



AMYA CODERS

204202001

SEMANTIC CODE SEARCH -1

Instruction Manual

Author:
Atul Sahay

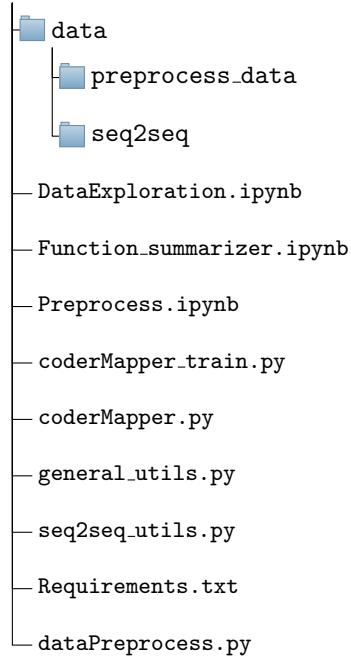
April 5, 2020

Contents

1	Project's Directory Structure	2
2	Requirements	3
2.1	How to install required modules for the project?	3
3	Data Preprocessing	4
4	Code and Doc-string Mapper	5
4.1	Pre-requisite	5
4.2	To Train the codeMapper Model	5
4.3	Code Mapper Model Inference	5

1 Project's Directory Structure

Semantic Code Matcher

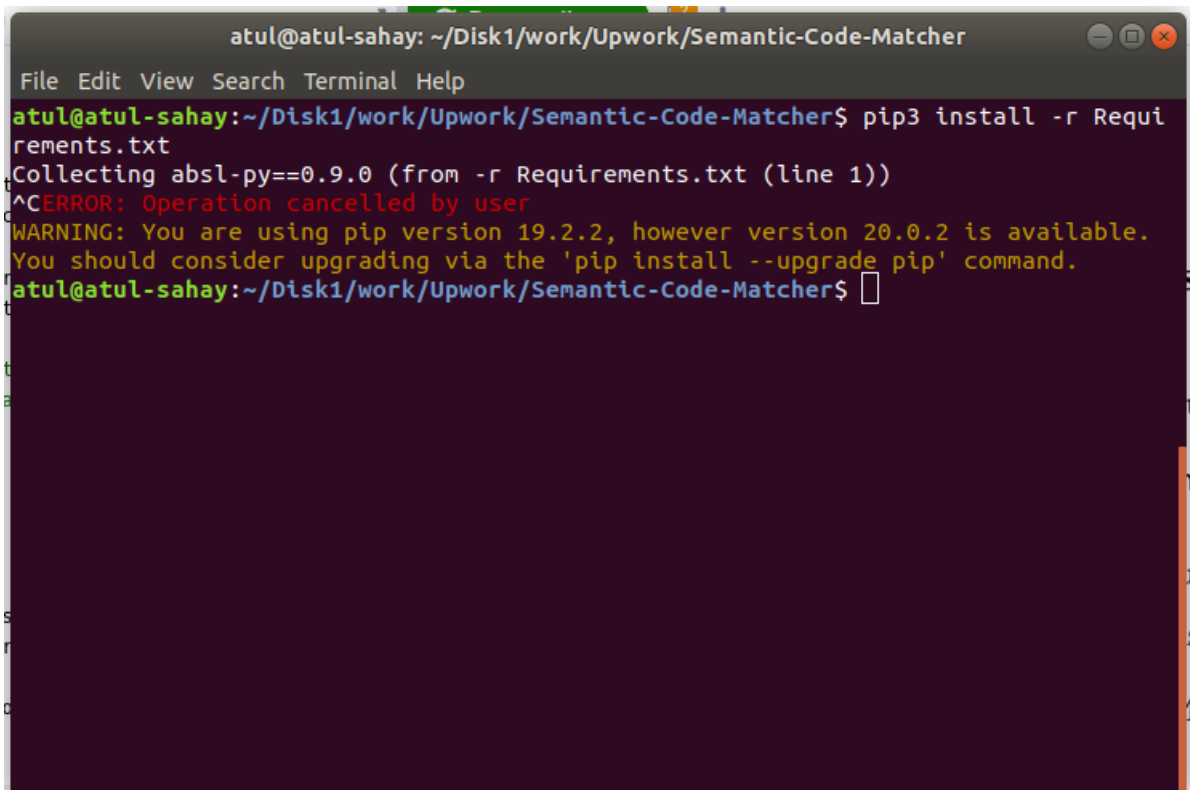


2 Requirements

2.1 How to install required modules for the project?

- Search for the **Requirements.txt** file in the project repository.
- Open the terminal and write (be sure you are in the project directory): See Fig: 2.1

user: Semantic-Code-Matcher\$: **pip3 install -r Requirements.txt**



```
atul@atul-sahay: ~/Disk1/work/Upwork/Semantic-Code-Matcher
File Edit View Search Terminal Help
atul@atul-sahay:~/Disk1/work/Upwork/Semantic-Code-Matcher$ pip3 install -r Requirements.txt
Collecting absl-py==0.9.0 (from -r Requirements.txt (line 1))
^CERROR: Operation cancelled by user
WARNING: You are using pip version 19.2.2, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
atul@atul-sahay:~/Disk1/work/Upwork/Semantic-Code-Matcher$
```

