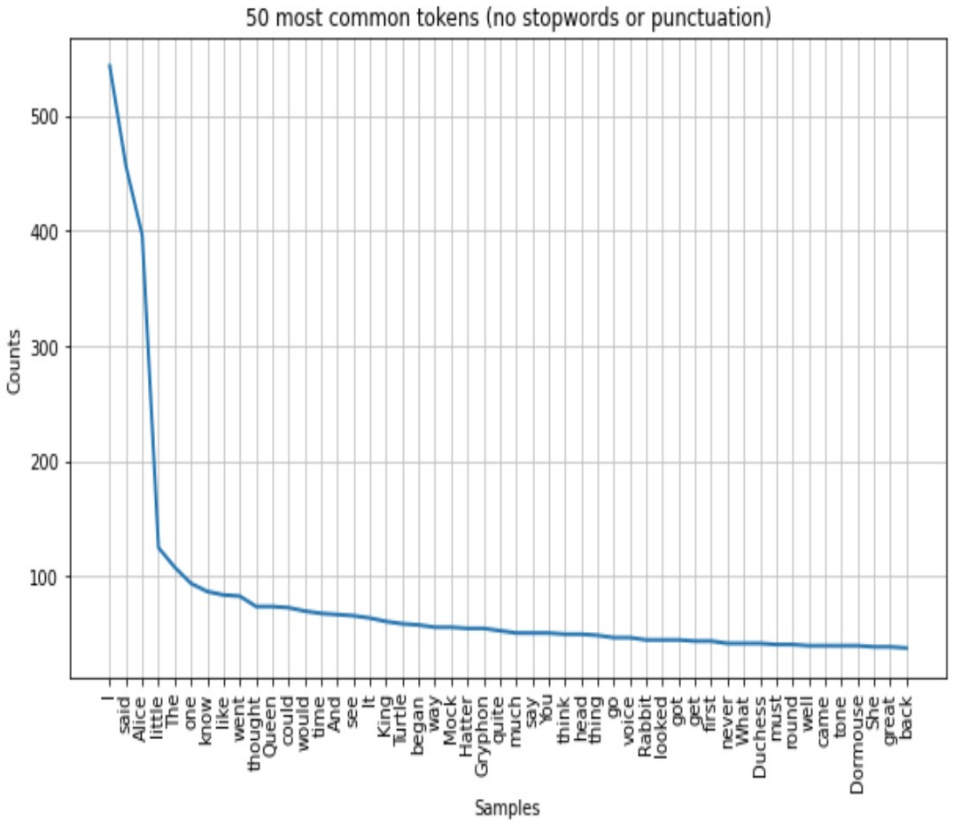
```
1 import nltk
2 nltk.download('stopwords')
3 stopwords = nltk.corpus.stopwords.words('english')
4 stopwords
```

```
'same',
'so',
'than',
'too',
'very',
's',
't',
'can',
'will',
'just',
'don',
"don't",
'should',
"should've",
'now',
```

