MULTITHREADING - SUBCLASSING THREAD

Ph.D. / Golden Gate Ave, San Francisco / Seoul National Univ / Carnegie Mellon / UC Berkeley / DevOps / Deep Learning / Visualization

Sponsor Open Source development activities and free contents for everyone.

Donate with PayPal

Thank you.

- K Hong (http://bogotobogo.com/about_us.php)

SHARE
(http://www.addthis.com/bookmark.php?v=250&username=khhong7)

bogotobogo.com site search:

Search

Python Multithread

Creating a thread and passing arguments to the thread (/python/Multithread/python_multithreading_creating_threads.php)

Identifying threads - naming and logging

run() methods

So far, we've been using a thread by instantiating the **Thread** class given by the package (threading.py (http://hg.python.org/cpython/file/3.4/Lib/threading.py)). To create our own thread in Python, we'll want to make our class to work as a thread. For this, we should subclass our class from the Thread class.

First thing we need to do is to import Thread using the following code:

```
from threading import Thread
```

Then, we should subclass our class from the **Thread** class like this:

Python tutorial

```
class MyThread(Thread):
```

Just for reference, here is a code snippet from the package for the Thread class:

```
class Thread:
    ...

    def start(self):
        """Start the thread's activity.

        It must be called at most once per thread object. It arranges for the
        object's run() method to be invoked in a separate thread of control.

        This method will raise a RuntimeError if called more than once on the
        same thread object.

        """
        if not self._initialized:
            raise RuntimeError("thread.__init__() not called")

        if self._started.is_set():
            raise RuntimeError("threads can only be started once")
        with _active_limbo_lock:
            _limbo[self] = self
        try:
            _start_new_thread(self._bootstrap, ())
        except Exception:
            with _active_limbo_lock:
                del _limbo[self]
            raise
        self._started.wait()

    def _bootstrap(self):
        try:
            self._bootstrap_inner()
        except:
            if self._daemonic and _sys is None:
                return
            raise

    def _bootstrap_inner(self):
        try:
         ...

            try:
                self.run()
            except SystemExit:
                pass
            except:

    def run(self):
        try:
            if self._target:
                self._target(*self._args, **self._kwargs)
        finally:
            # Avoid a refcycle if the thread is running a function with
            # an argument that has a member that points to the thread.
            del self._target, self._args, self._kwargs
```

As a Thread starts up, it does some basic initialization and then calls its **run()** method, which calls the target function passed to the constructor. The Thread class represents an activity that runs in a separate thread of control. There are two ways to specify the activity:

1. by passing a callable object to the constructor
2. by overriding the **run()** method in a subclass

No other methods (except for the constructor) should be overridden in a subclass. In other words, we only override the **__init__()** and **run()** methods of a class.

CONFLUENT
Apache Kafka
Transaction Data Streaming
Powering Change Data Capture through Apache Kafka®.
Get the eBook

In this section, we will create a subclass of Thread and override **run()** to do whatever is necessary:

```
import threading

class MyThread(threading.Thread):

    def run(self):
        pass

if __name__ == '__main__':
    for i in range(3):
        t = MyThread()
        t.start()
```

Once a thread object is created, its activity must be started by calling the thread's **start()** method. This invokes the **run()** method in a separate thread of control.

Once the thread's activity is started, the thread is considered 'alive'. It stops being alive when its **run()** method terminates - either normally, or by raising an unhandled exception. The **is_alive()** method tests whether the thread is alive.

if__name__ == '__main__'
(/python/python_if__name__equa

argparse
(/python/python_argparse.php)

Exceptions
(/python/python_try_except_final

@static method vs class method
(/python/python_differences_bet

Private attributes and private methods
(/python/python_private_attribut

bits, bytes, bitstring, and constBitStream
(/python/python_bits_bytes_bitst

json.dump(s) and json.load(s)
(/python/python-json-dumps-loads-file-read-write.php)

Python Object Serialization - pickle and json
(/python/python_serialization_pic

Python Object Serialization - yaml and json
(/python/python_yaml_json_conv

Priority queue and heap queue data structure
(/python/python_PriorityQueue_l

Graph data structure
(/python/python_graph_data_stru

Dijkstra's shortest path algorithm
(/python/python_Dijkstras_Short

Prim's spanning tree algorithm
(/python/python_Prims_Spanning

Closure
(/python/python_closure.php)

Functional programming in Python
(/python/python_functional_prog

```
import threading
import time

class MyThread(threading.Thread):

    def run(self):
        time.sleep(5)
        return

if __name__ == '__main__':
    for i in range(3):
        t = MyThread()
        t.start()
        print 't.is_alive()=', t.is_alive()
        t.join()
        print 't.is_alive()=', t.is_alive()
```

Output:

```
t.is_alive()= True
t.is_alive()= False
t.is_alive()= True
t.is_alive()= False
t.is_alive()= True
t.is_alive()= False
```

As we can see from the output, each of the three thread is alive just after the start but **t.is_alive()=False** after terminated.

