Stack Overflow Tag Prediction

Musquan Karovalia
Department of Computer Science
University of Texas at San Antonio
San Antonio, USA
musquanhazaratali.karovalia@my.utsa.edu

Rajendhra Varma Sagi
Department of Computer Science
University of Texas at San Antonio
San Antonio, USA
rajendhravarma.sagi@my.utsa.edu

Ian Burkett
Department of Information Systems and Cyber Security
University of Texas at San Antonio
San Antonio, USA
ian.burkett@my.utsa.edu

***Abstract*—Our project aims to provide a comprehensive analysis of the process of predicting tags for questions posted on the popular online community, Stack Overflow. The project is conducted using Google Cloud, which serves as the primary platform for developing and implementing a machine learning model capable of accurately predicting the most appropriate tag for a given question.**

Keywords—cloud computing, Google Cloud, Spark, Machine Learning

I. INTRODUCTION

Stack Overflow, the online developer community, has experienced remarkable growth since its launch in 2008, drawing over 50 million visitors each month. The website provides an invaluable platform for programmers to learn, share knowledge and develop their careers. However, the vast number of questions and answers on the site can sometimes create difficulties in searching for relevant information, leading to frustration and loss of time for users. To address this problem, we have conducted a study aimed at developing a machine learning model that can predict appropriate tags for questions posted on Stack Overflow. Based on the type of tags assigned to questions, the top nine tags which were appearing more than 1000 times are: Python, Java, JavaScript, C#, C, C++, PHP, Android, IOS and we used these tags for training our model.

We utilized jupyter notebook on cloud for coding and testing purposes. Our study also involved creating spark clusters on the cloud to execute our code and train our logistic regression model. The ultimate objective of this study is to gain a deeper understanding of the fundamental workings of Google Cloud while simultaneously exploring the efficacy of machine learning in the context of predicting relevant tags for Stack Overflow questions.

We will look into the dataset we used, cleaning and preprocessing of data, the platform used, and the algorithm used to train the model further in the paper.

II. HISTORY OF GOOGLE CLOUD

Google Cloud was introduced in 2008 with the launch of Google App Engine, which allowed developers to create and host web applications on Google's infrastructure. App Engine supported popular programming languages like Python, Java, and Go. Later in 2010, Google introduced Google Cloud Storage, a scalable and durable storage option for data backups, file serving, and Big Data analysis.

In 2011, Google acquired Postini, an email and web security company, and integrated it into Google Apps, the company's suite of business applications. This led to the creation of Google Vault, a data archiving and eDiscovery service that helps businesses comply with legal requirements for data retention and retrieval.

In 2013, Google launched Google Compute Engine, a virtual machine hosting service, and Google Cloud SQL, a cloud-based database service. The company also acquired Stackdriver, a cloud monitoring service, and integrated it into Google Cloud Platform, providing customers with a comprehensive monitoring and logging solution. In subsequent years, Google Cloud continued to expand its portfolio of services, including BigQuery, a cloud-based data warehousing and analytics service, and Cloud AI Platform, a suite of machine learning tools and APIs. Today, Google Cloud competes with other cloud providers like Amazon Web Services and Microsoft Azure, offering a wide range of services to businesses of all sizes.

III. GOOGLE CLOUD

Google Cloud Platform offers many services to developers. These cloud services are separated into nine different categories: Compute, Storage and Database, Networking, Big Data, Internet of Things, Machine Learning, Identity and Security, Management Tools, Developer Tools.

Google Cloud Platform provides a wide range of services and tools that are categorized into nine categories: compute, storage and database, networking, big data, Internet of Things, machine learning, identity and security, management tools, and developer tools.

These categories offer various services such as App Engine, Compute Engine, Container Registry, SQL databases, cloud storage, Virtual Private Clouds (VPCs), Load Balancers, Spark and Hadoop capabilities, Cloud DataLab, Google Data Studio, BigQuery, IoT core for secure device connection, Google's Machine Learning Engine and API's, Cloud Identity and Access Management (IAM), Cloud Data Loss Prevention API, Cloud Resource Management, monitoring and logging, cloud endpoints, cloud console, and the Cloud SDK. All of these categories and services are designed to provide end users with powerful and useful capabilities for their cloud computing needs.

