CS 410

Casey Blair

Final Project Progress Report

11/29/2020

My final project is to update the Metapy GitHub repo to allow for installation with Python 3. Currently, when trying to install Metapy using Pip on both Windows 10 and Mac OSX, if using a Python 3 environment, the setup wheel fails to install the Metapy package due to errors. However, if using a Python 2.7 environment on both Windows 10 and OSX, the Metapy library will install correctly. Since Python 2 was deprecated in January of 2020, updating the Metapy code base to work with Python 3.* environments is vital for using Metapy in production code bases.

I have been researching Pybind11 and how Python can be used with C and C++ code to understand how the Metapy Git repo works. Metapy is a Python wrapper for the Meta Github repo (https://github.com/meta-toolkit/meta). The Meta code base is written in C++, and when the Metapy package is installed with Pip, the Meta C++ code is downloaded and installed then a Python interface is constructed so the code can be used from a Python interpreter.

When installing Metapy in a Python 2.7 environment, the Pip installer downloads a tar.gz file from the Github repo's Releases page with the Metapy code already compiled for a Python 2.7 environment, which is why it can be installed correctly with a Python 2 environment. All the necessary files, including third party libraries, are already downloaded and bundled in the tar.gz file that is then used to install Metapy. However, when installing Metapy using Python 3.*, instead of using this tar.gz file from the GitHub repo's Releases page, the Pip setup wheel builds the Metapy code from scratch, which requires redownloading all third party libraries and compiling the C++ code in the Meta Git submodule. The third party library download step fails because a library the Meta repo uses is failing to download from the URL specified in the Meta Makefile build steps.

The following error is repeated over and over again and the installation ultimately fails after 5 retries of downloading the missing package:

```
-- Downloading...
  dst='/Users/caseyblair/Documents/Classes/CS_410/metapy/deps/meta/deps/icu-58.2/icu4c-58_2-src.tgz'
  timeout='none'
-- Using src='http://download.icu-project.org/files/icu4c/58.2/icu4c-58_2-src.tgz'
-- [download 100% complete]
* Closing connection 2
* Closing connection 0
* Closing connection 1
-- verifying file...
     file='/Users/caseyblair/Documents/Classes/CS_410/metapy/deps/meta/deps/icu-58.2/icu4c-58_2-src.tgz'
-- MD5 hash of
```

```
  /Users/caseyblair/Documents/Classes/CS_410/metapy/deps/meta/deps/icu-58.2/icu4c-58_2-src.tgz
 does not match expected value
   expected: 'fac212b32b7ec7ab007a12dff1f3aea1'
     actual: 'ac2ff460bdda9c70e6ed803f5e4d03fb'
-- Hash mismatch, removing...
-- Retrying...
```

First, I had to determine which errors were occurring on both operating systems (Windows 10 and OSX) to determine if the installation errors were related or operating system dependent. It turns out the same error is occurring on both Windows 10 and OSX which led me to believe the issue was not operating system dependent and that I needed to follow the build steps to determine why this third party library was failing to install.

I had to figure out why the Metapy package was installing correctly for Python 2 but not Python 3 as it didn't make sense that if the C++ code was being compiled on my laptop why it would compile correctly for Python 2 but not Python 3 since the installer would be using the same C++ compiler. Now that I understand that this "icu4c" library is included in the tar.gz file downloaded for Python 2 environments I now I understand that the C++ compile step would fail for both a Python 2 and Python 3 environment because this third party library is unable to be downloaded from this broken link.

After researching why this "icu4c" package was failing to download repeatedly, I found out that the URL used to download the "icu4c-58_2-src.tgz" no longer has this tgz file at this URL. I had to step though the setup.py file's build steps and figure out what exactly the build steps were. The setup.py file from the Metapy repo is actually running a C++ make file that downloads the Meta git repo (https://github.com/meta-toolkit/meta), then runs a Makefile to compile the C++ code for the host machine. The icu4c-58_2-src.tgz file download is performed in the Meta C++ Makefile system, so I need to figure out how to update this file download in this second Git repo.

While fixing this icu4c-58_2-src.tgz is definitely an important part of fixing the issues with the setup.py file's installation issues, it is unclear what further steps I will need to take to fix the Python 3.* installation at this time. I will not know what other steps I will have to take to fix the C++ makefile compile steps since I'm blocked from progressing further though the compilation steps at this time until I can fix the icu4c-58_2-src.tgz file download. However, figuring out this third party download issue is vital to fixing the Python 3 installation issues as the URL currently in the Meta Git repo's Makefile build steps to download the icu4c file no longer works. But it is possible I will run in to operating system dependent issues when the C++ code is able to be compiled. But until I can get to the point where the C++ compiler is able to attempt to compile the code, I have to fix these other issues that are preventing C++ code compilation from even being started.