https://tinyurl.com/imtkupython101

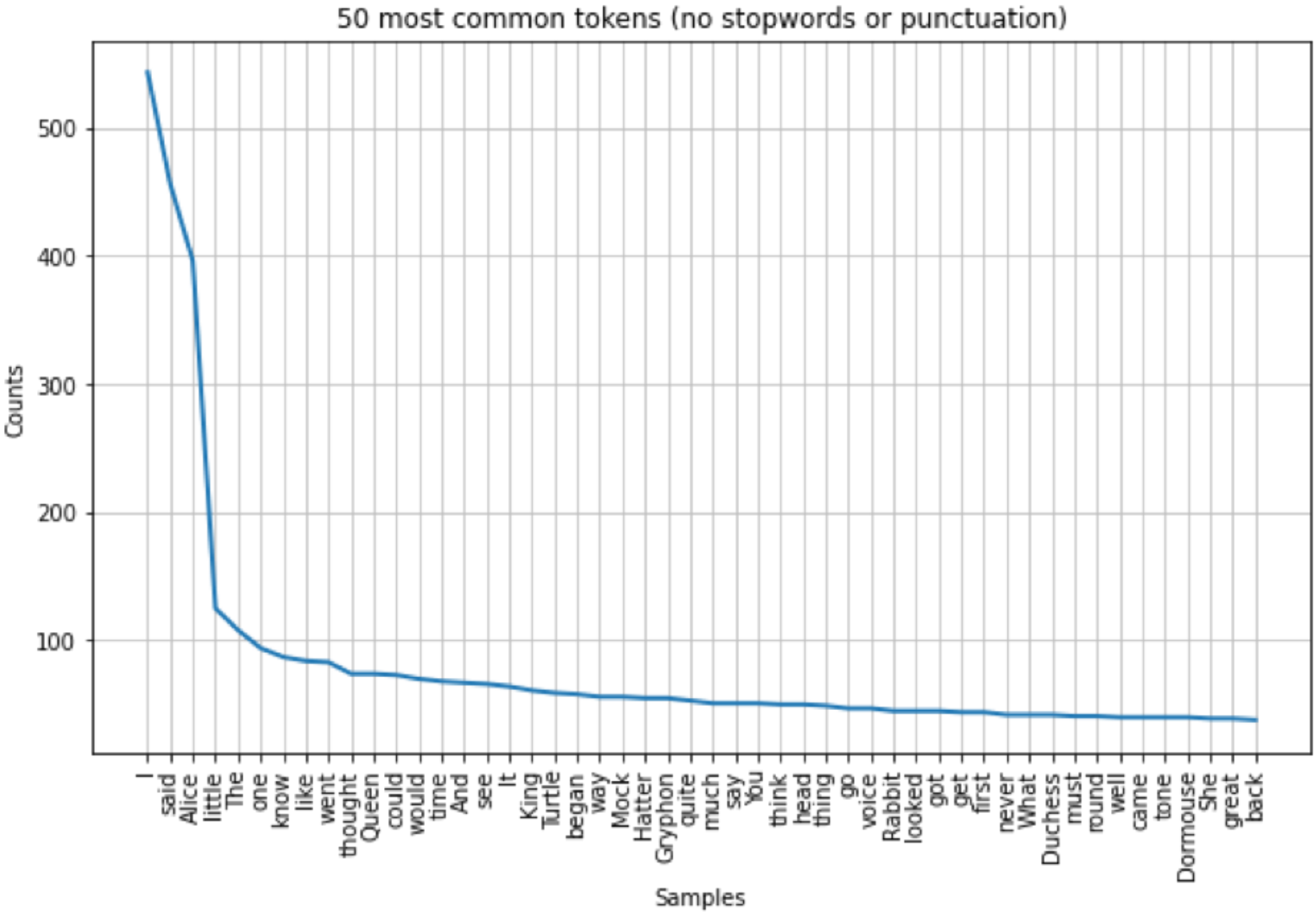
```
for word, freq in fdist.items()
if word not in stopwords and word.isalpha()
```

```
1 #import matplotlib.pyplot as plt
2 plt.figure(figsize=(10, 6))
3 fdist_no_punc_no_stopwords = nltk.FreqDist(dict((word, freq) for word, freq in fdist.items() if word not in stopwords and word.isalpha()))
4 fdist_no_punc_no_stopwords.plot(50, cumulative=False, title="50 most common tokens (no stopwords or punctuation)")
```



https://tinyurl.com/imtkupython101

Alice Top 50 Tokens



https://tinyurl.com/imtkupython101

BeautifulSoup

```
import requests
from bs4 import BeautifulSoup

url = 'https://www.gutenberg.org/files/11/11-h/11-h.htm'
reqs = requests.get(url)
html_doc = reqs.text

soup = BeautifulSoup(html_doc, 'html.parser')
text = soup.get_text()
```

https://tinyurl.com/imtkupython101

tensorflow.keras.preprocessing.text

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

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

tokenizer = Tokenizer(num_words = 100)
tokenizer.fit_on_texts(sentences)
word_index = tokenizer.word_index
print('sentences:', sentences)
print('word index:', word_index)
```

```
sentences: ['i love my dog', 'I, love my cat', 'You love my dog!']
word index: {'love': 1, 'my': 2, 'i': 3, 'dog': 4, 'cat': 5, 'you': 6}
```

https://tinyurl.com/imtkupython101

tensorflow.keras.preprocessing.sequence import pad_sequences

