Create an interactive onboarding environment for new users of Hub

Proposal Google Summer of Code 2021
Organization: Python Software Foundation
Sub-organization: Activeloop
Mentor: Jakub Boros

OVERVIEW

When working in the field of Data Science, management of datasets and setting up a proper data flow pipeline can be extremely time-consuming, cumbersome, and prone to failure. The Activeloop organization provides data scientists worldwide with a solution to these problems so that they can spend more time training their models and saving resources. Hub is a package that stores petabyte-scale datasets as a single numpy-like array on the cloud, so one can seamlessly access and work with it from any machine. Hub has a simple API that enables its users to obtain a lot of functionality with just a few lines of code. However, as a complete beginner, it is difficult for many to learn the nuances of a new package. Reading the documentation is something that helps, but it is not exactly a fun way of learning something new. It can become tedious and boring. This is exactly why it would be great if Hub had a package that interactively teaches the basics of hub in the CLI itself.

GOALS

1. Creating a new package that uses Hub internally to provide interactive courses on the basics of Hub in the CLI.
2. Make the package support different types of inputs such as code snippets, multiple choice question answers, etc.
3. Make the format of courses scalable and simple so that anyone can write new courses in the future.
4. Write courses involving the basics of Hub, so that newcomers may join the community seamlessly and learn faster.

ABOUT ME

Contact Details

Name	Debaditya Pal
Email	debaditya.pal6@gmail.com
Github	DebadityaPal
Slack ID	@Debaditya Pal
Time Zone	Indian Standard Time (GMT +5:30)
University	Indian Institute of Information Technology, Gwalior
Major	Computer Science and Engineering
LinkedIn	Debaditya Pal

Personal Background

I am a second-year undergraduate student currently pursuing a Bachelor of Technology Degree in Computer Science and Engineering. I prefer working on Ubuntu 20.04 with a Python3 REPL environment, however, I test my work on both Linux and Windows before making a PR.
I was heavily interested in competitive programming since high school, something which was fueled by my love for problem solving and puzzles. It is because of that, I developed a strong grip over data structures and algorithms even before entering my university which worked out perfectly as it allowed me to further explore the different branches of computer science. I even qualified for the ICPC regionals as a school team, but couldn't go further than that since it is a college-level competition. During my first year, I started experimenting with Machine Learning and then gradually shifted to Deep Learning. I was particularly interested in Natural Language Processing and published my first research paper in that domain titled "Data Agnostic RoBERTa-based Natural Language to SQL Query Generation".
After my freshman year ended, I sought out to explore the depths of Open Source Development as I was told, it was a very rewarding experience, something that I agree with wholeheartedly. The feeling of elation when a PR gets merged and you know you have impacted the lives of thousands of users who use the software is something that keeps me going every day. During my third semester, Mikayel introduced me to Activeloop and the Hub and I was immediately interested since it combines two of my dearest passions, Machine Learning and Open Source Development, and has an amazing ever-helpful community.

Logistics

The GSoC timeline is in sync with my university's summer break and thus it will give me enough time to work on this project. I understand that GSoC is equivalent to a full-time program and hence I plan to devote at least 30-35 hours a week to this project. My university reopens in mid-July and even if some part of the timeline coincides there will be no exams, tests, or assignments in the first half of the semester, allowing me to devote ample time to work on and complete this project in the stipulated time frame. I am excited to spend my summer working on this project!!

Open Source Contributions with Activeloop

Some of my contributions to the Activeloop community are as follows:

