Software Installation - Personal Laptop Setup

David Semedo

The main programming language for this course will be Python. If you are not familiar with the language, the following tutorials are a good starting point:

- Python Tutorial 1
- Python Tutorial 2
- Python Tutorial 3

Python is a general-purpose programming language, that was designed to be simple and easy-to-learn. Therefore, you should be able to quickly grasp the basics and start developing your programs.

Python Versions

There are two main versions: Python 2 (officially deprecated) and Python 3. For this course we encourage you to use Python 3, although you should be fine with both versions. While most syntax and functionality are similar, there are several differences that should not be ignored (some are really subtle).

For more information regarding differences between versions please check the following link: https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html

The Anaconda distribution

We strongly recommend that you use the Anaconda Python distribution. Anaconda is an highly flexible Python distribution, with support for several Python versions. It contains several useful tools like a great package manager and virtual environment creation/management tools. You can see more of its features in the official documentation.

Step 1: Installing Anaconda:
Download the Anaconda distribution from here.
Then, follow the instructions for Linux, Windows or macOS, accordingly to your operating system. During installation you can choose the path where Anaconda will be installed.

Step 2: Update your anaconda distribution.
Run the following command from a shell (open a new shell to make Anaconda changes effective):

- `> conda update conda`

Conda environments

Anaconda allows the creation of Python virtual environments. The main purpose of Python virtual environments is to create an isolated environment for Python projects. This means that each project can have its own dependencies, regardless of what dependencies every other project has.

In Anaconda, virtual environments are referred as conda environments.

To keep both Python and libraries required for the Web Search course isolated from other libraries installed in your computer, let's create a conda environment:

Step 3: Run the following command to create a conda env:

- `> conda create -n <env_name> python=3.6`

where you should replace <env_name> by any name you like. Note that we are specifying the python version. To create a Python 2.7 environment replace 3.6 by 2.7.

Step 4: Activate the environment in your current shell:

- `> source activate <env_name>`

To deactivate an environment, just run the command

- `> source deactivate.`

Required Libraries

For this course, we will need at least the following libraries:

- Numpy
- Scipy
- IPython
- Jupyter
- Scikit-learn
- Scikit-image
- Tensorflow
- Keras
- NLTK
- Gensim
- Pandas

NOTE: Before attempting to install these libraries, make sure your conda environment is activated.

Step 5: Installing the required Python libraries:

- `> conda install pip h5py numpy scipy scikit-learn scikit-image gensim nltk jupyter pandas`

Conda will also install the required libraries' dependencies.

Step 6: Installing tensorflow:
Linux:

- `> pip install --ignore-installed --upgrade https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.10.1-cp36-cp36m-linux_x86_64.whl`

Windows:

- `> pip install --ignore-installed --upgrade tensorflow`

macOS:

- `> pip install --ignore-installed --upgrade https://storage.googleapis.com/tensorflow/mac/cpu/tensorflow-1.10.1-py3-none-any.whl`

Note that for Linux and macOS you have to choose a different .whl link, if you used a different Python version. Choose the correct .whl link from the following links: Linux, macOS.

If you have an NVIDIA GPU, you can install tensorflow with GPU support. Check tensorflow documentation for installation instructions for your platform.

Step 6: Installing keras:

- `> pip install keras`

Step 7: Setup Keras to use tensorflow backend.
Create a Keras config file - Linux and macOS:

- `> mkdir ~/keras && touch ~/keras/keras.json`

Create a Keras config file - Windows:

- `> mkdir c:\users\<your_username>\.keras`
- `> cd  c:\users\<your_username>\.keras\`
- `> type nul >> keras.json`

Open the created file and paste:

```
{
   "floatx": "float32",
   "epsilon": 1e-07,
   "backend": "tensorflow",
   "image_data_format": "channels_last"
}
```

Save and exit.

Step 7: Test if Keras is using the tensorflow backend:

- `> python`
- `>>> import keras`

It should print the following message: `Using TensorFlow backend.`

Software Configuration - Lab Computers

Step 1: Setup Keras to use tensorflow backend.

Create a Keras config file - Linux and macOS:

- `> mkdir ~/.keras && touch ~/.keras/keras.json`

Create a Keras config file - Windows:

- `> mkdir c:\users\<your_username>\.keras`
- `> cd  c:\users\<your_username>\.keras\`
- `> type nul >> keras.json`

Open the created file and paste:

```
{
    "floatx": "float32",
    "epsilon": 1e-07,
    "backend": "tensorflow",
    "image_data_format": "channels_last"
}
```

Save and exit.

Step 2: Test if Keras is using the tensorflow backend:

- `> python`
- `>>> import keras`

Step 3: Setup models path for Keras and NLTK:

- `> export NLTK_DATA=”/opt/websearch/nltk_data”`

(You can add the export command to your .bashrc, such that it gets executed when you login)

- `> ln -s /opt/websearch/keras_models ~/.keras/models`