***I.J. Education and Management Engineering*, 2019, 6, 35-39**
Published Online September 2019 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijeme.2019.06.04

Available online at **http://www.mecs-press.net/ijem**

Running Google Colaboratory as a server – transferring dynamic data in and out of colabs

Samarth V. Halyal

Department of Computer Science and Engineering, KLE Dr. M. S. Sheshgiri College of Engineering and Technology, Udyambag, Belagavi, 590008, India

Received: 24 July 2019; Accepted: 15 October 2019; Published: 08 November 2019

Abstract

Mainly all the hype in the field of Machine Learning and Neural Network has led the students of this century learn and research in this field, and the primary issue faced by lot of students and researchers is the availability of proper infrastructure. The complex algorithms need good quality and quantity of CPU's and GPU's. Fortunately Google has an aid for such demand, They named it Google Colaboratory. which they provide free of cost. But has lesser flexibility for the user to pass data required for computational purposes. This paper talk about a way in which researchers may feed/retrieve data dynamically without using the manual upload feature of google colaboratory. This will help the researcher or student to concentrate more on their thesis rather than pay attention on data.

Index Terms: google colaboratory, shell script, ipython notebook, google drive.

© 2019 Published by MECS Publisher. Selection and/or peer review under responsibility of the Research Association of Mode rn Education and Computer Science

1. Introduction

The Field of Neural Networks and Artificial Intelligence is a fast growing area which has been giving enthusiasm to lot of students who have developed a aim to pursue their careers in this field. Not only the students but also the experienced fellows in the field who look forward for further research in these fields have been in constant look up for better self-performance.

* Corresponding author.
E-mail address:

But the main problem that is hindering most of the people is proper access to tools and hardware infrastructure to aid the study of the algorithms involved in these fields, but google has been helpful in providing this need for them through their project named as google colaboratory, an iPython notebook webapp. This webapp comes as an extension for the google chrome browser that gives users access to high end central processing unit (CPU), Tensor processing unit (TPU) and an graphics processing unit (GPU) with good amount of random access memory (RAM) and Video RAM (VRAM). This the users can select accordingly. But a slight problem now was that the user (Student/research person) has very less flexibility of training over the data he has, as the system refreshes every 12 hours or something and the actual maximum space on the system is also less. And also that the user is allowed to input the data into the working environment only through the manual button that exist on the webapp.

This paper discusses about a method (usually a series of code) that allows the user have complete control about the amount of data he/she needs to transfer and where and how to transfer. Google drive was a possible buffer that was used in this method to work around as a channel.

The paper released at IEEE spoke mainly on how colabs was used as performance analyzer as a tool for accelerating deep learning applications. It also gave a in-depth information on usage and technical aspects of google colaboratory (a.k.a. colabs)[1].

2. Methodology

To show the working and actual methodology let us consider a simple web application that generates a story given some seed text, this is working using recurrent neural network on long short term memory units that was trained on google colabs, its simple interface is as shown in Figure 1.

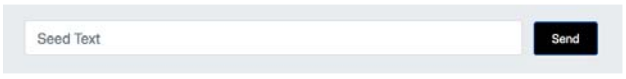
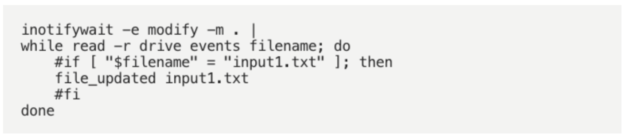


Fig. 1. The simple interface

Now after every line of story generated the current story in updated and given to the algorithm as seed text. This web application is hosted on Heroku and serves only as an user interface, the real computation occurs on colabs. This specific application is a working model on the more general topic of ours that is the way of sending unconditional data to and from colabs, so a very small amount of data travels in this exhibition. This similar idea can be extended to any amount of data with proper careful changes.

