Artificial Intelligence for Text Analytics

Python for Natural Language Processing

Min-Yuh Day, Ph.D, Associate Professor
Institute of Information Management, National Taipei University

https://web.ntpu.edu.tw/~myday

1102AITA03
MBA, IM, NTPU (M5026) (Spring 2022)
Tue 2, 3, 4 (9:10-12:00) (B8F40)

https://meet.google.com/paj-zhhj-my8
<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2022/02/22</td>
<td>Introduction to Artificial Intelligence for Text Analytics</td>
</tr>
<tr>
<td>2</td>
<td>2022/03/01</td>
<td>Foundations of Text Analytics: Natural Language Processing (NLP)</td>
</tr>
<tr>
<td>3</td>
<td>2022/03/08</td>
<td>Python for Natural Language Processing</td>
</tr>
<tr>
<td>4</td>
<td>2022/03/15</td>
<td>Natural Language Processing with Transformers</td>
</tr>
<tr>
<td>5</td>
<td>2022/03/22</td>
<td>Case Study on Artificial Intelligence for Text Analytics I</td>
</tr>
<tr>
<td>6</td>
<td>2022/03/29</td>
<td>Text Classification and Sentiment Analysis</td>
</tr>
<tr>
<td>Week</td>
<td>Date</td>
<td>Subject/Topics</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>-------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>7</td>
<td>2022/04/05</td>
<td>Tomb-Sweeping Day (Holiday, No Classes)</td>
</tr>
<tr>
<td>8</td>
<td>2022/04/12</td>
<td>Midterm Project Report</td>
</tr>
<tr>
<td>9</td>
<td>2022/04/19</td>
<td>Multilingual Named Entity Recognition (NER), Text Similarity and Clustering</td>
</tr>
<tr>
<td>10</td>
<td>2022/04/26</td>
<td>Text Summarization and Topic Models</td>
</tr>
<tr>
<td>11</td>
<td>2022/05/03</td>
<td>Text Generation</td>
</tr>
<tr>
<td>12</td>
<td>2022/05/10</td>
<td>Case Study on Artificial Intelligence for Text Analytics II</td>
</tr>
</tbody>
</table>
# Syllabus

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Subject/Topics</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>2022/05/17</td>
<td>Question Answering and Dialogue Systems</td>
</tr>
<tr>
<td>14</td>
<td>2022/05/24</td>
<td>Deep Learning, Transfer Learning, Zero-Shot, and Few-Shot Learning for Text Analytics</td>
</tr>
<tr>
<td>15</td>
<td>2022/05/31</td>
<td>Final Project Report I</td>
</tr>
<tr>
<td>16</td>
<td>2022/06/07</td>
<td>Final Project Report II</td>
</tr>
<tr>
<td>17</td>
<td>2022/06/14</td>
<td>Self-learning</td>
</tr>
<tr>
<td>18</td>
<td>2022/06/21</td>
<td>Self-learning</td>
</tr>
</tbody>
</table>
Python for Natural Language Processing
Outline

• Python for Natural Language Processing
  • Python Ecosystem for Data Science
  • Python
    • Programming language
• Numpy
  • Scientific computing
• SpaCy
  • Natural Language Processing
Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.

Source: https://www.python.org/doc/essays/blurb/
Python Ecosystem for Data Science

Few (~10) main libraries

- NumPy
- pandas
- SciPy
- Matplotlib
- Jupyter
- PyTorch
- TensorFlow
- Learn
- K
- GitHub
- Stack Overflow

- Statistics
- Scientific
- Graphics
- Structured data
- Machine Learning

- Anconda
- Lab book

Python Ecosystem for Data Science

Visualization
- plotly
- Bokeh
- matplotlib

Data Structures
- NumPy
- pandas

Statistics
- StatsModels

Machine Learning & AI
- scikit-learn
- PyTorch

Source: https://duchesnay.github.io/pystatsml/introduction/python_ecosystem.html
The Quant Finance PyData Stack

PyThalesians  Zipline  DX Analytics
Quantopian  PyTables  PyAlgoTrade  QuantLib
NetworkX

SciPy

pandas

matplotlib

y_t = \beta x_{it} + \mu_t + \epsilon_t

NumPy

SymPy

IPython

Jupyter

TensorFlow

PyTorch

StatsModels

Scikit-learn

Source: http://nbviewer.jupyter.org/format/slides/github/quantopian/pyfolio/blob/master/pyfolio/examples/overview_slides.ipynb#5
Welcome to Colaboratory!

Colaboratory is a free Jupyter notebook environment that requires no setup and runs entirely in the cloud. See our FAQ for more info.

Getting Started

- Overview of Colaboratory
- Loading and saving data: Local files, Drive, Sheets, Google Cloud Storage
- Importing libraries and installing dependencies
- Using Google Cloud BigQuery
- Forms, Charts, Markdown, & Widgets
- TensorFlow with GPU
- Machine Learning Crash Course: Intro to Pandas & First Steps with TensorFlow

Highlighted Features

Seedbank

Looking for Colab notebooks to learn from? Check out Seedbank, a place to discover interactive machine learning examples.