```
import tensorflow as tf
from tensorflow import keras

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, maxlen=5)
print("sentences = ", sentences)
print("Word Index = " , word_index)
print("Sequences = " , sequences)
print("Padded Sequences:")
print(padded)
```

https://tinyurl.com/imtkupython101

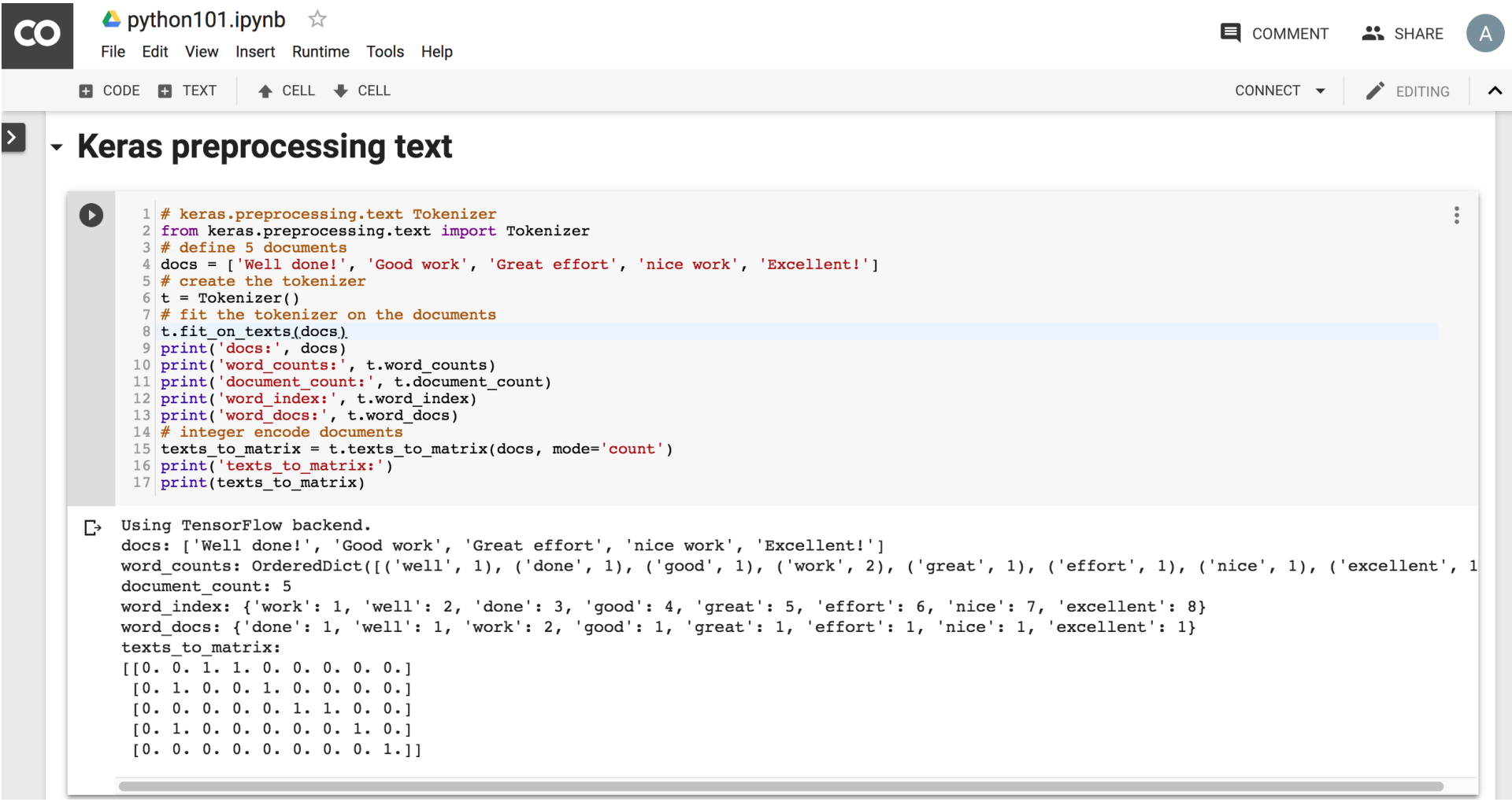
```
tensorflow.keras.preprocessing.sequence
import pad_sequences
```

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

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

Sequences = [[5, 3, 2, 4], [5, 3, 2, 7], [6,
3, 2, 4], [8, 6, 9, 2, 4, 10, 11]]

Padded Sequences: [[ 0 5 3 2 4] [ 0 5 3 2 7]
[ 0 6 3 2 4] [ 9 2 4 10 11]]
```