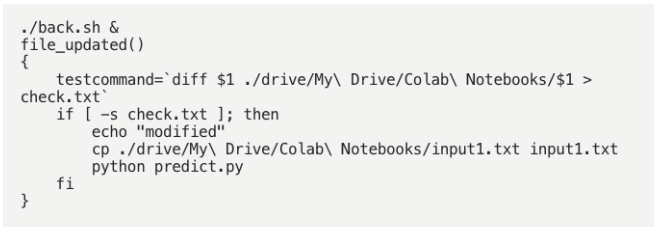
So the basic technique that we can use here is keeping the colabs notebook always ready to accept input, this can be done using the “inotifywait” command on linux[2]. As colabs can run linux commands just by using an exclamation mark(!) this becomes easy to install and use “inotifywait” command. Figure – 2 shows the shell script implemented to do so.



```
inotifywait -e modify -m . |
while read -r drive events filename; do
    #if [ "$filename" = "input1.txt" ]; then
    file_updated input1.txt
    #fi
done
```

Fig. 2. the shell script implemented

What this script does is that is watches over the current directory and triggers the code written after the pipeline whenever there is a modification to current directory. Different options and arguments to be passed for this command can be seen at its documentation page referred in the reference section. This shell script after the pipeline, states that read all the files in the current directory till we find the file named input1.txt, then check whether it is updated. This logic to check whether it is updated in written in another subroutine as per Figure 3.



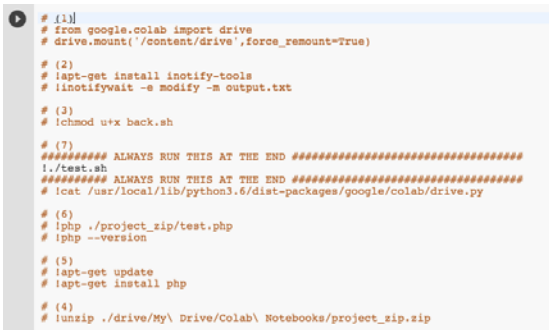
```
./back.sh &
file_updated()
{
    testcommand=`diff $1 ./drive/My\ Drive/Colab\ Notebooks/$1 >
check.txt`
    if [ -s check.txt ]; then
        echo "modified"
        cp ./drive/My\ Drive/Colab\ Notebooks/input1.txt input1.txt
        python predict.py
    fi
}
```

Fig. 3. This logic to check whether it is updated in written in another subroutine

This subroutine as can be seen has a background command which has a shell script as its first line. This shell script is nothing but the code we saw earlier in Figure 2. In this we are running a command “diff” on two files “$1” at two different folders. What this “diff” does is it gives output as null if both files are similar and some information as in word of mismatch if both files are dis-similar. The one in current directory is the one we need to check whether is updated?. This argument $1 is taken from previous code where the input file was passed with subroutine call. We store this difference in a file named “check.txt” and on next line check whether its Not empty. If yes then the file was updated so we print “modified” and copy the “modified” file. And the further steps are user depended, And is not related much to current discussion. Till now we have seen how to keep running colabs as a server ready to accept input. Figure 4 shows how these files are kept ready on colabs to run at every 12 hours when data in refreshed and everything is rewritten.

Every step is numbered (1) to (7), these work with the project in discussion. But the one that’s most important is the one that is not commented out. “Test.sh” is the name of the code shown in Figure 3. As we could see the drive word being used, this is the google drive that needs to be authenticated before using it as shown in the figures. This can be done using the code shown in Figure 5. This is the colabs specific code that mounts the google drive into our working directory, specifically in this example we are refreshing this every 4 seconds(totally user standard).

Till now explanation included pulling data inside colabs, the way data was being updated at the so called buffer, our google drive is completely in the hands of the user. Figure 6 shows one simple way to doing this. This method is used because the application in discussion was working with php codes[3-4]. This needs authentication using tokens and credentials all of which can be found at the links given in the reference section.

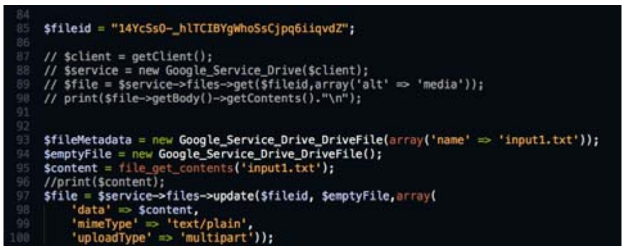


```
# (1)
# from google.colab import drive
# drive.mount('/content/drive',force_remount=True)

# (2)
# !apt-get install inotify-tools
# !inotifywait -e modify -m output.txt

# (3)
# !chmod u+x back.sh

# (7)
########## ALWAYS RUN THIS AT THE END ###################################
!./test.sh
########## ALWAYS RUN THIS AT THE END ###################################
# !cat /usr/local/lib/python3.6/dist-packages/google/colab/drive.py

# (6)
# !php ./project_zip/test.php
# !php --version

# (5)
# !apt-get update
# !apt-get install php

# (4)
# !unzip ./drive/My\ Drive/Colab\ Notebooks/project_zip.zip
```

