Provenance based recommender system for CONP pipelines and datasets

Mandana Mazaheri

40086407

COMP 6971 Project and Report

Concordia University

Professor Tristan Glatard

Introduction

This project aims to propose some helpful aspects to CONP (Canadian Open Neuroscience Platform) as a provenance-based recommendation system to be able to recommend pipelines to datasets and datasets to pipelines in CONP. Meaning that if pipeline A has processed dataset X and Y successfully, and pipeline B has processed dataset Y and Z successfully, then pipeline A and dataset Z would be recommended to each other and pipeline B and dataset X would be recommended to each other, more details are available in following.

This recommendation is considerably helpful regarding the fact that it will address the issue of finding applicable pipelines and datasets. It provides a list of recommended pipelines to users and scientists who are willing to find appropriate pipelines for a specific dataset to be run, also, provides a list of recommended datasets when finding applicable datasets for a candidate pipeline is the issue.

As far as Intellectual property rights concerns, all implementation parts of this project is completely done by myself except for following parts. Parsing provenance records which was a joint development by Aida Vahdani and I, boutiques modification and dashboard development which is completely implemented by her. It would be possible to use those boutiques commands after finalizing her modifications. Also, the results of my project will be illustrated on the dashboard she has designed.

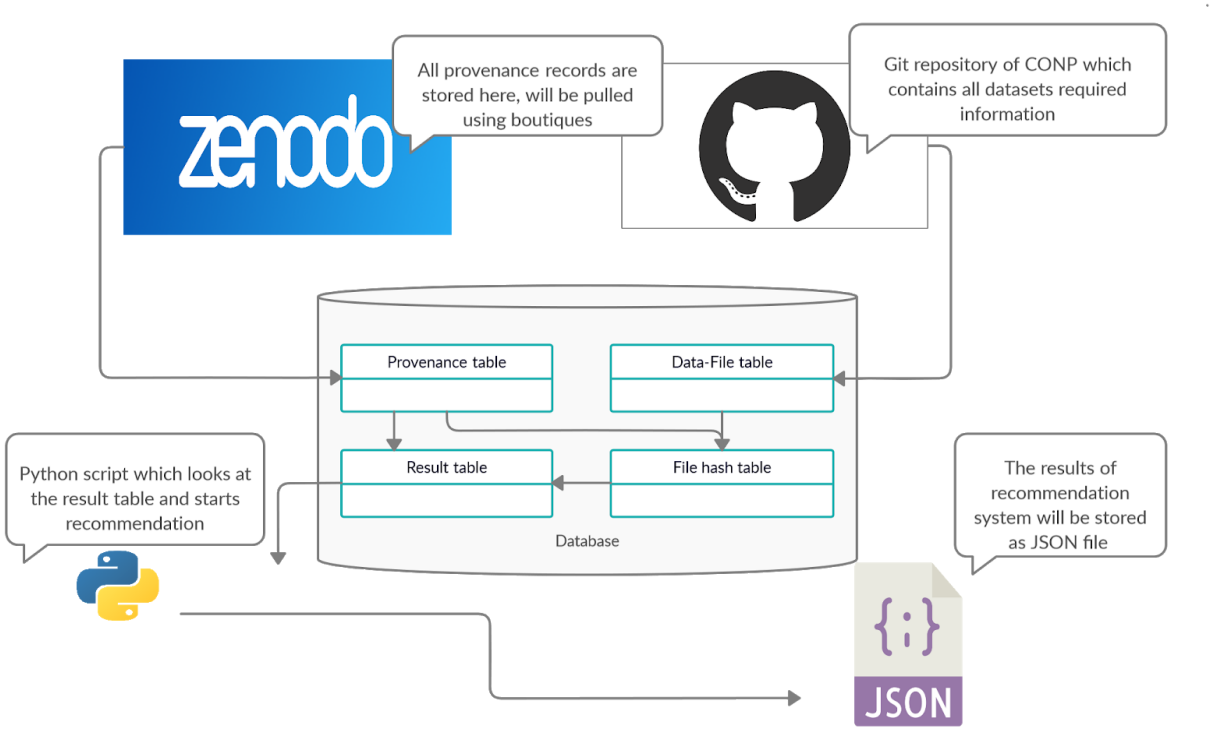
Methods

The general approach was to

1. Getting familiar with all required steps and design software architecture for that
2. Develop crawlers to pull CONP datasets and fill associated tables in the database (required investigations)
3. Extract file hashes from each dataset (much investigation on this phase)
4. Match datasets and pipelines through these file hashes
5. Put all extracted information in a database to employ in the recommendation system
6. Create a model to recommend based on matched datasets and pipeline using hashes

7. Test the model and make the system more efficient and platform-independent

After much investigation which will be explained in this report, the final approach made the project much simpler to develop and implement. The following diagram, 1- General approach of implementation, illustrates the main idea of how this development has been done. All required information for datalad datasets are available in CONP github, there are annexed files available for each file in each dataset. Each annexed file contains symbolic link (sym-link) to the exact repository or directory in which the data file exists, and particularly the MD5E of the file which is exactly a hash of file contents [1]. Data-file table is filled with such information for each file in each dataset.