Before we move forward, for our convenience, let's put a logging feature into a place:

```
import threading
import time
import logging

logging.basicConfig(level=logging.DEBUG,
                    format='(%(threadName)-9s) %(message)s',)

class MyThread(threading.Thread):

    def run(self):
        logging.debug('running')
        return

if __name__ == '__main__':
    for i in range(3):
        t = MyThread()
        t.start()
```

Output:

```
(Thread-1 ) running
(Thread-2 ) running
(Thread-3 ) running
```

Passing args to the customized thread

Because the ***args** and ****kwargs** values passed to the Thread constructor are saved in private variables, they are not easily accessed from a subclass. To pass arguments to a custom thread type, we need to redefine the constructor to save the values in an instance attribute that can be seen in the subclass:

framework
(/python/python_network_progra

Python Network Programming IV - Asynchronous Request Handling : ThreadingMixIn and ForkingMixIn
(/python/python_network_progra

Python Coding Questions I
(/python/python_interview_quest

Python Coding Questions II
(/python/python_interview_quest

Python Coding Questions III
(/python/python_interview_quest

Python Coding Questions IV
(/python/python_interview_quest

Python Coding Questions V
(/python/python_interview_quest

Python Coding Questions VI
(/python/python_interview_quest

Python Coding Questions VII
(/python/python_interview_quest

Python Coding Questions VIII
(/python/python_interview_quest

Image processing with Python image library Pillow
(/python/python_image_processi

Python and C++ with SIP
(/python/python_cpp_sip.php)

PyDev with Eclipse
(/python/pydev_eclipse_plugin_ir

Matplotlib
(/python/python_matplotlib.php)

Redis with Python
(/python/python_redis_with_pyth

NumPy array basics A
(/python/python_numpy_array_tu

NumPy Matrix and Linear Algebra

```
import threading
import time
import logging

logging.basicConfig(level=logging.DEBUG,
                    format='(%(threadName)-9s) %(message)s',)

class MyThread(threading.Thread):


    def __init__(self, group=None, target=None, name=None,
                 args=(), kwargs=None, verbose=None):
        super(MyThread,self).__init__(group=group, target=target,
                                      name=name, verbose=verbose)
        self.args = args
        self.kwargs = kwargs
        return

    def run(self):
        logging.debug('running with %s and %s', self.args, self.kwargs)
        return

if __name__ == '__main__':
    for i in range(3):
        t = MyThread(args=(i,), kwargs={'a':1, 'b':2})
        t.start()
```

Output:

```
(Thread-1 ) running with (0,) and {'a': 1, 'b': 2}
(Thread-2 ) running with (1,) and {'a': 1, 'b': 2}
(Thread-3 ) running with (2,) and {'a': 1, 'b': 2}
```

We overrided the **__init__()** using:

```
super(MyThread,self).__init__()
```

For Python 3, we could have used without any args within the **super()**, like this:

```
super().__init__()
```

Python Multithread

Creating a thread and passing arguments to the thread
(/python/Multithread/python_multithreading_creating_threads.php)

Identifying threads - naming and logging
(/python/Multithread/python_multithreading_Identify_Naming_Logging_threads.php)

(/python/python_numpy_matrix_

Pandas with NumPy and Matplotlib
(/python/python_Pandas_NumPy

Celluar Automata
(/python/python_cellular_automa

Batch gradient descent algorithm
(/python/python_numpy_batch_g

Longest Common Substring Algorithm
(/python/python_longest_commo

Python Unit Test - TDD using unittest.TestCase class
(/python/python_unit_testing.ph

Simple tool - Google page ranking by keywords
(/python/python_site_page_ranki

Google App Hello World
(/python/GoogleApp/python_Goo

Google App webapp2 and WSGI
(/python/GoogleApp/python_Goo

Uploading Google App Hello World
(/python/GoogleApp/python_Goo

Python 2 vs Python 3
(/python/python_differences_Pyt

virtualenv and virtualenvwrapper
(/python/python_virtualenv_virtu

Uploading a big file to AWS S3 using boto module
(/DevOps/AWS/aws_S3_uploadin

Scheduled stopping and starting an AWS instance
(/DevOps/AWS/aws_stopping_sta

Cloudera CDH5 - Scheduled stopping and starting services
(/Hadoop/BigData_hadoop_CDH