TensorFlow execution

Colaboratory allows you to execute TensorFlow code in your browser with a single click. The example below adds two matrices.

\[
\begin{bmatrix}
1 & 1 & 1
\end{bmatrix}
+ \\
\begin{bmatrix}
1 & 2 & 3
\end{bmatrix}
= \\
\begin{bmatrix}
2 & 3 & 4
\end{bmatrix}
\]
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
NumPy

NumPy Base

N-dimensional array package
Python

matplotlib

Source: https://matplotlib.org/
Python Pandas

$y_{it} = \beta' x_{it} + \mu_i + \epsilon_{it}$

http://pandas.pydata.org/
Connect Google Colab in Google Drive
Google Colab

Connect apps to Drive

Colaboratory
offered by https://colab.research.google.com
A data analysis tool that combines code, output, and descriptive text into one collaborative document.
Connect Colaboratory to Google Drive
Google Colab
Google Colab
Run Jupyter Notebook
Python3 GPU
Google Colab
Google Colab Python Hello World
print('Hello World')
Anaconda
The Most Popular
Python
Data Science Platform

Source: https://www.anaconda.com/
Data science technology for a better world.

Anaconda offers the easiest way to perform Python/R data science and machine learning on a single machine. Start working with thousands of open-source packages and libraries today.

Download Anaconda

https://www.anaconda.com/download
Python
HelloWorld
Anaconda-Navigator

Launchpad
Jupyter Notebook
Jupyter Notebook
New Python 3
print("hello, world")
from platform import python_version
print("Python Version:", python_version())
Python Programming
print("Hello World")

print("Hello World\nThis is a message")

x = 3
print(x)

x = 2
y = 3
print(x, ' ', y)

name = input("Enter a name: ")

x = int(input("What is x? "))

x = float(input("Write a number "))
Python in Google Colab

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

https://tinyurl.com/aintpupython101
Text input and output

In [1]:
print("Hello World")
Hello World

In [2]:
print("Hello World\nThis is a message")
Hello World
This is a message

In [3]:
x = 3
print(x)
3

In [4]:
x = 2
y = 3
print(x, ', ', y)
2 3

In [5]:
name = input("Enter a name: ")
Enter a name: Myday

In [6]:
x = int(input("What is x? "))
What is x? 80

In [7]:
x = float(input("Write a number "))
Write a number 3.6
Variables

x = 2
price = 2.5
word = 'Hello'

word = 'Hello'
word = "Hello"
word = '''Hello'''

x = 2
x = x + 1
x = 5
Python Basic Operators

```
print('7 + 2 =', 7 + 2)
print('7 - 2 =', 7 - 2)
print('7 * 2 =', 7 * 2)
print('7 / 2 =', 7 / 2)
print('7 // 2 =', 7 // 2)
print('7 % 2 =', 7 % 2)
print('7 ** 2 =', 7 ** 2)
```
BMI Calculator in Python

```python
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))
```

Source: http://code.activestate.com/recipes/580615-bmi-code/
BMI Calculator in Python

```python
In [1]:
height_cm = float(input("Enter your height in cm: "))
weight_kg = float(input("Enter your weight in kg: "))

height_m = height_cm/100
BMI = (weight_kg/(height_m**2))

print("Your BMI is: " + str(round(BMI,1)))

Enter your height in cm: 170
Enter your weight in kg: 60
Your BMI is: 20.8
```

Source: http://code.activestate.com/recipes/580615-bmi-code/
Future value of a specified principal amount, rate of interest, and a number of years.
Future Value (FV)

# How much is your $100 worth after 7 years?
print(100 * 1.1 ** 7)
# output = 194.87

Source: https://www.w3resource.com/python-exercises/python-basic-exercise-39.php
Future Value (FV)

\[
\text{fv} = \text{pv} \times ((1 + (r)) ^ n)
\]

\[
\text{print(round(fv, 2))}
\]

\[
\begin{align*}
\text{pv} & = 100 \\
\text{r} & = 0.1 \\
\text{n} & = 7 \\
\text{fv} & = \text{pv} \times ((1 + (r)) ^ n) \\
\text{print(round(fv, 2))} & = 194.87
\end{align*}
\]
Future Value (FV)

```python
amount = 100
interest = 10 #10% = 0.01 * 10
years = 7

future_value = amount * ((1 + (0.01 * interest)) ** years)
print(round(future_value, 2))
```

194.87

if statements

>  greater than
<  smaller than
==  equals
!=  is not

score = 80
if score >=60 :
    print("Pass")
else:
    print("Fail")
```python
score = 90
grade = ""
if score >=90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "E"
print(grade)
# grade = "A"
```

Source: [http://pythonprogramminglanguage.com/](http://pythonprogramminglanguage.com/)

http://pythontutor.com/visualize.html
https://goo.gl/E6w5ph
for loops

```python
for i in range(1,11):
    print(i)
```