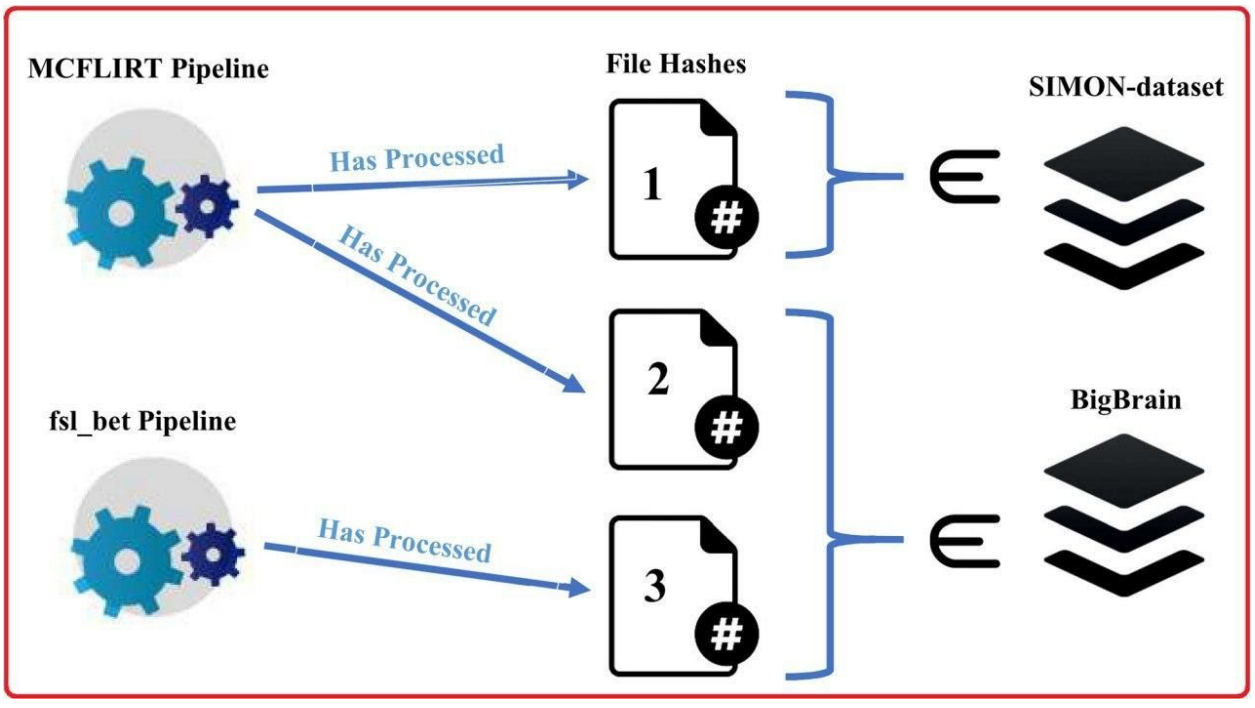
1- General approach of implementation

Furthermore, considering the aim of this project, I needed to access all boutiques provenance records for CONP, available in Zenodo, which is done by boutiques pull command (since 'bosh data pull' command for boutiques is not published yet, getting provenance records is handled manually as a temporary solution). Storing, reading, extracting required information for each provenance record is done by some python scripts, and finally the provenance table is filled. When comparing the MD5E hash for input files in each provenance record with the MD5E

hashes available in the data-file table, the interesting matched results will be provided, which pipeline has run one or more files from which dataset, employing such outcomes, the result table will be filled appropriately. Moreover, considering links between datasets and pipelines, extracted from the result table, I could develop a recommendation system which recommends a list of pipelines to a specific dataset or a list of datasets to a specific pipeline. These results are stored in JSON files to be readable in the dashboard of CONP.