Python tutorial

Removing Cloud Files - Rackspace API with curl and subprocess (/python/python_Rackspace_API_

Checking if a process is running/hanging and stop/run a scheduled task on Windows (/python/python-Windows-Check-if-a-Process-is-Running-Hanging-Schtasks-Run-Stop.php)

Apache Spark 1.3 with PySpark (Spark Python API) Shell (/Hadoop/BigData_hadoop_Apac

Apache Spark 1.2 Streaming (/Hadoop/BigData_hadoop_Apac

bottle 0.12.7 - Fast and simple WSGI-micro framework for small web-applications ... (/python/Bottle/Python_Bottle_Fr

Flask app with Apache WSGI on Ubuntu14/CentOS7 ... (/python/Flask/Python_Flask_Blog

Selenium WebDriver (/python/python_Selenium_WebD

Fabric - streamlining the use of SSH for application deployment (/python/Fabric/python_Fabric.ph

Ansible Quick Preview - Setting up web servers with Nginx, configure enviroments, and deploy an App (/DevOps/Ansible/Ansible_Setting

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networ

NLP - NLTK (Natural Language Toolkit) ... (/python/NLTK/NLTK_install.php)

RabbitMQ(Message broker server) and Celery(Task queue) ...

(/python/RabbitMQ_Celery/pytho

OpenCV3 and Matplotlib ... (/python/OpenCV_Python/pythor

Simple tool - Concatenating slides using FFmpeg ... (/FFMpeg/ffmpeg_fade_in_fade_c

iPython - Signal Processing with NumPy (/python/OpenCV_Python/pythor

iPython and Jupyter - Install Jupyter, iPython Notebook, drawing with Matplotlib, and publishing it to Github (/python/IPython/IPython_Jupyte

iPython and Jupyter Notebook with Embedded D3.js (/python/IPython/iPython_Jupyte

Downloading YouTube videos using youtube-dl embedded with Python (/VideoStreaming/YouTube/youtu dl-embedding.php)

Machine Learning : scikit-learn ... (/python/scikit-learn/scikit_machine_learning_Su

Django 1.6/1.8 Web Framework ... (/python/Django/Python_Django_

Exceptions (/python/python_try_except_finally_raise_syntax_error.php)

@static method vs class method (/python/python_differences_between_static_method_and_class_method_instance_method.php)

Private attributes and private methods (/python/python_private_attributes_methods.php)

bits, bytes, bitstring, and constBitStream (/python/python_bits_bytes_bitstring_constBitStream.php)

json.dump(s) and json.load(s) (/python/python-json-dumps-loads-file-read-write.php)

Python Object Serialization - pickle and json (/python/python_serialization_pickle_json.php)

Python Object Serialization - yaml and json (/python/python_yaml_json_conversion.php)

Priority queue and heap queue data structure (/python/python_PriorityQueue_heapq_Data_Structure.php)

Graph data structure (/python/python_graph_data_structures.php)

Dijkstra's shortest path algorithm (/python/python_Dijkstras_Shortest_Path_Algorithm.php)

Prim's spanning tree algorithm (/python/python_Prims_Spanning_Tree_Data_Structure.php)

Closure (/python/python_closure.php)

Functional programming in Python (/python/python_functional_programming.php)

Remote running a local file using ssh (/python/python_ssh_remote_run.php)

SQLite 3 - A. Connecting to DB, create/drop table, and insert data into a table (/python/python_sqlite_connect_create_drop_table.php)

SQLite 3 - B. Selecting, updating and deleting data (/python/python_sqlite_select_update_delete.php)

MongoDB with PyMongo I - Installing MongoDB ... (/python/MongoDB_PyMongo/python_MongoDB_pyMongo_tutorial_installing.php)

Python HTTP Web Services - urllib, httplib2 (/python/python_http_web_services.php)

Web scraping with Selenium for checking domain availability (/python/python_Web_scraping_with_selenium_for_domain_availability.php)

REST API : Http Requests for Humans with Flask (/python/python-REST-API-Http-Requests-for-Humans-with-Flask.php)

Blog app with Tornado (/python/Tornado/Python_Tornado_Blog_App.php)

Multithreading ... (/python/Multithread/python_multithreading_creating_threads.php)