1
2
3
4
5
6
7
8
9
10

Source: [http://pythonprogramminglanguage.com/](http://pythonprogramminglanguage.com/)
for loops

```python
for i in range(1,10):
    for j in range(1,10):
        print(i, ' * ' , j , ' = ' , i*j)
```

9 * 1 = 9
9 * 2 = 18
9 * 3 = 27
9 * 4 = 36
9 * 5 = 45
9 * 6 = 54
9 * 7 = 63
9 * 8 = 72
9 * 9 = 81
while loops

```python
age = 10

while age < 20:
    print(age)
    age = age + 1
```

10
11
12
13
14
15
16
17
18
19

Source: https://learnpython.trinket.io/learn-python-part-8-loops/#/while-loops/about-while-loops
def Functions

def convertCMtoM(xcm):
    m = xcm/100
    return m

cm = 180
m = convertCMtoM(cm)
print(str(m))

1.8
 Lists [] 

```
x = [60, 70, 80, 90]
print(len(x))
print(x[0])
print(x[1])
print(x[-1])
```

```
4
60
70
90
```
A tuple in Python is a collection that cannot be modified. A tuple is defined using parenthesis.

```python
tax = (10, 20, 30, 40, 50)
print(x[0])  # 10
print(x[1])  # 20
print(x[2])  # 30
print(x[-1])  # 50
```

Source: http://pythonprogramminglanguage.com/tuples/
Dictionary \{key : value\}

```python
k = { 'EN':'English', 'FR':'French' }
print(k['EN'])
```

Sets {}

animals = {'cat', 'dog'}

animals = {'cat', 'dog'}
print('cat' in animals)  # Check if an element is in a set; prints "True"
print('fish' in animals)  # prints "False"
animals.add('fish')  # Add an element to a set
print('fish' in animals)  # Prints "True"
print(len(animals))  # Number of elements in a set; prints "3"
animals.add('cat')  # Adding an element that is already in the set does nothing
print(len(animals))  # Prints "3"
animals.remove('cat')  # Remove an element from a set
print(len(animals))  # Prints "2"
**File Input / Output**

```python
with open('myfile.txt', 'w') as file:
    file.write('Hello World
This is Python File Input Output')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)
```

Hello World
This is Python File Input Output

text

'Hello World
This is Python File Input Output'
with open('myfile.txt', 'a+') as file:
    file.write('\n' + 'New line')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)

with open('myfile.txt', 'a+') as file:
    file.write('\n' + 'New line')

with open('myfile.txt', 'r') as file:
    text = file.read()
print(text)

Hello World
This is Python File Input Output
New line

Source: https://github.com/TiesdeKok/LearnPythonforResearch/blob/master/0_python_basics.ipynb
try except finally

```python
try:
    file = open("myfile.txt")
    #file = open("myfile.txt", 'w')
    file.write("Python write file")
    print("file saved")

except:
    print("Exception file Error")

finally:
    file.close()
    print("finally process")
```

Exception file Error
finally process

Source: https://pythonbasics.org/try-except/
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello my name is " + self.name)

p1 = Person("Alan", 20)
p1.myfunc()
print(p1.name)
print(p1.age)

Hello my name is Alan
Alan
20

Source: https://www.w3schools.com/python/python_classes.asp
Big Data Analytics with Numpy in Python
Numpy

NumPy
Base
N-dimensional array
package
NumPy is the fundamental package for scientific computing with Python.

Source: http://www.numpy.org/
NumPy

- NumPy provides a multidimensional array object to store homogenous or heterogeneous data; it also provides optimized functions/methods to operate on this array object.

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Getting Started

- Getting NumPy
- Installing the SciPy Stack
- NumPy and SciPy documentation page
- NumPy Tutorial
- NumPy for MATLAB® Users
- NumPy functions by category
- NumPy Mailing List

For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see scipy.org.

http://www.numpy.org/
NumPy ndarray

One-dimensional Array
(1-D Array)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
</tbody>
</table>

Two-dimensional Array
(2-D Array)

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>
v = list(range(1, 6))
v
2 * v

import numpy as np
v = np.arange(1, 6)
v
2 * v
NumPy

Base
N-dimensional array package

```python
1  v = list(range(1, 6))
2  v

[1, 2, 3, 4, 5]

1  2 * v

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

1  import numpy as np
2  v = np.arange(1, 6)
3  v

array([[1, 2, 3, 4, 5]])

1  2 * v

array([[2, 4, 6, 8, 10]])
```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
c = a * b
c
array([ 4, 10, 18])