https://tinyurl.com/imtkupython101

Python in Google Colab

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



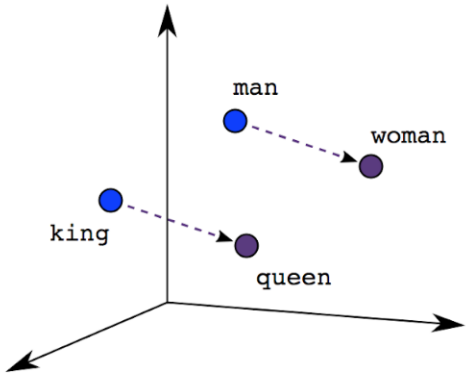
https://tinyurl.com/imtkupython101

One-hot encoding

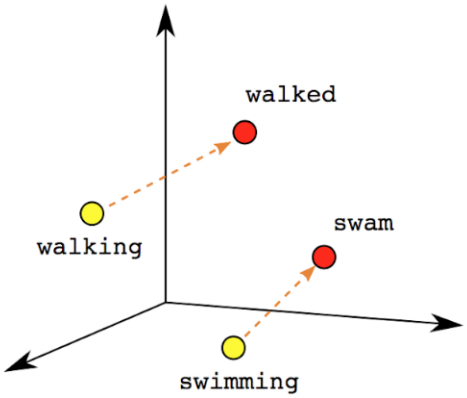
```
                'The mouse ran up the clock' =

The      1   [ [0, 1, 0, 0, 0, 0, 0],
mouse    2     [0, 0, 1, 0, 0, 0, 0],
ran      3     [0, 0, 0, 1, 0, 0, 0],
up       4     [0, 0, 0, 0, 1, 0, 0],
the      1     [0, 1, 0, 0, 0, 0, 0],
clock    5     [0, 0, 0, 0, 0, 1, 0] ]

                [0, 1, 2, 3, 4, 5, 6]
```