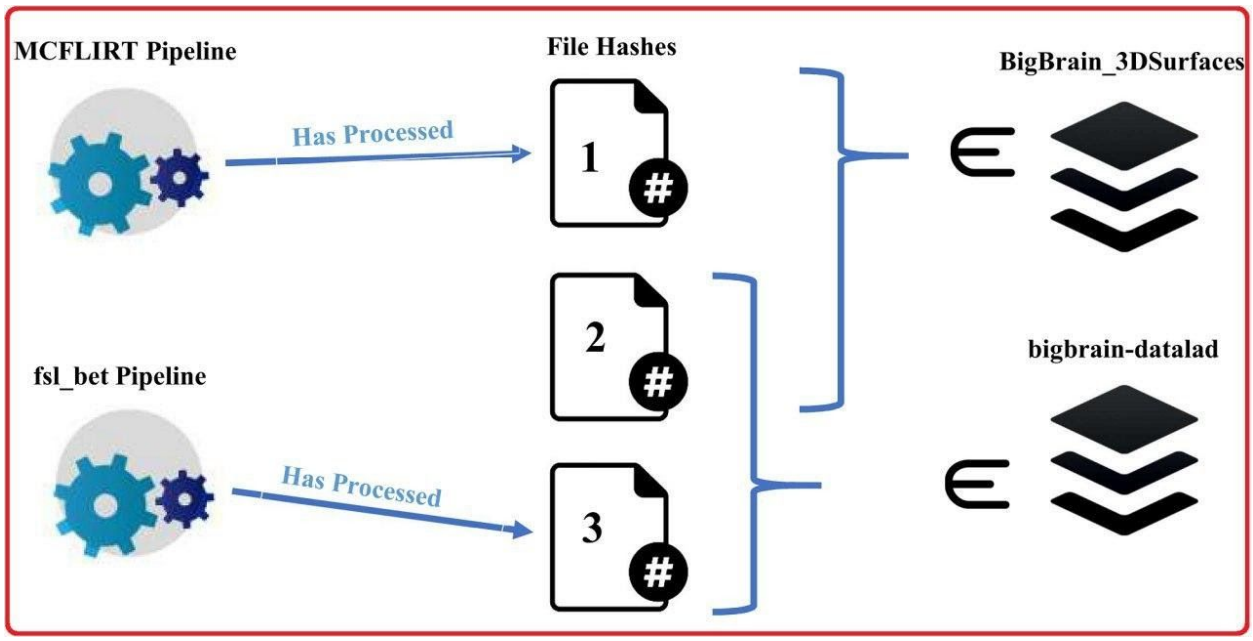
Finally, the most interesting part of this project is how this recommender works, it seems like a 3-partite graph. First layer contains pipeline nodes, the second layer contains file nodes and the third layer contains dataset nodes. The edge which connects a pipeline node to a file node indicates a successful run (pipeline has run this file successfully), the edge which connects a file node to a dataset node illustrates that the file is part of the dataset. Therefore, the path from a pipeline to a dataset through a file tells that there is at least one provenance record containing a successful run for this pipeline and this dataset. More specifically, the key to connect a dataset to a pipeline is the MD5E hash of the file (in that dataset) which is run by that pipeline.

Currently, there are two reasons to recommend pipelines and datasets, first, when there is at least one successful run which is in common between pipelines or datasets. For instance, in figure 2-Successful runs in common , MCFLIRT has run successfully some files in SIMON and BigBrain datasets, FSL-bet also, has run a file in BigBrain dataset, therefore, SIMON dataset will be added to recommendation list for FSL-bet, and FSL-bet will be added to SIMON's list.



2- Successful runs in common

Another important result of investigations on files and datasets, there might be some duplicated files in more than one dataset, 3- Same files in datasets, particularly in BigBrain datasets. In such cases, those datasets will be in the same group and their recommendation list will be joint together. For example, since BigBrain_3dSurface and bigbrain-datalad have one file in common, #2, MCFLIRT and FSL-bet are in their recommendation lists.



3- Same files in datasets

Helpful investigation

In first skim of project, in order to match datasets and pipelines through file hashes, there was two options, first, to download all datasets on a server, compute hash for each file and fill a database table (same as data-file table in current version), and second, modify boutiques to provide users with information about datasets. The first solution was time and memory consuming and clearly inefficient, the second one was a solution only applicable for provenance records which would have been created by boutiques in future, therefore, we would have lost currently available provenance records on Zenodo.

