Web_Server

DHEERAJ SHARMA

Reg.no -11802178

Artificial Intelligence

Project Name : Creating web server using machine learning

Library + Some Important Tools

- Pandas
- Numpy
- Keras
- Tensorflow
- Theano
- Matplotlib
- Json
- Csv file
- jupyter

Pandas :

pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, **real world** data analysis in Python. Additionally, it has the broader goal of becoming **the most powerful and flexible open source data analysis / manipulation tool available in any language**. It is already well on its way toward this goal.

Pandas is well suited for many different kinds of data:

- Tabular data with heterogeneously-typed columns as in an SQL table or Excel spreadsheet
- Ordered and unordered time series data
- Arbitrary matrix data with row and column labels
- Any other from of observation/statistical data sets. The data actually need not be labeled at all to be placed into a pandas data structure

The two primary data structures of pandas, Series (1-dimensional) and DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, social science, and many areas of engineering. For R users, DataFrame provides everything that R's `data.frame` provides and much more. pandas is built on top of NumPy and is intended to integrate well within a scientific computing environment with many other 3rd party libraries.

Install :

Installing pandas and the rest of the NumPy and SciPy stack can be a little difficult for inexperienced users.

The simplest way to install not only pandas, but Python and the most popular packages that make up the SciPy stack (IPython, NumPy, Matplotlib, …) is with Anaconda, a cross-platform (Linux, Mac OS X, Windows) Python distribution for data analytics and scientific computing.

After running the installer, the user will have access to pandas and the rest of the SciPy stack without needing to install anything else, and without needing to wait for any software to be compiled.

Installation instructions for Anaconda can be found here.

A full list of the packages available as part of the Anaconda distribution can be found here.

Another advantage to installing Anaconda is that you don't need admin rights to install it. Anaconda can install in the user's home directory, which makes it trivial to delete Anaconda if you decide (just delete that folder).

On Windows the command is:

```
Pip install pandas
```

NUMPY :

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data.

It provides:

- 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
- and much more

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.

All NumPy wheels distributed on PyPI are BSD licensed.

KERAS :

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. *Being able to go from idea to result with the least possible delay is key to doing good research.*

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: **Python 2.7-3.6**.

Multi-backend Keras and tf.keras:

At this time, we recommend that Keras users who use multi-backend Keras with the TensorFlow backend switch to `tf.keras` in TensorFlow 2.0. `tf.keras` is better maintained and has better integration with TensorFlow features (eager execution, distribution support and other).

Keras 2.2.5 was the last release of Keras implementing the 2.2.* API. It was the last release to only support TensorFlow 1 (as well as Theano and CNTK).

The current release is Keras 2.3.0, which makes significant API changes and add support for TensorFlow 2.0. The 2.3.0 release will be the last major release of multi-backend Keras. Multi-backend Keras is superseded by `tf.keras`.

Bugs present in multi-backend Keras will only be fixed until April 2020 (as part of minor releases).

For more information about the future of Keras, see the Keras meeting notes.

Guiding principles

- **User friendliness.** Keras is an API designed for human beings, not machines. It puts user experience front and center. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear and actionable feedback upon user error.
- **Modularity.** A model is understood as a sequence or a graph of standalone, fully configurable modules that can be plugged together with as few restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions and regularization schemes are all standalone modules that you can combine to create new models.
- **Easy extensibility.** New modules are simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.

- **Work with Python**. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.

Tensorflow:

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications.

TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well.

TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages.

Keep up-to-date with release announcements and security updates by subscribing to announce@tensorflow.org. See all the mailing lists.

Install :

See the TensorFlow install guide for the pip package, to enable GPU support, use a Docker container, and build from source.

To install the current release, which includes support for CUDA-enabled GPU cards *(Ubuntu and Windows)*:

```
$ pip install tensorflow
```

A smaller CPU-only package is also available:

```
$ pip install tensorflow-cpu
```

To update TensorFlow to the latest version, add `--upgrade` flag to the above commands.
Nightly binaries are available for testing using the tf-nightly and tf-nightly-cpu packages on PyPi.

Try your first TensorFlow program

```
$ python
>>> import tensorflow as tf
```

```
>>> tf.add(1, 2).numpy()
3
>>> hello = tf.constant('Hello, TensorFlow!')
>>> hello.numpy()
b'Hello, TensorFlow!'
```

Theano:

Theano is a Python library that allows you to define, optimize, and evaluate mathematical expressions involving multi-dimensional arrays efficiently. Theano features:

- **tight integration with NumPy** – Use *numpy.ndarray* in Theano-compiled functions.
- **transparent use of a GPU** – Perform data-intensive computations much faster than on a CPU.
- **efficient symbolic differentiation** – Theano does your derivatives for functions with one or many inputs.
- **speed and stability optimizations** – Get the right answer for `log(1+x)` even when `x` is really tiny.
- **dynamic C code generation** – Evaluate expressions faster.
- **extensive unit-testing and self-verification** – Detect and diagnose many types of errors.

Theano has been powering large-scale computationally intensive scientific investigations since 2007. But it is also approachable enough to be used in the classroom (University of Montreal's deep learning/machine learning classes).

Download

Theano is now available on PyPI, and can be installed via `easy_install Theano`, `pip install Theano` or by downloading and unpacking the tarball and typing `python setup.py install`.

Those interested in bleeding-edge features should obtain the latest development version, available via:

```
git clone git://github.com/Theano/Theano.git
```

You can then place the checkout directory on your `$PYTHONPATH` or use `python setup.py develop` to install a `.pth` into your `site-packages` directory, so that

when you pull updates via Git, they will be automatically reflected the “installed” version. For more information about installation and configuration, see installing Theano.

Matplotlib : is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

Matplotlib 2.0.x supports Python versions 2.7 through 3.6. Python 3 support started with Matplotlib 1.2. Matplotlib 1.4 is the last version to support Python 2.6. Matplotlib has pledged to not support Python 2 past 2020 by signing the Python 3 Statement.

JSON:

(JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.
JSON is built on two structures:
A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.
These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.
In JSON, they take on these forms:
An *object* is an unordered set of name/value pairs. An object begins with `{` *left brace* and ends with `}` *right brace*. Each name is followed by `:` *colon* and the name/value pairs are separated by `,` *comma*.

Motivation:

**Emojifier

Emojifier is the Android application which uses ML kit and classification. I takes the picture from the User's mobile phone and based on his/her expression, run the ML algorithms to get to know about the facial expressions of all the faces in an image and then places the appropriate emoji on each face. It also allows you to save and share the new emojified image.

Goal of the project

We will be building an Emojifier by using word vector representations. Our model will take an input sentence and find the most appropriate emoji to be used with this sentence - from an assortment of 5 emoji's at its disposal.

- Heart
- Baseball
- Smile
- Disappointment
- Fork and Knife

We need to use only the first two columns. First column contains the sentence and the second column contains the Emoji associated with the sentence.