IV. DATA OVERVIEW

The file we used was Train.csv which had 4 columns:
Id – Unique identifier for each question [1]

Title–The question's Title [1]

Body-The body of the question [1]

Tags - The tags associated with the question in a space separated format. [1]

The dataset under consideration in this study comprises a variety of questions that are randomly selected from multiple sources, including verbose text sites, math-related sites, and programming-related sites. The number of questions taken from each site is not fixed and may vary. Most of the questions were related to programming concepts. The dataset has not undergone any sort of filtering, such as excluding closed questions or any other criteria. After analyzing the whole problem, we realized that we don't require Id and Body column. That's why we did not use those columns while training the model.

V. DATA CLEANING

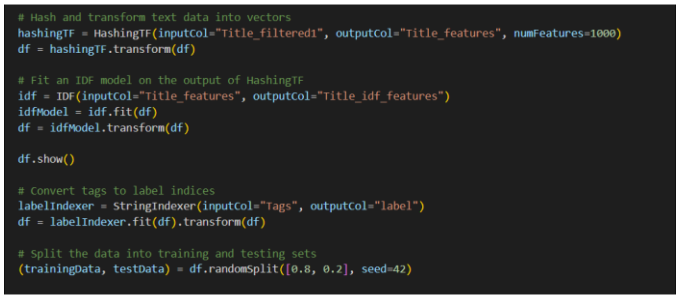
As part of our study on predicting tags for questions, we performed several data preprocessing steps. Firstly, we removed the "Id" and "Body" columns from the dataset as they were not relevant to our study. Next, we focused on the "Title" column, which contained textual information that we used to predict the appropriate tags. To enhance the efficiency of this column, we removed all non-alphanumeric characters such as punctuation marks, symbols, and white spaces. This was done to simplify the data and reduce noise, thereby enabling our machine learning model to better identify patterns and relationships in the data. Overall, our data preprocessing steps were designed to improve the quality and relevance of the data for our study, ultimately leading to better prediction accuracy for tags on Stack Overflow.

VI. DATA PREPROCESSING

Data preprocessing is an essential step in machine learning that involves transforming raw data into a more manageable and meaningful format. Machine learning algorithms often require specific types of data, such as numeric data, normalized data, or data with a specific distribution.

For our project, we converted the text data in the Title column into numerical data using a technique known as TF-IDF. TF-IDF stands for Term Frequency-Inverse Document Frequency. It is a statistical measure used to evaluate the importance of a word in a document, based on how frequently it appears in the document and how rare it is in the corpus of all documents.

After this, we converted the Tags column into labels. For each tag in the list, we created a binary label vector. Then, we splitted the dataset into a training set and a testing set. The training set is used to train the machine learning model, while the testing set is used to evaluate the performance of the model. Figure 1 shows the coding part of all the preprocessing steps mentioned above.



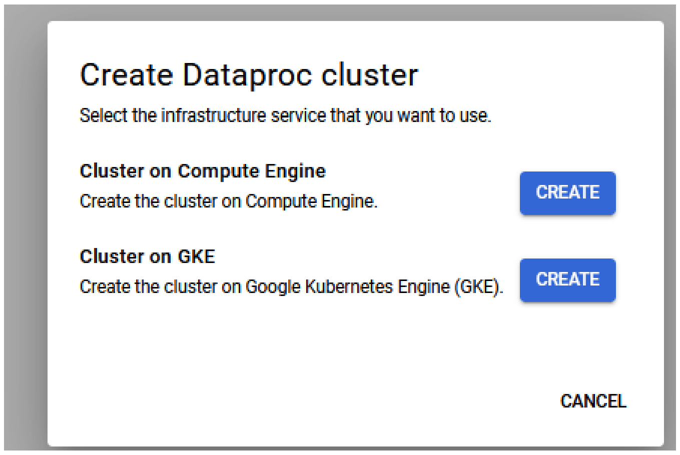
```
# Hash and transform text data into vectors
hashingTF = HashingTF(inputCol="Title_filtered1", outputCol="Title_features", numFeatures=1000)
df = hashingTF.transform(df)

# Fit an IDF model on the output of HashingTF
idf = IDF(inputCol="Title_features", outputCol="Title_idf_features")
idfModel = idf.fit(df)
df = idfModel.transform(df)

df.show()

# Convert tags to label indices
labelIndexer = StringIndexer(inputCol="Tags", outputCol="label")
df = labelIndexer.fit(df).transform(df)

# Split the data into training and testing sets
(trainingData, testData) = df.randomSplit([0.8, 0.2], seed=42)
```

Figure 1

VII. GOOGLE CLOUD SETUP

For this project, we utilized a virtual machine on Google Cloud. To set up this virtual machine, we first took advantage of the free 300 credits that Google Cloud provides. In order to operate the cloud service beyond this free period, we had to purchase an API that came with a monthly cost. The machines themselves were also subject to payment while they were in use, at a cost of approximately one dollar per day for a basic level of machine use. It's important to note that we were charged even for the resources we weren't using, so if we had turned off the clusters when they were not in use, we could have saved some credits or money.