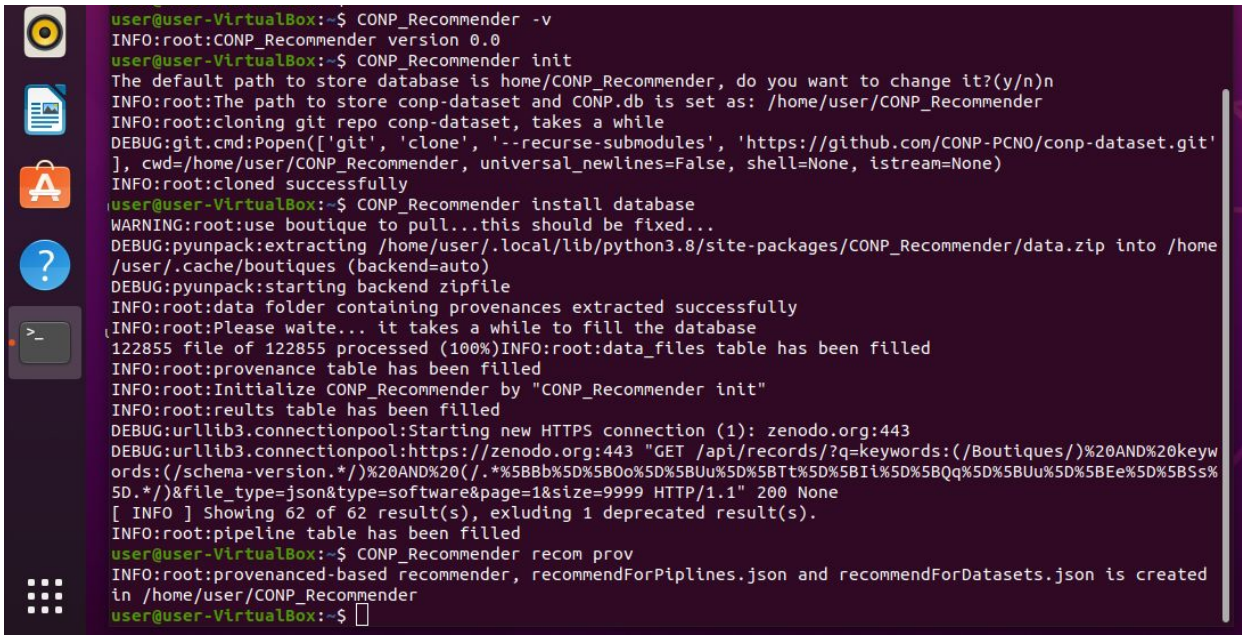
After reading about git-annex, datalad [2], and considering carefully annexed sym-links in CONP github, I surprisingly found that the MD5E in sym-link for each file on github is exactly

the same as for MD5E of that file's contents when computing file hash locally. This was a fantastic solution that speeded up the procedure to get the result, make this project reasonably efficient in the matter of time and memory, also simplified realizing how exactly it works.

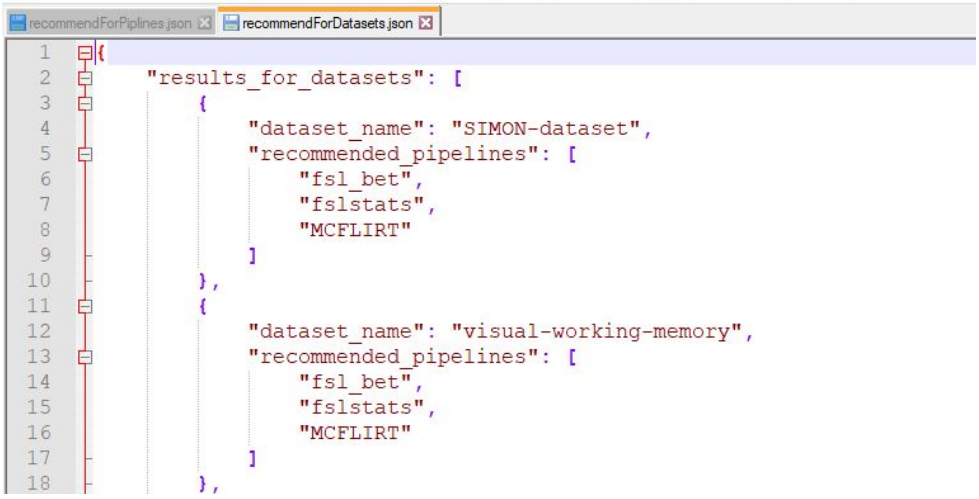
Furthermore, at first, to have these file hashes locally I should use datalad install recursively [2] to get all these sym-links on my computer, it was a very time-consuming process. Also, for updating I had many difficulties, I had to reinstall all these datalad datasets recursively, to make it faster I should have stored them and just check for new files, although I tried hard for that, it did not happen. Additionally, I got that github is a beautiful solution here, by cloning CONP github with submodules I could have all sym-links locally, and interestingly, by git pull submodules, updating was beautifully solved.