Python Network Programming I - Basic Server / Client : A Basics (/python/python_network_programming_server_client.php)

Python Network Programming I - Basic Server / Client : B File Transfer (/python/python_network_programming_server_client_file_transfer.php)

Python Network Programming II - Chat Server / Client (/python/python_network_programming_tcp_server_client_chat_server_chat_client_select.php)

Python Network Programming III - Echo Server using socketserver network framework (/python/python_network_programming_socketserver_framework_for_network_servers.php)

Python Network Programming IV - Asynchronous Request Handling : ThreadingMixIn and ForkingMixIn (/python/python_network_programming_socketserver_framework_for_network_servers_asynchronous_request_ThreadingMixIn_ForkingMix

Python Coding Questions I (/python/python_interview_questions.php)

Python Coding Questions II (/python/python_interview_questions_2.php)

Python Coding Questions III (/python/python_interview_questions_3.php)

Python Coding Questions IV (/python/python_interview_questions_4.php)

Python Coding Questions V (/python/python_interview_questions_5.php)

Python Coding Questions VI (/python/python_interview_questions_6.php)

Python Coding Questions VII (/python/python_interview_questions_7.php)

Python Coding Questions VIII (/python/python_interview_questions_8.php)

Image processing with Python image library Pillow (/python/python_image_processing_with_Pillow_library.php)

Python and C++ with SIP (/python/python_cpp_sip.php)

PyDev with Eclipse (/python/pydev_eclipse_plugin_install_python_IDE.php)

Matplotlib (/python/python_matplotlib.php)

Redis with Python (/python/python_redis_with_python.php)

NumPy array basics A (/python/python_numpy_array_tutorial_basic_A.php)

NumPy Matrix and Linear Algebra (/python/python_numpy_matrix_tutorial.php)

Pandas with NumPy and Matplotlib (/python/python_Pandas_NumPy_Matplotlib.php)

Celluar Automata (/python/python_cellular_automata.php)

Batch gradient descent algorithm (/python/python_numpy_batch_gradient_descent_algorithm.php)

Longest Common Substring Algorithm (/python/python_longest_common_substring_lcs_algorithm_generalized_suffix_tree.php)

Python Unit Test - TDD using unittest.TestCase class (/python/python_unit_testing.php)

OpenCV 3 image and video processing with Python

OpenCV 3 with Python (/python/OpenCV_Python/pythor

Image - OpenCV BGR : Matplotlib RGB (/python/OpenCV_Python/pythor

Basic image operations - pixel access (/python/OpenCV_Python/pythor

iPython - Signal Processing with NumPy (/python/OpenCV_Python/pythor

Signal Processing with NumPy I - FFT and DFT for sine, square waves, unitpulse, and random signal (/python/OpenCV_Python/pythor

Signal Processing with NumPy II - Image Fourier Transform : FFT & DFT (/python/OpenCV_Python/pythor

Inverse Fourier Transform of an Image with low pass filter: cv2.idft() (/python/OpenCV_Python/pythor

Image Histogram (/python/OpenCV_Python/pythor

Video Capture and Switching colorspaces - RGB / HSV (/python/OpenCV_Python/pythor

Adaptive Thresholding - Otsu's clustering-based image thresholding (/python/OpenCV_Python/pythor

Simple tool - Google page ranking by keywords (/python/python_site_page_ranking_by_keywords.php)

Google App Hello World (/python/GoogleApp/python_GoogleApp_HelloWorld.php)

Google App webapp2 and WSGI (/python/GoogleApp/python_GoogleApp_WebApp2_WSGI.php)

Uploading Google App Hello World (/python/GoogleApp/python_GoogleApp_Uploading_HelloWorld.php)

Python 2 vs Python 3 (/python/python_differences_Python2_vs_Python3_port.php)

virtualenv and virtualenvwrapper (/python/python_virtualenv_virtualenvwrapper.php)

Uploading a big file to AWS S3 using boto module (/DevOps/AWS/aws_S3_uploading_large_file.php)

Scheduled stopping and starting an AWS instance (/DevOps/AWS/aws_stopping_starting_instances.php)

Cloudera CDH5 - Scheduled stopping and starting services (/Hadoop/BigData_hadoop_CDH5_stop_start_services.php)

Removing Cloud Files - Rackspace API with curl and subprocess (/python/python_Rackspace_API_curl_subprocess_Cloud_Files.php)

Checking if a process is running/hanging and stop/run a scheduled task on Windows (/python/python-Windows-Check-if-a-Process-is-Running-Hanging-Schtasks-Run-Stop.php)