hub/api/tensorview.py	#489	**get_label** parameter added to compute() of Tensorview	**Merged** (+149 −18)
benchmark_sequential_access.py	#508	Benchmark for **TileDB** and Hub added	**Merged** (+76 -0)
benchmark_sequential_access.py	#512	Added benchmarks for **Zarr** compared to Hub	**Merged** (+104 -0)
benchmark_sequential_write.py	#538	Added **sequential write** benchmarks	**Merged** (+133 -0)
hub/store/shape_detector.py	#616	Removed Assertions from shape_detector.py and **added exceptions**	**Merged** (+25 -16)
hub/api/integrations.py	#576	**Refactored** Dataset Class	**Merged** (+628 −483)
nlp_using_hub.ipynb	#542	hub and **huggingface transformers tutorial** added	**Merged** (+540 −0)
dnafrag_hub_backend.ipynb	#676	Added ipynb file with **benchmark** tests for **dnafrag package**	**Merged** (+222 -0)
genomelake_hub_backend.ipynb	#680	Added **genomelake hub backend benchmarks**	**Merged** (+214 -0)
webdataset_hub_benchmarks.ipynb	#733	Added **WebDataset Hub** benchmarks	**Merged** (+411 -0)

Open Source Contributions with other organizations

JuliaLang/stdlib/Test/src/test.jl	#38270	**automatic keyword assignment** support to test macro added	**Merged** (+11 -0)

The Proposal

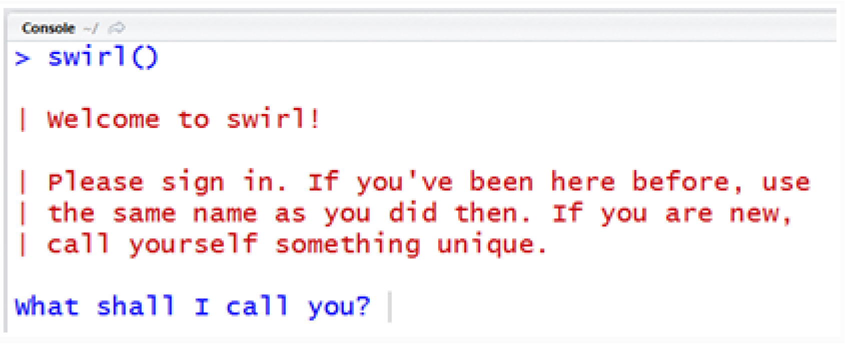
Current Situation

Hub has recently been getting a lot of traction on GitHub and has even trended in the top 10 multiple times. The participation of Activeloop as a sub-organization in Google Summer of Code has also attracted many developers to the package. Since we have a lot of users and developers who are new to the package and want to get to know how to use it, it is the perfect time to release an interactive course that informs them about the basics of Hub and how they can get to use it fast. As seen in the Slack workspace, many users are asking for video demonstrations on how Hub works as reading the documentation is a really slow process of learning. The documentation can often become overwhelming and confusing as it contains both user-level and internal function descriptions together in the API reference. Now, an interactive course system could be a more effective solution than video tutorials as this lets the users simultaneously code, learn new topics and answer questions to strengthen their grip on these topics whereas videos are mostly one-sided.

Proposed Features and API(s)

1. Creating a self-contained Python package using Hub internally

The ultimate goal of this project would be to create a separate independent package using the Python language that would be easy to install in a local development environment. For ease of usage, proper documentation of the package will be maintained. This package would be responsible for cataloging and managing the course directories and providing the available courses as and when requested by a user. It would be using Hub internally to check for the syntax and semantics of the user inputs. Ideally, I would like to recreate something like swirl for R, but in Python for Hub.

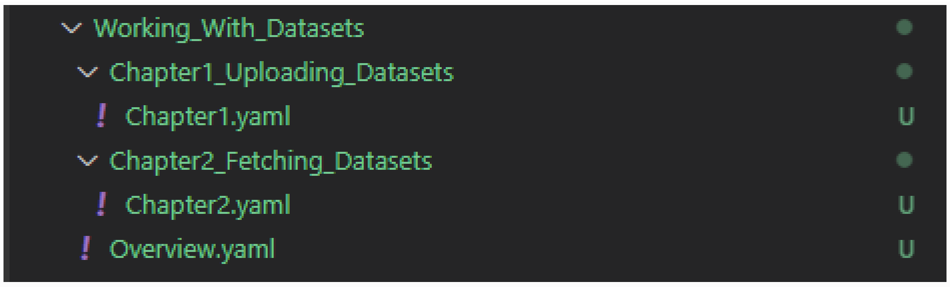


```
> swirl()

| Welcome to swirl!

| Please sign in. If you've been here before, use
| the same name as you did then. If you are new,
| call yourself something unique.

What shall I call you?
```