Source: Yves Hilpisch (2014), Python for Finance: Analyze Big Financial Data, O'Reilly
```python
import numpy as np

a = np.zeros((2,2))  # Create an array of all zeros
print(a)            # Prints "[[ 0.  0.]
                    #    [ 0.  0.]]"

b = np.ones((1,2))  # Create an array of all ones
print(b)            # Prints "[[ 1.  1.]]"

c = np.full((2,2), 7) # Create a constant array
print(c)            # Prints "[[ 7.  7.]
                    #    [ 7.  7.]]"

d = np.eye(2)       # Create a 2x2 identity matrix
print(d)            # Prints "[[ 1.  0.]
                    #    [ 0.  1.]]"

e = np.random.random((2,2)) # Create an array filled with random values
print(e)            # Might print "[[ 0.91940167  0.08143941]
                    #    [ 0.68744134  0.87236687]]"
```

Quickstart tutorial

Prerequisites

Before reading this tutorial you should know a bit of Python. If you would like to refresh your memory, take a look at the Python tutorial.

If you wish to work the examples in this tutorial, you must also have some software installed on your computer. Please see http://scipy.org/install.html for instructions.

The Basics

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

For example, the coordinates of a point in 3D space \([1, 2, 1]\) is an array of rank 1, because it has one axis. That axis has a length of 3. In the example pictured below, the array has rank 2 (it is 2-dimensional). The first dimension (axis) has a length of 2, the second dimension has a length of 3.

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 1 & 2
\end{bmatrix}
\]

NumPy’s array class is called ndarray. It is also known by the alias array. Note that numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays and offers less functionality. The more important attributes of an ndarray object are:

- ndarrays.ndim
  - the number of axes (dimensions) of the array. In the Python world, the number of dimensions is referred to as rank.
- ndarrays.shape
import numpy as np
a = np.arange(15).reshape(3, 5)
a.shape
a.ndim
a.dtype.name
Matrix

An m-by-n matrix is a rectangular array of numbers, consisting of m rows and n columns. Each entry in the matrix is denoted by a_{i,j}, where i is the row index and j is the column index.

\[
\begin{bmatrix}
  a_{1,1} & a_{1,2} & a_{1,3} & \cdots \\
  a_{2,1} & a_{2,2} & a_{2,3} & \cdots \\
  a_{3,1} & a_{3,2} & a_{3,3} & \cdots \\
  \vdots & \vdots & \vdots & \ddots \\
\end{bmatrix}
\]

Source: https://simple.wikipedia.org/wiki/Matrix_(mathematics)
NumPy ndarray: Multidimensional Array Object
NumPy ndarray

One-dimensional Array
(1-D Array)

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
</tr>
<tr>
<td>m-1</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Two-dimensional Array
(2-D Array)
import numpy as np
a = np.array([1, 2, 3, 4, 5])

One-dimensional Array
(1-D Array)

0  1  2  3  4  5

a = np.array([1, 2, 3, 4, 5])
a
array([1, 2, 3, 4, 5])
Two-dimensional Array
(2-D Array)

\[
a = \text{np.array}([[1, 2, 3, 4, 5], [6, 7, 8, 9, 10], [11, 12, 13, 14, 15], [16, 17, 18, 19, 20]])
\]
import numpy as np
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])
a

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>10</td>
<td>11</td>
<td>12</td>
<td>13</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>21</td>
<td>22</td>
<td>23</td>
</tr>
</tbody>
</table>
```python
import numpy as np

a = np.array([[0, 1, 2, 3], [10, 11, 12, 13], [20, 21, 22, 23]])

print(a.ndim)
# 2

print(a.shape)
# (3, 4)
```
NumPy Basics: Arrays and Vectorized Computation
NumPy Array

axis 1

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0,0</td>
<td>0,1</td>
<td>0,2</td>
</tr>
<tr>
<td>1</td>
<td>1,0</td>
<td>1,1</td>
<td>1,2</td>
</tr>
<tr>
<td>2</td>
<td>2,0</td>
<td>2,1</td>
<td>2,2</td>
</tr>
</tbody>
</table>

## Numpy Array

<table>
<thead>
<tr>
<th>Expression</th>
<th>Shape</th>
</tr>
</thead>
<tbody>
<tr>
<td>arr[:, 1:]</td>
<td>(2, 2)</td>
</tr>
<tr>
<td>arr[2]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2, :]</td>
<td>(3,)</td>
</tr>
<tr>
<td>arr[2:, :]</td>
<td>(1, 3)</td>
</tr>
<tr>
<td>arr[:, 2]</td>
<td>(3, 2)</td>
</tr>
<tr>
<td>arr[1, :]</td>
<td>(2,)</td>
</tr>
<tr>
<td>arr[1:2, :]</td>
<td>(1, 2)</td>
</tr>
</tbody>
</table>

Tensor

- 3
  - a rank 0 tensor; this is a scalar with shape []
- [1., 2., 3.]
  - a rank 1 tensor; this is a vector with shape [3]
- [[1., 2., 3.], [4., 5., 6.]]
  - a rank 2 tensor; a matrix with shape [2, 3]
- [[[1., 2., 3.]], [[7., 8., 9.]]]
  - a rank 3 tensor with shape [2, 1, 3]

https://www.tensorflow.org/
Scalar

Vector

Matrix

Tensor

Materials and IPython notebooks for "Python for Data Analysis" by Wes McKinney, published by O'Reilly Media.