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

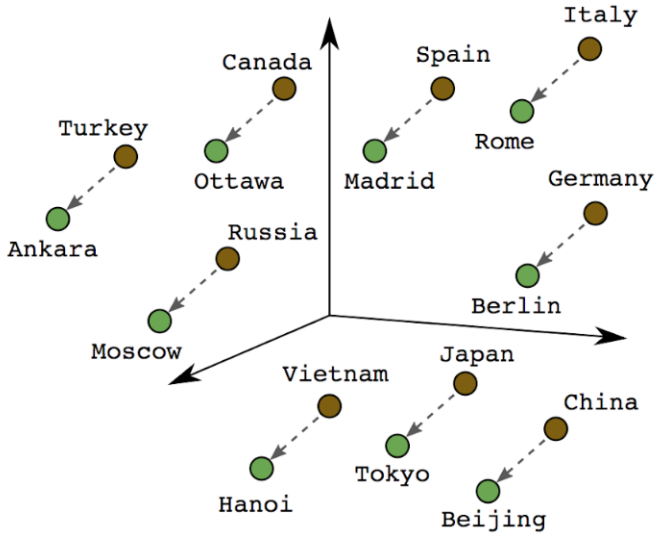
Word embeddings



Male-Female



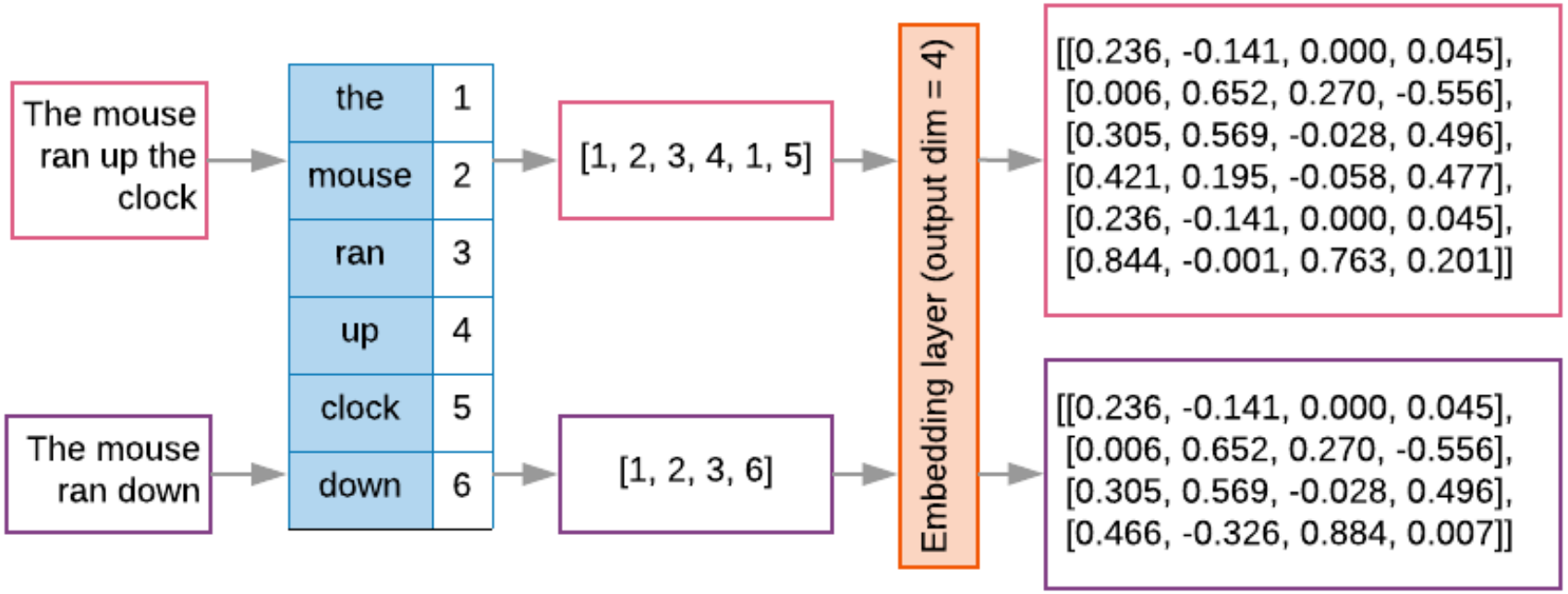
Verb Tense



Country-Capital

Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

Word embeddings



Source: https://developers.google.com/machine-learning/guides/text-classification/step-3

```
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
sortedset = sorted(set(terms))
print('terms =', terms)
print('sortedset =', sortedset)
```

```
1 t1 = 'The mouse ran up the clock'
2 t2 = 'The mouse ran down'
3 s1 = t1.lower().split(' ')
4 s2 = t2.lower().split(' ')
5 terms = s1 + s2
6 sortedset = sorted(set(terms))
7 print('terms =', terms)
8 print('sortedset =', sortedset)
```

```
terms = ['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
sortedset = ['clock', 'down', 'mouse', 'ran', 'the', 'up']
```

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```
t1 = 'The mouse ran up the clock'
t2 = 'The mouse ran down'
s1 = t1.lower().split(' ')
s2 = t2.lower().split(' ')
terms = s1 + s2
print(terms)

tfdict = {}
for term in terms:
    if term not in tfdict:
        tfdict[term] = 1
    else:
        tfdict[term] += 1

a = []
for k,v in tfdict.items():
    a.append('{}, {}'.format(k,v))
print(a)
```

```
['the', 'mouse', 'ran', 'up', 'the', 'clock', 'the', 'mouse', 'ran', 'down']
['the, 3', 'mouse, 2', 'ran, 2', 'up, 1', 'clock, 1', 'down, 1']
```

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```
sorted_by_value_reverse = sorted(tfdict.items(),
key=lambda kv: kv[1], reverse=True)

sorted_by_value_reverse_dict =
dict(sorted_by_value_reverse)

id2word = {id: word for id, word in
enumerate(sorted_by_value_reverse_dict)}

word2id = dict([(v, k) for (k, v) in
id2word.items()])
```

```
sorted_by_value:  [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2:  ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse:  [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key:  [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1
```