Apache Spark 1.3 with PySpark (Spark Python API) Shell (/Hadoop/BigData_hadoop_Apache_Spark_PySpark.php)

Apache Spark 1.2 Streaming (/Hadoop/BigData_hadoop_Apache_Spark_Streaming.php)

bottle 0.12.7 - Fast and simple WSGI-micro framework for small web-applications ... (/python/Bottle/Python_Bottle_Framework.php)

Flask app with Apache WSGI on Ubuntu14/CentOS7 ... (/python/Flask/Python_Flask_Blog_App_Production_with_MongoDB_and_Apache_WSGI.php)

Selenium WebDriver (/python/python_Selenium_WebDriver.php)

Fabric - streamlining the use of SSH for application deployment (/python/Fabric/python_Fabric.php)

Ansible Quick Preview - Setting up web servers with Nginx, configure enviroments, and deploy an App (/DevOps/Ansible/Ansible_SettingUp_Webservers_Nginx_Install_Env_Configure_Deploy_App.php)

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networks_Backpropagation_for_XOR_using_one_hidden_layer.php)

NLP - NLTK (Natural Language Toolkit) ... (/python/NLTK/NLTK_install.php)

RabbitMQ(Message broker server) and Celery(Task queue) ... (/python/RabbitMQ_Celery/python_Installing_RabbitMQ_Celery.php)

OpenCV3 and Matplotlib ...

Edge Detection - Sobel and Laplacian Kernels (/python/OpenCV_Python/pythor

Canny Edge Detection (/python/OpenCV_Python/pythor

Hough Transform - Circles (/python/OpenCV_Python/pythor

Watershed Algorithm : Marker-based Segmentation I (/python/OpenCV_Python/pythor

Watershed Algorithm : Marker-based Segmentation II (/python/OpenCV_Python/pythor

Image noise reduction : Non-local Means denoising algorithm (/python/OpenCV_Python/pythor local_Means_Denoising_Algorithr

Image object detection : Face detection using Haar Cascade Classifiers (/python/OpenCV_Python/pythor

Image segmentation - Foreground extraction Grabcut algorithm based on graph cuts (/python/OpenCV_Python/pythor

Image Reconstruction - Inpainting (Interpolation) - Fast Marching Methods (/python/OpenCV_Python/pythor

Video : Mean shift object tracking (/python/OpenCV_Python/pythor

Machine Learning : Clustering - K-Means clustering I (/python/OpenCV_Python/pythor Means_Clustering_Vector_Quanti

Machine Learning : Clustering - K-Means clustering II (/python/OpenCV_Python/pythor Means_Clustering_Vector_Quanti

Machine Learning :

(/python/OpenCV_Python/python_opencv3_matplotlib_rgb_brg_image_load_display_save.php)

Simple tool - Concatenating slides using FFmpeg ... (/FFMpeg/ffmpeg_fade_in_fade_out_transitions_effects_filters_slideshow_concat.php)

iPython - Signal Processing with NumPy (/python/OpenCV_Python/python_opencv3_NumPy_Arrays_Signal_Processing_iPython.php)

iPython and Jupyter - Install Jupyter, iPython Notebook, drawing with Matplotlib, and publishing it to Github (/python/IPython/IPython_Jupyter_Install_iPython_Notebook_Matplotlib_Publishing_it_to_Github.php)

iPython and Jupyter Notebook with Embedded D3.js (/python/IPython/iPython_Jupyter_Notebook_with_Embedded_D3.php)

Downloading YouTube videos using youtube-dl embedded with Python (/VideoStreaming/YouTube/youtube-dl-embedding.php)

Machine Learning : scikit-learn ... (/python/scikit-learn/scikit_machine_learning_Supervised_Learning_Unsupervised_Learning.php)

Django 1.6/1.8 Web Framework ... (/python/Django/Python_Django_tutorial_introduction.php)

Classification - k-nearest neighbors (k-NN) algorithm (/python/OpenCV_Python/pythor nearest_neighbors_k-NN.php)