<table>
<thead>
<tr>
<th>betatim committed with wesm</th>
<th>Add requirements (#71)</th>
<th>Latest commit ea47998 5 days ago</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasets</td>
<td>Add Kaggle titanic dataset</td>
<td>5 months ago</td>
</tr>
<tr>
<td>examples</td>
<td>Remove sex column from tips dataset</td>
<td>4 months ago</td>
</tr>
<tr>
<td>.gitignore</td>
<td>Add gitignore</td>
<td>2 years ago</td>
</tr>
<tr>
<td>COPYING</td>
<td>Use MIT license for code examples</td>
<td>a month ago</td>
</tr>
<tr>
<td>README.md</td>
<td>Add launch in Azure Notebooks button (#70)</td>
<td>19 days ago</td>
</tr>
<tr>
<td>appa.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch02.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch03.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch04.ipynb</td>
<td>Convert all notebooks to v4 format</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch05.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch06.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch07.ipynb</td>
<td>Convert all notebooks to v4 format</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch08.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch09.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
<tr>
<td>ch10.ipynb</td>
<td>Make more cells markdown instead of raw</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

https://github.com/wesm/pydata-book

NumPy Basics: Arrays and

```python
In [ ]:
import numpy as np
np.random.seed(12345)
import matplotlib.pyplot as plt
plt.rc('figure', figsize=(10, 6))
np.set_printoptions(precision=4, suppress=True)

In [ ]:
import numpy as np
my_arr = np.arange(1000000)
my_list = list(range(1000000))

In [ ]:
#time for _ in range(10): my_arr2 = my_arr * 2
#time for _ in range(10): my_list2 = [x * 2 for x in my_list]

The NumPy ndarray: A Multidimensional Array Object

In [ ]:
import numpy as np
# Generate some random data
data = np.random.randn(2, 3)
data
```

## Natural Language Processing (NLP) and Text Mining

<table>
<thead>
<tr>
<th>Process</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw text</td>
<td></td>
</tr>
<tr>
<td>Sentence Segmentation</td>
<td></td>
</tr>
<tr>
<td>Tokenization</td>
<td></td>
</tr>
<tr>
<td>Part-of-Speech (POS)</td>
<td></td>
</tr>
<tr>
<td>Stop word removal</td>
<td>word’s stem am → am</td>
</tr>
<tr>
<td></td>
<td>word’s lemma am → be</td>
</tr>
<tr>
<td>Stemming / Lemmatization</td>
<td>having → hav</td>
</tr>
<tr>
<td>Dependency Parser</td>
<td>having → have</td>
</tr>
<tr>
<td>String Metrics &amp; Matching</td>
<td></td>
</tr>
</tbody>
</table>

Source: Nitin Hardeniya (2015), NLTK Essentials, Packt Publishing; Florian Leitner (2015), Text mining - from Bayes rule to dependency parsing
spaCy: Natural Language Processing

Industrialy-Strength Natural Language Processing

IN PYTHON

Get things done

spaCy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It’s easy to install, and its API is simple and productive. We like to think of spaCy as the Ruby on Rails of Natural Language Processing.

Blazing fast

spaCy excels at large-scale information extraction tasks. It’s written from the ground up in carefully memory-managed Cython. Independent research in 2015 found spaCy to be the fastest in the world. If your application needs to process entire web dumps, spaCy is the library you want to be using.

Deep learning

spaCy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python’s awesome AI ecosystem. With spaCy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

https://spacy.io/
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

Text Analytics and Natural Language Processing (NLP)

Python for Natural Language Processing

spaCy Chinese Model

Open Chinese Convert (OpenCC, 開放中文轉換)

Jieba 結巴中文分詞

Natural Language Toolkit (NLTK)

Stanza: A Python NLP Library for Many Human Languages

Text Processing and Understanding

NLTK (Natural Language Processing with Python – Analyzing Text with the Natural Language Toolkit)

NLP Zero to Hero

Natural Language Processing - Tokenization (NLP Zero to Hero, part 1)

Natural Language Processing - Sequencing - Turning sentence into data (NLP Zero to Hero, part 2)

Natural Language Processing - Training a model to recognize sentiment in text (NLP Zero to Hero, part 3)

Python for Natural Language Processing

spaCy

- spaCy: Industrial-Strength Natural Language Processing in Python
- Source: https://spacy.io/usage/spacy-101

```python
>>> import spacy
>>> nlp = spacy.load("en_core_web_sm")
>>> doc = nlp("Apple is looking at buying U.K. startup for $1 billion")
>>> for token in doc:
...     print(token.text, token.pos_, token.dep_)
```

https://tinyurl.com/aintpuppython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

```python
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)
```

Stanford University ORG
California GPE

```python
from spacy import displacy
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
displacy.render(doc, style="ent", jupyter=True)
```

Stanford University ORG is located in California GPE. It is a great university.

https://tinyurl.com/aintpupypython101
```python
from spacy import displacy

text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
displacy.render(doc, style="ent", jupyter=True)
displacy.render(doc, style="dep", jupyter=True)
```

Stanford University **ORG** is located in **California **GPE**. It is a great university.