Figure 1: Similar UI will be reproduced in Python

2. Course Structure

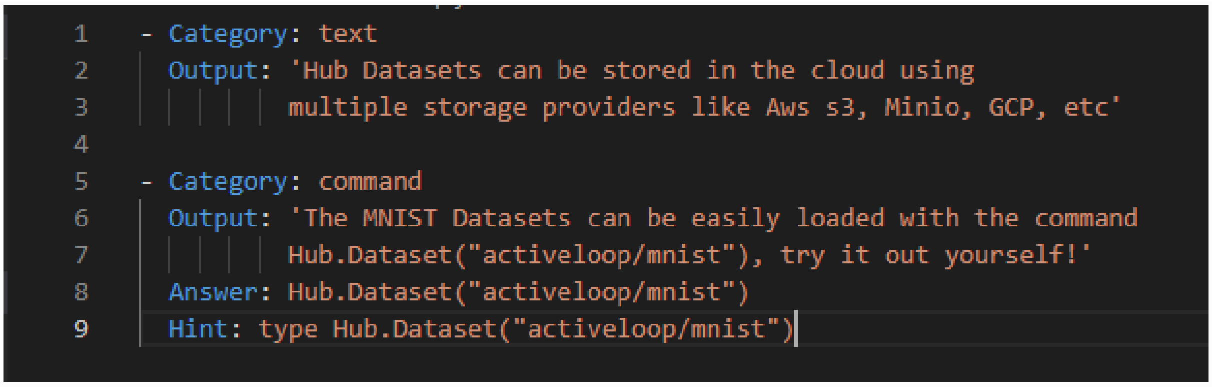
A course is expected to cover an entire broad topic of features and each course will have multiple chapters. Each chapter is going to deal with a specific feature. For example: Working with Datasets could be a course, where Uploading a Dataset, Fetching a Dataset, etc would be all the chapters.
The courses would be a tree of directories where each chapter is a subdirectory. This step is taken to ensure a structure for the course content so that the package can easily know where to look for a particular file should it need to.



```
Working_With_Datasets
  Chapter1_Uploading_Datasets
    ! Chapter1.yaml                U
  Chapter2_Fetching_Datasets
    ! Chapter2.yaml                U
  ! Overview.yaml                  U
```

Figure 2: Structure of Course Directory

Each of these individual files that contain the course content would be a text file or a YAML file so that the data is stored in a human-readable format so that new authors could refer to existing courses to get ideas.



```yaml
- Category: text
  Output: 'Hub Datasets can be stored in the cloud using
          multiple storage providers like Aws s3, Minio, GCP, etc'

- Category: command
  Output: 'The MNIST Datasets can be easily loaded with the command
          Hub.Dataset("activeloop/mnist"), try it out yourself!'
  Answer: Hub.Dataset("activeloop/mnist")
  Hint: type Hub.Dataset("activeloop/mnist")
```

Figure 3: Sample course chapter file

3. Extending Support to multiple types of user input

As Figure 3 illustrates, there is a category field for each course content entry. This field exists to inform the parser of the kind of input to expect from the user. For example:

1. The text category does not expect an input
2. The command category expects a code snippet as input

Right now I have 3 types of categories in mind, namely: Text, Command and Multiple Choice Q&A. The variation in the type of inputs serves the purpose of not making a course monotonous or boring to the learner. The support can be extended to newer types of user inputs in the future.

4. Robust API system for managing the streaming of a course

Instead of a hardcoded script for a particular course, I would implement a course engine. This would involve creating a system of APIs that work with courses as the variable. The idea is to keep the course content separate from the backend and the mechanical aspect of the package, this enables us to create a scalable package where the API is extendable so that new courses and course content may be added in the future when newer features are released. Aside from that, the types of user inputs will also be extendable along with the course API. The work of the backend is essentially that of a parser. Given the name of a course and the chapter within, the package would read the respective course file and follow the entries sequentially. The .yaml files will contain the course content, questions and code snippets whereas the package would be responsible to present them to the user and check the user inputs for a match with the answers provided in the course file. The provision of providing a hint upon request may also be implemented.