Figure 1: Installation of the required modules

3 Data Preprocessing

For the data loading and data preprocessing, you need to follow the given mentioned steps:

1. First you need to create a directory structure: as provided in section 1 for "data" and "preprocess_data". Here the data will be stored.
2. Now just run the file name "dataPreprocess.py"

```
-- > $ python3 dataPreprocess.py
```

After these steps please run these command and check whether you are getting the similar directory structure or not.

```
-- > $ ls -lah ./data/processed_data/
```

```
In [25]: !ls -lah ./data/processed_data/
total 2.6G
drwxrwxr-x 2 ritesh ritesh 4.0K Apr  4 05:22 .
drwxrwxr-x 3 ritesh ritesh 4.0K Apr  4 05:16 ..
-rw-rw-r-- 1 ritesh ritesh 16M Apr  4 05:19 test.docstring
-rw-rw-r-- 1 ritesh ritesh 55M Apr  4 05:19 test.function
-rw-rw-r-- 1 ritesh ritesh 18M Apr  4 05:19 test.lineage
-rw-rw-r-- 1 ritesh ritesh 25M Apr  4 05:19 test_original_function.json.gz
-rw-rw-r-- 1 ritesh ritesh 70M Apr  4 05:18 train.docstring
-rw-rw-r-- 1 ritesh ritesh 308M Apr  4 05:17 train.function
-rw-rw-r-- 1 ritesh ritesh 86M Apr  4 05:19 train.lineage
-rw-rw-r-- 1 ritesh ritesh 140M Apr  4 05:18 train_original_function.json.gz
-rw-rw-r-- 1 ritesh ritesh 16M Apr  4 05:19 valid.docstring
-rw-rw-r-- 1 ritesh ritesh 69M Apr  4 05:19 valid.function
-rw-rw-r-- 1 ritesh ritesh 19M Apr  4 05:19 valid.lineage
-rw-rw-r-- 1 ritesh ritesh 31M Apr  4 05:19 valid_original_function.json.gz
-rw-rw-r-- 1 ritesh ritesh 1.1G Apr  4 05:19 without_docstrings.function
-rw-rw-r-- 1 ritesh ritesh 344M Apr  4 05:22 without_docstrings.lineage
-rw-rw-r-- 1 ritesh ritesh 356M Apr  4 05:22 without_docstrings_original_function.json.gz
```

Figure 2: Directory structure of data/preprocess_data folder

4 Code and Doc-string Mapper

4.1 Pre-requisite

Make Sure you have the right files prepared from Step dataPreprocess

1. You should have these files in the root of the `./data/processed_data/` directory:
 - (a) `{train/valid/test.function}` - these are python function definitions tokenized (by space), 1 line per function.
 - (b) `{train/valid/test.docstring}` - these are docstrings that correspond to each of the python function definitions, and have a 1:1 correspondence with the lines in `*.function` files.
 - (c) `{train/valid/test.lineage}` - every line in this file contains a link back to the original location (github repo link) where the code was retrieved. There is a 1:1 correspondence with the lines in this file and the other two files. This is useful for debugging.
2. **Set the value of `use_cache` appropriately.**

In the file: `codeMapper`, if `use_cache = True`, preprocessed data will be downloaded where possible from the `blog googleapis` link instead of re-computing. However, it is highly recommended that you set `use_cache = False`
3. Make sure the `data/seq2seq` directory exists see the Section 1.

4.2 To Train the codeMapper Model

Run the command specified below

```
$ python3 codeMapper_train.py
```

After the completion of training you will find a `"code_summary_seq2seq_model.h5"` in `data/seq2seq` directory. see Section 1 for the directory structure.

4.3 Code Mapper Model Inference

You need to get ready with the model file stored in `data/seq2seq` folder or you can download the pretrained model file from the internet.

Further you need to provide the `"input_file"` that contains the code text one code per line and `"output_file"` where corresponding code and docstring can be stored.

To run the command:

```
$ python3 Semantic-Code-Matcher/codeMapper.py -d True -I input_file -O out.csv
```

1. `-d[Optional]` if `True`, pretrained model will downloaded from the internet . `False` local saved model will be used
2. `-I[Required]` A text file containing the raw code segments one per line.
3. `-O[Required]` A csv file containing the code,docstring pairs