Overview

StarFlow uses the keywords “depends\_on” and “creates” in user-annotated functions to establish dependencies between files and functions in a python script.

The DDG that is produced has data nodes that correspond to files, and process nodes that correspond to functions in the python script.

Each function must have an input and output file that are defined using the keywords, or else the dependencies are not found.

Code blocks that are not in a function are lost.

Code blocks that do not have an explicit input file or output file should be grouped into a function so that they do have an input file and output file.

However, only the functions are shown, unlike in RDT, where each line the script corresponds to a node.

The main problem seems to be that the style of R programming in code blocks rather than in functions does not translate well into Python and StarFlow, which gets its information at the function level.

How it Works/Basic Usage.

See examples.

Write a python script, making sure that each function has an input and output file. Import csv, time, datetime, pickle, and starflow.make\_json\_prov.

In the main function of the script, call the functions in the workflow in the proper order and record the time\_executed for easier sorting later. Pickle this information for accessing later.

Call starflow.make\_json\_prov.StarFlowLL\_to\_DDG using the script name, the output\_json\_file, and the location of the pickled dictionary.

StarFlowLL\_to\_DDG calls StarFlow.linkmanagement.LinksFromOperations(FileList).

LinksFromOperations finds all the links using the keywords and returns this list as a numpy record array. This array has the fields: LinkType, LinkSource, and LinkTarget. The main internal functions are ComputeLInksFromOperations, GutsComputeLinks, and GetStoredModule.

After obtaining the LL, StarFLowLL\_to\_DDG sorts it depending on the pickled file with the times, and converts it to a Prov-JSON file depending on the link type.

Open the .json files using DDGExplorer.

Basic Installation

Starflow requires Python 3.4 because of its dependency on the distribute package.

Install dependencies and use setup.py install to build the package.

Graphviz is not needed if DDGs are made.

Tabular for Python 3 is included in the github repo.

Scripts to analyze must be in a “data environment” that is set up by typing starflow init name\_of\_environment /path/to/environment.

This registers an environment based on global configurations/templates that are set up, such as configure\_live\_module\_filters (which should contain the scripts directory of the environment) and set\_up\_functions.py. Global copies of these files serve as templates for the local copies, so this should be done first.

In an environment, there is a .starflow directory which stores all of the data collected by SF and the local configuration file.

Make scripts, data, results directories in the environment and store files accordingly.

Detailed Installation

After python setup.py install:

Check the installation in the python shell.

>>> import starflow #no errors

>>> starflow.\_\_file\_\_

'/Users/jen/miniconda3/envs/python3.4/lib/python3.4/site-packages/StarFlow-0.9999-py3.4.egg/starflow/\_\_init\_\_.py'

Go to this folder: /Users/jen/miniconda3/envs/python3.4/lib/python3.4/site-packages/StarFlow-0.9999-py3.4.egg/starflow/

Make changes to:

1. config.py

Function load\_live\_module\_filters(self)

~line 335

FilterPath = “your/path/to/cloned/StarFlow/docs/Samples/config/configure\_live\_module\_filters.txt”

1. metadata.py

~ line 40

sys.path.append(“your/path/to/cloned/StarFlow”)

Set Up Global Templates:

Go back to the git cloned dir/docs/Samples/config

Python initialize\_global\_reg\_file.py

YourStarFlow/docs/Samples/config/PerMachineSetUp.py

Use which python to set paths to python3 dir

After starflow init starflow init name\_of\_environment /path/to/environment:

Instead of using entire initialize, use python /path/to/linkmanagement.py

Make sure it says registering environment, check with listDEs

Put a copy of /docs/Samples/config/setupfunctions.py into data environment

Duplicate a copy of Starflow/docs/Samples/config/SetupFunctions.py into data environment as setupfunctions.py

Make a copy of live\_module\_filters from Samples/docs/config and put it into the local .starflow directory.

Add path to environment/scripts dir.