Results

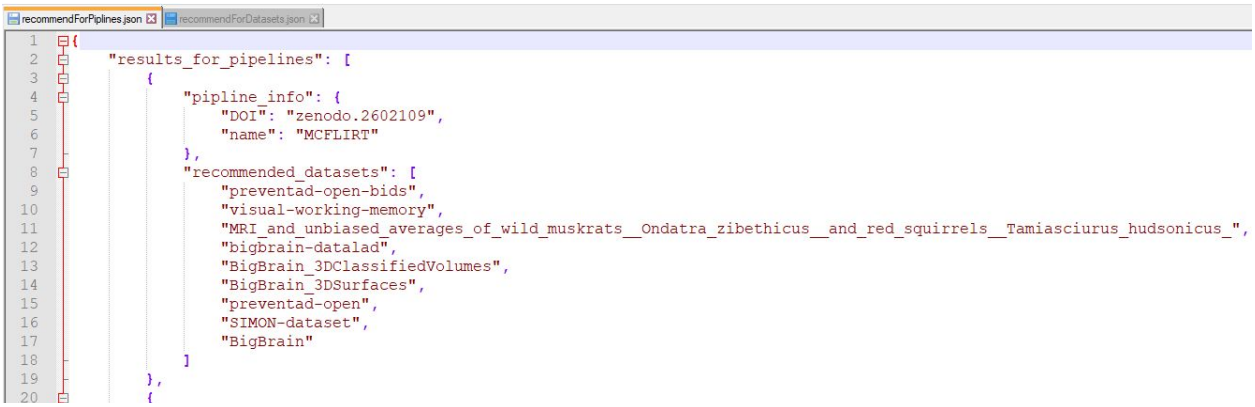
All these expectations are satisfied and are available when using the designed python API, CONP_Recommender. Everything on how to install and use this API is available on my git repository , I would be happy to see any comment there. To have a review on what has been done and the outcomes of the project, following screenshots are provided to illustrate how straightforward is to use this API. And finally, the result of the recommender system which is some JSON files is provided as well.



4- CONP_Recommender commands



5- Recommendations for datasets



6- Recommendations for pipelines

Conclusion

To summarize, to have such a recommendation ready to simply use, investigation, implementation and test were the key ingredients. The main process is to match datalad datasets with CONP pipelines through file hashes and employing this connection to implement the recommendation system. Obtaining datalad datasets' information and file hashes, how to design

a recommendation system and how to update the results automatically were the most challenging parts of this project.

Future works

Provenance-based recommendation system, and the idea behind that is robust enough to be continued, improved and more helpful to scientists who are working with such datasets and pipelines. From a personal perspective, improving the recommendation idea and its implementation would be the next interesting step. There are some aspects that would help the recommender system to be more accurate and reliable.

To have a glimpse of what could be done, the recommendation system is based on successful runs, however, successful runs only, does not necessarily mean to be the robust reason to recommend a dataset to a pipeline. More specifically, although BigBrain dataset should not be recommended to MCFLIRT or FSL-bet pipeline, it is one of the results of provenance-based recommender since currently this recommendation system gets successful runs as the key ingredients and successful run in this case just means that these pipelines did not crash on BigBrain. This part is on top of to-do list for future works to be improved.

To improve the reliability and accuracy of provenance-based recommender, adding more features as cardinalities, pipelines parameters and explanations, as some measures to specify the strength of path between a pipeline and a dataset, might be helpful. Cardinalities here for a path between a pipeline and a dataset, represents the number of times that pipeline has successfully run some files in that dataset, even it can be more detailed, ratio of successful runs to total runs ,as an example.

What I mean by pipeline's parameters is that each pipeline has some features and is designed for a specific group of datasets, also each dataset has some attributes that illustrate how and where it should be employed. Considering all these features we might be able to strengthen this recommender and its results. Also, explanations are the reasons why a pipeline is recommended to a dataset, is there a successful run for those two? is it a result of duplicate files in more than one datasets (5- Same files in datasets)? Or this is a result of common successful runs (4- Successful runs in common)? Since the main idea is a graph-based recommender, all these measures are potential to be added to each node or each path to show the strength of each path or recommendation.

References

1. git-annex https://git-annex.branchable.com/

2. datalad https://www.datalad.org/datasets.html