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

https://tinyurl.com/aintpupython101
Python in Google Colab

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4z1zTunjMqf2RkCrT

```python
import spacy
nlp = spacy.load("en_core_web_sm")
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
for ent in doc.ents:
    print(ent.text, ent.label_)
```

Stanford University ORG
California GPE

```python
from spacy import displacy
text = "Stanford University is located in California. It is a great university."
doc = nlp(text)
displacy.render(doc, style="ent", jupyter=True)
```

Stanford University ORG is located in California GPE. It is a great university.

https://tinyurl.com/aintpupython101
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Keras pre-processing text

```python
# keras.preprocessing.text 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('document_count:', t.document_count)
print('word_counts:', t.word_counts)
print('word_index:', t.word_index)

# integer encode documents
texts_to_matrix = t.texts_to_matrix(docs, mode='count')
print('texts_to_matrix:')
print(texts_to_matrix)
```

Using TensorFlow backend.

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: {1: ['well', 'work'], 2: ['good'], 3: ['done'], 4: ['great']}

texts_to_matrix:

```
[[0. 0. 1. 1. 0. 0. 0. 0. ... 0. 0. 0. 0. 1.]
 [0. 1. 0. 0. 0. 1. 0. 0. ... 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 1. 0. 0. ... 0. 0. 1. 0. 1.]
 [0. 1. 0. 0. 0. 0. 0. 0. ... 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. ... 0. 0. 0. 0. 1.]]
```
Text Classification

Source: https://developers.google.com/machine-learning/guides/text-classification/
Text Classification Workflow

• Step 1: Gather Data
• Step 2: Explore Your Data
• Step 2.5: Choose a Model*
• Step 3: Prepare Your Data
• Step 4: Build, Train, and Evaluate Your Model
• Step 5: Tune Hyperparameters
• Step 6: Deploy Your Model

Source: https://developers.google.com/machine-learning/guides/text-classification/
Text Classification Flowchart

Text Classification S/W<1500: N-gram

Text Classification S/W>=1500: Sequence

1. Select top_k features [freq]
   - min(top_1K, 2K, ... 15K, 20K, 25K, ... 90K, all)

2. Normalization mode
   - samplewise
   - None
   - featurewise

3. Embeddings

4. S/W < 15K
   - Yes
     - Fine-tuned pre-trained embedding
   - No
     - Frozen pre-trained embedding
     - Embeddings learned from scratch

5. Build model
   - RNN
   - stacked RNN
   - CNN-RNN
   - sepCNN
   - CNN

6. Hyperparameter tuning

Step 2.5: Choose a Model

Samples/Words < 1500

150,000/100 = 1500

IMDb review dataset, the samples/words-per-sample ratio is ~ 144

Step 2.5: Choose a Model

Samples/Words < 15,000

\[1,500,000/100 = 15,000\]

Step 3: Prepare Your Data

Texts:
T1: 'The mouse ran up the clock'
T2: 'The mouse ran down'

Token Index:
{'the': 1, 'mouse': 2, 'ran': 3, 'up': 4, 'clock': 5, 'down': 6}.
NOTE: 'the' occurs most frequently,
so the index value of 1 is assigned to it.
Some libraries reserve index 0 for unknown tokens,
as is the case here.

Sequence of token indexes:
T1: 'The mouse ran up the clock' =
[1, 2, 3, 4, 1, 5]
T1: 'The mouse ran down' =
[1, 2, 3, 6]
One-hot encoding

'The mouse ran up the clock' =

<table>
<thead>
<tr>
<th>Word</th>
<th>Index</th>
<th>Encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>The</td>
<td>1</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>mouse</td>
<td>2</td>
<td>[0, 0, 1, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>ran</td>
<td>3</td>
<td>[0, 0, 0, 1, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>up</td>
<td>4</td>
<td>[0, 0, 0, 0, 1, 0, 0, 0]</td>
</tr>
<tr>
<td>the</td>
<td>1</td>
<td>[0, 1, 0, 0, 0, 0, 0, 0]</td>
</tr>
<tr>
<td>clock</td>
<td>5</td>
<td>[0, 0, 0, 0, 0, 1, 0, 0]</td>
</tr>
</tbody>
</table>

[0, 1, 2, 3, 4, 5, 6]
Word embeddings

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

The mouse ran up the clock

The mouse ran down

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

https://colab.research.google.com/drive/1FEG6DnGwvfwUdeo4zJ1zTunjMqf2RkCrT
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)