Machine Learning with scikit-learn

scikit-learn installation (/python/scikit-learn/scikit-learn_install.php)

scikit-learn : Features and feature extraction - iris dataset (/python/scikit-learn/scikit_machine_learning_fe

scikit-learn : Machine Learning Quick Preview (/python/scikit-learn/scikit_machine_learning_qu

scikit-learn : Data Preprocessing I - Missing / Categorical data (/python/scikit-learn/scikit_machine_learning_Da Missing-Data-Categorical-Data.php)

scikit-learn : Data Preprocessing II - Partitioning a dataset / Feature scaling / Feature Selection / Regularization (/python/scikit-learn/scikit_machine_learning_Da II-Datasets-Partitioning-Feature-scaling-Feature-Selection-Regularization.php)

scikit-learn : Data Preprocessing III - Dimensionality reduction vis Sequential feature selection / Assessing feature importance via random forests (/python/scikit-

learn/scikit_machine_learning_Da
III-Dimensionality-reduction-
via-Sequential-feature-
selection-Assessing-feature-
importance-via-random-
forests.php)

Data Compression via
Dimensionality Reduction I -
Principal component analysis
(PCA) (/python/scikit-
learn/scikit_machine_learning_Da
_PCA.php)

scikit-learn : Data Compression
via Dimensionality Reduction II
- Linear Discriminant Analysis
(LDA) (/python/scikit-
learn/scikit_machine_learning_Da

scikit-learn : Data Compression
via Dimensionality Reduction III
- Nonlinear mappings via
kernel principal component
(KPCA) analysis (/python/scikit-
learn/scikit_machine_learning_Da
nonlinear-mappings-via-kernel-
principal-component-
analysis.php)

scikit-learn : Logistic
Regression, Overfitting &
regularization (/python/scikit-
learn/scikit-
learn_logistic_regression.php)

scikit-learn : Supervised
Learning & Unsupervised
Learning - e.g. Unsupervised
PCA dimensionality reduction
with iris dataset (/python/scikit-
learn/scikit_machine_learning_Su

scikit-learn :
Unsupervised_Learning -
KMeans clustering with iris
dataset (/python/scikit-
learn/scikit_machine_learning_Un

scikit-learn : Linearly Separable
Data - Linear Model &
(Gaussian) radial basis function
kernel (RBF kernel)
(/python/scikit-

learn/scikit_learn_Support_Vecto

MACHINE LEARNING ALGORITHMS AND CONCEPTS

Batch gradient descent algorithm (/python/python_numpy_batch_g

Single Layer Neural Network - Perceptron model on the Iris dataset using Heaviside step activation function (/python/scikit-learn/Perceptron_Model_with_Iri

Batch gradient descent versus stochastic gradient descent (/python/scikit-learn/scikit-learn_batch-gradient-descent-versus-stochastic-gradient-descent.php)

Single Layer Neural Network - Adaptive Linear Neuron using linear (identity) activation function with batch gradient descent method (/python/scikit-learn/Single-Layer-Neural-Network-Adaptive-Linear-Neuron.php)

Single Layer Neural Network : Adaptive Linear Neuron using linear (identity) activation function with stochastic gradient descent (SGD) (/python/scikit-learn/Single-Layer-Neural-Network-Adaptive-Linear-Neuron-with-Stochastic-Gradient-Descent.php)

Logistic Regression (/python/scikit-learn/logistic_regression.php)

VC (Vapnik-Chervonenkis)

Dimension and Shatter (/python/scikit-learn/scikit_machine_learning_VC

Bias-variance tradeoff (/python/scikit-learn/scikit_machine_learning_Bi variance-Tradeoff.php)

Maximum Likelihood Estimation (MLE) (/python/scikit-learn/Maximum-Likelyhood-Estimation-MLE.php)

Neural Networks with backpropagation for XOR using one hidden layer (/python/python_Neural_Networ

minHash (/Algorithms/minHash_Jaccard_Si

tf-idf weight (/Algorithms/tf_idf_term_frequen

Natural Language Processing (NLP): Sentiment Analysis I (IMDb & bag-of-words) (/Algorithms/Machine_Learning_I

Natural Language Processing (NLP): Sentiment Analysis II (tokenization, stemming, and stop words) (/Algorithms/Machine_Learning_I

Natural Language Processing (NLP): Sentiment Analysis III (training & cross validation) (/Algorithms/Machine_Learning_I

Natural Language Processing (NLP): Sentiment Analysis IV (out-of-core) (/Algorithms/Machine_Learning_I

Locality-Sensitive Hashing (LSH) using Cosine Distance (Cosine Similarity) (/Algorithms/Locality_Sensitive_H

ARTIFICIAL NEURAL NETWORKS (ANN)

[Note] Sources are available at Github - Jupyter notebook files (https://github.com/Einsteinish/A
Neural-Networks-with-Jupyter.git)

Golden Gate Ave, San Francisco, CA 94115