5. Create an initial library of courses covering all the basic topics

Once the final draft of the package has been made, and the package passes all development tests, I would begin working on courses themselves. There are many basic topics that I plan to implement courses for. These include but are not limited to creation, upload, and streaming of datasets as well as transformations and visualization. All the courses created will also be tested for errors and erroneous behaviour.

Q. Who will benefit from this?

All the new developers and users who have recently joined Hub and are interested in learning the workflow fast would be helped by this project. It would help the Activeloop organization in general as this idea lets you to quickly provide a seamless onboarding experience for new learners. Interactive courses are known to capture the attention of their audience more than written text material like the Documentation, so this will also help minimize the user bounce rate. Additionally it can provide a quick learning method for newer features in the future as authors can write courses for them as they are released so that even the veteran Hub users can keep themselves up to date with the new additions to the package. Lastly, this project could be helpful to those who want to help others learn about hub but are looking for an efficient way to do so, as they can contribute by writing more courses that people will undertake.

Q. Why choose me for this project?

Having tried multiple domains, the most fulfilling for me was open-source development. I have been an active user and contributor to hub for the past couple of months which has made me extremely well accustomed to the codebase. Not too long ago, I was a beginner myself and having made the journey of learning the nuances of this package I know about the problems one might face during theirs. This makes me well equipped to address those issues. I have a clear idea of what I want to implement but I am also open to feedback from the community on how this project should be shaped. My background in development allows me to get an idea of how to go about implementing the course engine and my commitment towards the project can be seen through the number of questions I ask the mentors and my readiness to contribute to new tasks.
Being able to be a part of this organization through the Google Summer of Code will be an experience unparalleled in its learning outcome and with my enthusiasm, I believe I would be a good candidate for the project.

Timeline

This is a tentative list of objectives I plan to achieve in the mentioned time frame. There could be situations in which the project may get delayed or lead to an early completion. I plan to handle such cases by having a buffer week in the timeline and stretch goals respectively.

Pre GSoC Period

In this period I aim to study swirl and understand the way their backend works. Since I know a bit about R, this should not be very difficult. Once that is over, I could begin searching for existing packages that might help during the development of this one, although it would be a prime goal of having as few dependencies as possible to make the package light quick to install.

Community Bonding Period

This period will primarily be dedicated to communication with the mentors. I will discuss potential changes to the API and the proposed course structure. Approach to sub parts will be discussed and corrected if required. Additionally, I would reach out to the beginners who are on the Hub Slack workspace to get their inputs and feedback, since this project is mainly aimed towards them.

Week 1 - 2 [June 7 - June 20]

- Implementation of a YAML parser that will read the course content from a chapter file
- Creating a basic course engine
- Create a class for managing chapters

Week 3 [June 21 - June 27]

- Documentation
- Testing

Week 4 [June 28 - July 4]

- Extending support to Text type of course content
- Improving Error Handling

Week 5 [July 5 - July 11]

- Pre - Release in the community to gather feedback.
- Documentation
- Testing

------------------ PHASE 1 EVALUATIONS ------------------

Week 6 [July 12 - July 18]

- Extending support to Command type Course Content
- Documentation

Week 7 [July 19 - July 25]

- Extending support to MultipleChoiceQ&A type Course Content
- Create a class to handle questions and get answers from users.
- Documentation

Week 8 [July 26 - August 1]

- Finalizing the Course Engine
- Writing Courses

Week 9 [August 2 - August 8]

- Writing Courses
- Writing Tests involving all 3 types of course content

Week 10 [August 9 - August 15]

- Buffer Week
- Documentation
- Testing

------------------ FINAL EVALUATIONS ------------------

Stretch Goals

- Addition of colors to make the courses more attractive
- Optimizing the performance of the course engine to reduce overhead.

Post GSoC

Continue to work on stretch goals. Newer features will keep on getting added to Hub and I could help with writing courses on them so that people can get themselves up-to-date fast. Try to implement new methods of user input to make courses more interactive.