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)]
from keras.preprocessing.text import Tokenizer

docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
t = Tokenizer()
t.fit_on_texts(docs)
print('word_count': t.word_count)
print('document_count': t.document_count)
print('word_index': t.word_index)
print('word_docs': t.word_docs)

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

docs: ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']
word_count: {'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': 2, 'work': 2, 'good': 1, 'great': 1, 'effort': 1, 'nice': 1, 'excellent': 1}
texts_to_matrix:
[[0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1.]]
from keras.preprocessing.text import Tokenizer

docs = ['Well done!', 'Good work', 'Great effort', 'nice work', 'Excellent!']

t = Tokenizer()
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)

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/
```python
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)
```

Source: https://machinelearningmastery.com/prepare-text-data-deep-learning-keras/
NLP
Libraries and Tools
spacy: Natural Language Processing

Industrial-Strength Natural Language Processing in Python

Get things done
spacy is designed to help you do real work — to build real products, or gather real insights. The library respects your time, and tries to avoid wasting it. It's easy to install, and its API is simple and productive. We like to think of spacy as the Ruby on Rails of Natural Language Processing.

Blazing fast
spacy excels at large-scale information extraction tasks. It's written from the ground up in carefully memory-managed Cython. Independent research in 2015 found spacy to be the fastest in the world. If your application needs to process entire web dumps, spacy is the library you want to be using.

Deep learning
spacy is the best way to prepare text for deep learning. It interoperates seamlessly with TensorFlow, PyTorch, scikit-learn, Gensim and the rest of Python's awesome AI ecosystem. With spacy, you can easily construct linguistically sophisticated statistical models for a variety of NLP problems.

https://spacy.io/
NLTK (Natural Language Toolkit)

NLTK 3.0 documentation

NEXT | MODULES | INDEX

Natural Language Toolkit

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

Thanks to a hands-on guide introducing programming fundamentals alongside topics in computational linguistics, plus comprehensive API documentation, NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike. NLTK is available for Windows, Mac OS X, and Linux. Best of all, NLTK is a free, open source, community-driven project.

NLTK has been called "a wonderful tool for teaching, and working in, computational linguistics using Python," and "an amazing library to play with natural language."

Natural Language Processing with Python provides a practical introduction to programming for language processing. Written by the creators of NLTK, it guides the reader through the fundamentals of writing Python programs, working with corpora, categorizing text, analyzing linguistic structure, and more. The book is being updated for Python 3 and NLTK 3. (The original Python 2 version is still available at http://nltk.org/book_1ed.)

Some simple things you can do with NLTK

Tokenize and tag some text:

>>> import nltk

http://www.nltk.org/
Natural Language Processing with Python
– Analyzing Text with the Natural Language Toolkit

Steven Bird, Ewan Klein, and Edward Loper

This version of the NLTK book is updated for Python 3 and NLTK 3. The first edition of the book, published by O'Reilly, is available at http://nltk.org/book_1ed/. (There are currently no plans for a second edition of the book.)

0. Preface
1. Language Processing and Python
2. Accessing Text Corpora and Lexical Resources
3. Processing Raw Text
4. Writing Structured Programs
5. Categorizing and Tagging Words (minor fixes still required)
6. Learning to Classify Text
7. Extracting Information from Text
8. Analyzing Sentence Structure
9. Building Feature Based Grammars
10. Analyzing the Meaning of Sentences (minor fixes still required)
11. Managing Linguistic Data (minor fixes still required)
12. Afterword: Facing the Language Challenge

Bibliography
Term Index

This book is made available under the terms of the Creative Commons Attribution Noncommercial No-Derivative-Works 3.0 US License. Please post any questions about the materials to the nltk-users mailing list. Please report any errors on the issue tracker.

http://www.nltk.org/book/
gensim

Gensim is a FREE Python library

- Scalable statistical semantics
- Analyze plain-text documents for semantic structure
- Retrieve semantically similar documents

https://radimrehurek.com/gensim/
TextBlob: Simplified Text Processing

Release v0.12.0. (Changelog)

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.

```python
from textblob import TextBlob
text = 'The titular threat of The Blob has always struck me as the ultimate movie monster: an insatiably hungry, amoeba-like mass able to penetrate virtually any safeguard, capable of—as a doomed doctor chillingly describes it—"assimilating flesh on contact. Snide comparisons to gelatin be damned, it’s a concept with the most devastating of potential consequences, not unlike the grey goo scenario proposed by technological theorists fearful of artificial intelligence run rampant.'

blob = TextBlob(text)

blob.tags  # [('The', 'DT'), ('titular', 'JJ'),
# ('threat', 'NN'), ('of', 'IN'), ...]

blob.noun_phrases            # WordList(['titular threat', 'blob',
# 'ultimate movie monster',
# 'amoeba-like mass', ...])

for sentence in blob.sentences:
    print(sentence.sentiment.polarity)
    # 0.050
```

https://textblob.readthedocs.io
Welcome to polyglot’s documentation!

polyglot

Polyglot is a natural language pipeline that supports massive multilingual applications.