Fig. 4. how these files are kept ready on colabs to run at every 12 hours

```
from google.colab import drive
drive.mount('/content/drive')
```

Fig. 5. the drive word being used



```
$fileid = "14YcSsO-_hlTCIBYgWhoSsCjpq6iiqvdZ";

// $client = getClient();
// $service = new Google_Service_Drive($client);
// $file = $service->files->get($fileid,array('alt' => 'media'));
// print($file->getBody()->getContents()."\n");


$fileMetadata = new Google_Service_Drive_DriveFile(array('name' => 'input1.txt'));
$emptyFile = new Google_Service_Drive_DriveFile();
$content = file_get_contents('input1.txt');
//print($content);
$file = $service->files->update($fileid, $emptyFile,array(
    'data' => $content,
    'mimeType' => 'text/plain',
    'uploadType' => 'multipart'));
```

Fig. 6. The one simple way

3. Flowchart

Figure 7 shows the flow of control from beginning of the application to the end, as can be seen our sample application when started and data is passed to google drive, drive stores it as buffer and if this new data is not the already existing data and is modified, it is passed to google colabs for processing else we wait looking in drive. We poll the google drive using the inotify command till there is some updation at google drive, this triggers copy of the new updated data to colabs and the user may work on it as intended. So the flowchart explains this process respectively i.e., from start of the application the google drive is polled for data update and further process is shown accordingly in flowchart.

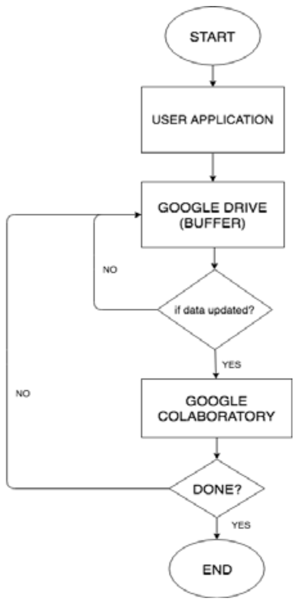


Fig. 7 the flow of control

4. Result

1. data is transferred from storage location where user wants, to google colabs without loss of any data.
2. Unlimited and unconditional flow of data, generally needed for training purpose.
3. Stored results are also transferred back to local disk and any location user wants by small possible modification in the code.University backed Business Incubator

5. Conclusion and Recommendation

This paper showed how using this small code snippet can one convert google colabs into a temporary server waiting for input. How this can be used to transfer huge amounts of data every time colabs refresh every 12 hours using the above mentioned technique. This provided shell script also showed how data can be taken from anywhere independent of the manual way google colabs grants us(through use of button).

References

[1] Tiago Carneiro , Raul Victor Medeiros Da Nóbrega, Thiago Nepomuceno , Gui-Bin Bian , Victor Hugo C. De Albuquerque , And Pedro Pedrosa Rebouças Filho : Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications.
[2] inotifywait documentation https://linux.die.net/man/1/inotifywait
[3] Introduction to google drive API - https://developers.google.com/drive/api/v3/about-sdk
[4] Uploading files to google drive https://developers.google.com/drive/api/v3/reference/files/update
[5] Blog post for the same topic - https://medium.com/@samarthhalyal/uncloaking-google-collaboratory -8c52e0ecd55b
[6] HAROON SHAKIRAT OLUWATISON : Client-Server Model
[7] HIROSHI NISHIDA, THINH NGUYEN : Optimal Client-Server Assignment for Internet Distributed Systems
[8] LEEJA JOSEPH : Real Time Communication with Client/Server Architecture Using Secure Shell Protocol

Author's Profile

Samarth Venkatesh Halyal completed his schooling and engineering in district of belgaum in india. This paper is written as a support paper for the final year project during his engineering. He is currently working in an embedded company at bangloare Machine Learning, Artificial intelligence, Deep learning, Operating systems, Linux are some of his fields of interest.

How to cite this paper: Samarth V. Halyal," Running Google Colaboratory as a server – transferring dynamic data in and out of colabs ", International Journal of Education and Management Engineering(IJEME), Vol.9, No.6, pp.35-39, 2019.DOI: 10.5815/ijeme.2019.06.04