For the project set up itself, we first had to create a project which was labeled my first project. In **Figure 2,** we have the option of GKE cluster or data proc. From our research, the data proc is unmanaged which means maintaining the instance is on us and there is more of a setup for the cluster. Then in **Figure 3,** we are creating a data proc cluster labeled team17. We used the default settings because it has the least cost and because we did not need a heavy machine to run Spark. In **Figure 4,** you can see that there are multiple CPU options. Changing these options would change the price of the monthly cluster. We chose to lower the CPU usage to 8 GB of memory and lowered the hard drive to 500. There is 1 master node and 2 workers. After that, we created the cluster.



Create Dataproc cluster

Select the infrastructure service that you want to use.

Cluster on Compute Engine

Create the cluster on Compute Engine.

Cluster on GKE

Create the cluster on Google Kubernetes Engine (GKE).

Figure 2

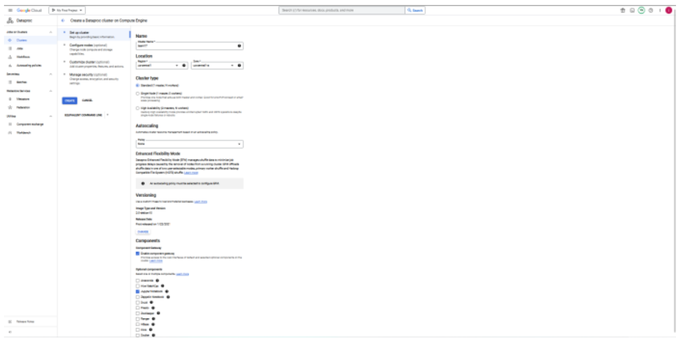


Figure 3

Figure 4

In **Figure 5**, the network usage, memory allocation, and disk space monitoring options were very useful to us. They allowed us to keep track of the resources being used by our cluster, which was particularly important during instances when the machines were not running the project properly and showed 0 memory allocation. At such times, we could stop and start the cluster to fix the issue. Another crucial feature in this figure is the job submission tab and the job tab itself, which enabled us to run our project without having to SSH into the cluster like in previous labs. Additionally, the web interfaces shown in **Figure 6** allowed us to attach Jupyter lab to our machine without having to SSH into it. This, in turn, helped us run our code seamlessly without the need to execute it manually on the cluster each time.

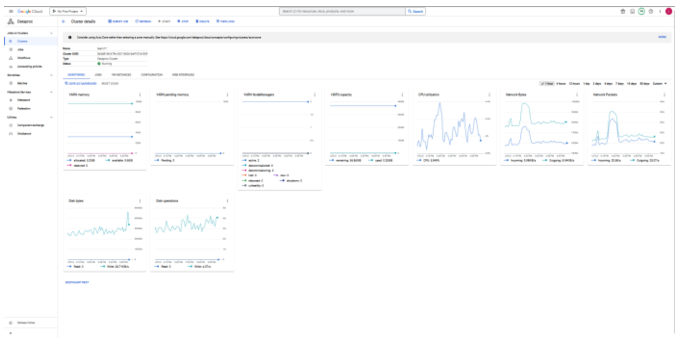
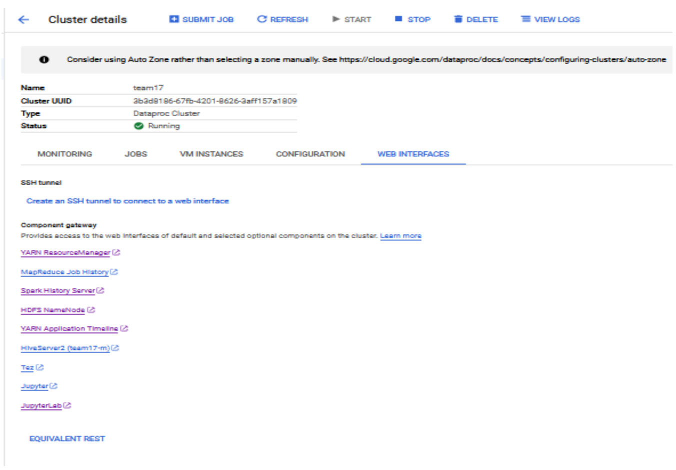