- Free software: GPLv3 license
- Documentation: http://polyglot.readthedocs.org

Features

- Tokenization (165 Languages)
- Language detection (196 Languages)
- Named Entity Recognition (40 Languages)
- Part of Speech Tagging (16 Languages)
- Sentiment Analysis (136 Languages)
- Word Embeddings (137 Languages)
- Morphological analysis (135 Languages)
- Transliteration (69 Languages)

https://polyglot.readthedocs.io/
scikit-learn

Machine Learning in Python

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

Classification

Identifying to which category an object belongs to.

Applications: Spam detection, Image recognition.
Algorithms: SVM, nearest neighbors, random forest, ...

Regression

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.
Algorithms: SVR, ridge regression, Lasso, ...

Clustering

Automatic grouping of similar objects into sets.

Applications: Customer segmentation, Grouping experiment outcomes
Algorithms: k-Means, spectral clustering, mean-shift, ...

Dimensionality reduction

Reducing the number of random variables to consider.

Applications: Visualization, Increased efficiency

Model selection

Comparing, validating and choosing parameters and models.

Goal: Improved accuracy via parameter tuning

Preprocessing

Feature extraction and normalization.

Application: Transforming input data such as text for use with machine learning algorithms.
Modules: preprocessing, feature extraction.

http://scikit-learn.org/
TensorFlow NLP Examples

• Basic Text Classification
  (Text Classification) (46 Seconds)
  • https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_text_classification.ipynb

• NMT with Attention
  (20-30 minutes)
  • https://colab.research.google.com/github/tensorflow/tensorflow/blob/master/tensorflow/contrib/eager/python/examples/nmt_with_attention/nmt_with_attention.ipynb
Text Classification
IMDB Movie Reviews

https://colab.research.google.com/drive/1x16h1GhHsL IrLYtPCvCHaoO1W-i_gror

This notebook classifies movie reviews as positive or negative using the text of the review. This is an example of binary—or two-class—classification, an important and widely applicable kind of machine learning problem.

We'll use the IMDB dataset that contains the text of 50,000 movie reviews from the Internet Movie Database. These are split into 25,000 reviews for training and 25,000 reviews for testing. The training and testing sets are balanced, meaning they contain an equal number of positive and negative reviews.

This notebook uses tf.keras, a high-level API to build and train models in TensorFlow. For a more advanced text classification tutorial using tf.keras, see the MLCC Text Classification Guide.
NLP with Transformers Github

https://github.com/nlp-with-transformers/notebooks
Running on a cloud platform

To run these notebooks on a cloud platform, just click on one of the badges in the table below:

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Colab</th>
<th>Kaggle</th>
<th>Gradient</th>
<th>Studio Lab</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Text Classification</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Transformer Anatomy</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Multilingual Named Entity Recognition</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Text Generation</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Summarization</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Question Answering</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Making Transformers Efficient in Production</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Dealing with Few to No Labels</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Training Transformers from Scratch</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
<tr>
<td>Future Directions</td>
<td>Open in Colab</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open in Kaggle" /></td>
<td>Run on Gradient</td>
<td><img src="https://github.com/nlp-with-transformers/notebooks" alt="Open Studio Lab" /></td>
</tr>
</tbody>
</table>

Nowadays, the GPUs on Colab tend to be K80s (which have limited memory), so we recommend using Kaggle, Gradient, or SageMaker Studio Lab. These platforms tend to provide more performant GPUs like P100s, all for free!

https://github.com/nlp-with-transformers/notebooks
Python in Google Colab (Python101)

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

https://tinyurl.com/aintpupython101
Summary

• Python for Natural Language Processing
  • Python Ecosystem for Data Science
• Python
  • Programming language
• Numpy
  • Scientific computing
• SpaCy
  • Natural Language Processing
References

- Lewis Tunstall, Leandro von Werra, and Thomas Wolf (2022), Natural Language Processing with Transformers: Building Language Applications with Hugging Face, O'Reilly Media.
- Denis Rothman (2021), Transformers for Natural Language Processing: Build innovative deep neural network architectures for NLP with Python, PyTorch, TensorFlow, BERT, RoBERTa, and more, Packt Publishing.
- Savaş Yıldırım and Meysam Asgari-Chenaghlu (2021), Mastering Transformers: Build state-of-the-art models from scratch with advanced natural language processing techniques, Packt Publishing.
- Charu C. Aggarwal (2018), Machine Learning for Text, Springer.
- Gabe Ignatow and Rada F. Mihalcea (2017), An Introduction to Text Mining: Research Design, Data Collection, and Analysis, SAGE Publications.
- Rajesh Arumugam (2018), Hands-On Natural Language Processing with Python: A practical guide to applying deep learning architectures to your NLP applications, Packt.
- Python Programming, https://pythonprogramming.net/
- Python, https://www.python.org/
- Pandas, http://pandas.pydata.org/