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

```
sorted_by_value = sorted(tfdict.items(), key=lambda kv: kv[1])
print('sorted_by_value: ', sorted_by_value)
sorted_by_value2 = sorted(tfdict, key=tfdict.get, reverse=True)
print('sorted_by_value2: ', sorted_by_value2)
sorted_by_value_reverse = sorted(tfdict.items(), key=lambda kv: kv[1], reverse=True)
print('sorted_by_value_reverse: ', sorted_by_value_reverse)
sorted_by_value_reverse_dict = dict(sorted_by_value_reverse)
print('sorted_by_value_reverse_dict', sorted_by_value_reverse_dict)
id2word = {id: word for id, word in enumerate(sorted_by_value_reverse_dict)}
print('id2word', id2word)
word2id = dict([(v, k) for (k, v) in id2word.items()])
print('word2id', word2id)
print('len_words:', len(word2id))

sorted_by_key = sorted(tfdict.items(), key=lambda kv: kv[0])
print('sorted_by_key: ', sorted_by_key)

tfstring = '\n'.join(a)
print(tfstring)
tf = tfdict.get('mouse')
print(tf)
```

```
sorted_by_value:  [('up', 1), ('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3)]
sorted_by_value2:  ['the', 'mouse', 'ran', 'up', 'clock', 'down']
sorted_by_value_reverse:  [('the', 3), ('mouse', 2), ('ran', 2), ('up', 1), ('clock', 1), ('down', 1)]
sorted_by_value_reverse_dict {'the': 3, 'mouse': 2, 'ran': 2, 'up': 1, 'clock': 1, 'down': 1}
id2word {0: 'the', 1: 'mouse', 2: 'ran', 3: 'up', 4: 'clock', 5: 'down'}
word2id {'the': 0, 'mouse': 1, 'ran': 2, 'up': 3, 'clock': 4, 'down': 5}
len_words: 6
sorted_by_key:  [('clock', 1), ('down', 1), ('mouse', 2), ('ran', 2), ('the', 3), ('up', 1)]
the, 3
mouse, 2
ran, 2
up, 1
clock, 1
down, 1
```

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

from keras.preprocessing.text import Tokenizer

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1), ('good', 1), ('work', 2), ('great', 1), ('effort', 1), ('ni
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good': 4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

from keras.preprocessing.text import Tokenizer

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice
work', 'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

texts_to_matrix = t.texts_to_matrix(docs, mode='count')

```
docs: ['Well done!', 'Good work', 'Great effort',
'nice work', 'Excellent!']
word_counts: OrderedDict([('well', 1), ('done', 1),
('good', 1), ('work', 2), ('great', 1), ('effort', 1),
('nice', 1), ('excellent', 1)])
document_count: 5
word_index: {'work': 1, 'well': 2, 'done': 3, 'good':
4, 'great': 5, 'effort': 6, 'nice': 7, 'excellent': 8}
word_docs: {'done': 1, 'well': 1, 'work': 2, 'good': 1,
'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 1. 1. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 1.]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

t.texts_to_matrix(docs, mode='tfidf')

```
from keras.preprocessing.text import Tokenizer
# define 5 documents
docs = ['Well done!', 'Good work', 'Great effort', 'nice work',
'Excellent!']
# create the tokenizer
t = Tokenizer()
# fit the tokenizer on the documents
t.fit_on_texts(docs)
print('docs:', docs)
print('word_counts:', t.word_counts)
print('document_count:', t.document_count)
print('word_index:', t.word_index)
print('word_docs:', t.word_docs)
# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='tfidf')
print('texts_to_matrix:')
print(texts_to_matrix)
```

```
texts_to_matrix:
[[0.         0.         1.25276297 1.25276297 0.         0.
  0.         0.         0.        ]
 [0.         0.98082925 0.         0.         1.25276297 0.
  0.         0.         0.        ]
 [0.         0.         0.         0.         0.         1.25276297
  1.25276297 0.         0.        ]
 [0.         0.98082925 0.         0.         0.         0.
  0.         1.25276297 0.        ]
 [0.         0.         0.         0.         0.         0.
  0.         0.         1.25276297]]
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/

Summary

- Processing Text
- Understanding Text

References

- Vincent Russo (2018), Natural Language Processing in Python: Part 1 -- Introduction, https://vprusso.github.io/blog/2018/natural-language-processing-python-1/
- NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit) Book: https://www.nltk.org/book/
- Beautiful Soup 4 (BS4): https://www.crummy.com/software/BeautifulSoup/bs4/doc/
- Laurence Moroney (2020), Natural Language Processing - Tokenization (NLP Zero to Hero, part 1), https://www.youtube.com/watch?v=fNxaJsNG3-s
- Laurence Moroney (2020), Natural Language Processing - Sequencing - Turning sentence into data (NLP Zero to Hero, part 2), https://www.youtube.com/watch?v=r9QjkdSJZ2g
- Laurence Moroney (2020), Natural Language Processing - Training a model to recognize sentiment in text (NLP Zero to Hero, part 3), https://www.youtube.com/watch?v=Y_hzMnRXjhI