Figure 5



Consider using Auto Zone rather than selecting a zone manually. See https://cloud.google.com/dataproc/docs/concepts/configuring-clusters/auto-zone

Name	team17
Cluster UUID	3b3d9186-67fb-4201-8626-3aff157a1809
Type	Dataproc Cluster
Status	Running

SSH tunnel

Create an SSH tunnel to connect to a web interface

Component gateway

Provides access to the web interfaces of default and selected optional components on the cluster. Learn more

YARN ResourceManager
MapReduce Job History
Spark History Server
HDFS NameNode
YARN Application Timeline
HiveServer2 (team17-m)
Tez
Jupyter
JupyterLab

Figure 6

In **Figure 7**, we can go to the storage to get to our buckets. Buckets are the location where we are uploading and saving our data. This is important because this is how we call the CSV file for our program. In **Figure 8,** we have clicked on a file and can see when it was created its file size, and some URLs. If someone has our HTTP URL, they can download the file that is in our bucket. The second URL gs:// is used like Hadoop HDFS file storage. Instead of manually uploading to Hadoop we can use this and it's a alternative for running our programs or calling CSV for use in our program.

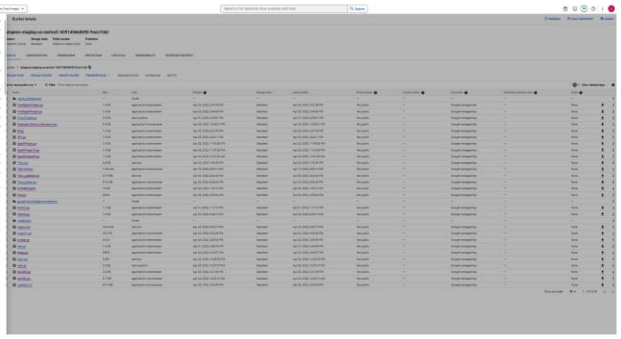
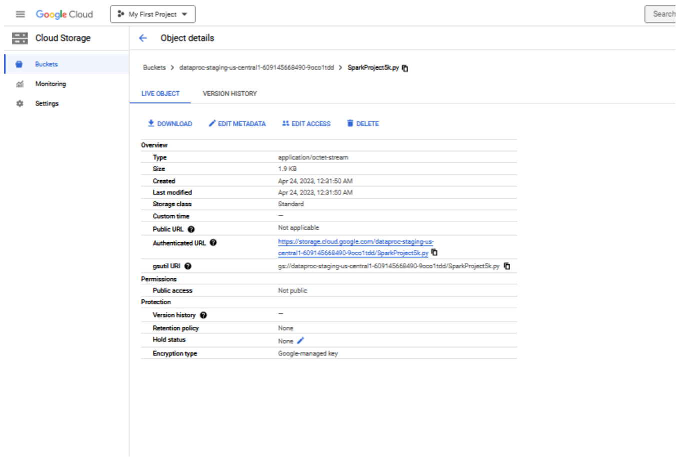


Figure 7



Object details

Buckets > dataproc-staging-us-central1-609145668490-9oco1tdd > SparkProject5k.py

Overview	
Type	application/octet-stream
Size	1.9 KB
Created	Apr 24, 2023, 12:31:50 AM
Last modified	Apr 24, 2023, 12:31:50 AM
Storage class	Standard
Custom time	—
Public URL	Not applicable
Authenticated URL	https://storage.cloud.google.com/dataproc-staging-us-central1-609145668490-9oco1tdd/SparkProject5k.py
gsutil URI	gs://dataproc-staging-us-central1-609145668490-9oco1tdd/SparkProject5k.py
Permissions	
Public access	Not public
Protection	
Version history	—
Retention policy	None
Hold status	None
Encryption type	Google-managed key

Figure 8

The last section before running the actual file is the section before running the program. We have the IAM, seen in **Figure 9.** In this section, we can set up users to be able to make changes on the cloud. All three of us are set up as owners in cases big changes need to be made when the others are not available. In an organization, they could assign users different roles to run Spark programs but could not make changes to the cluster itself.

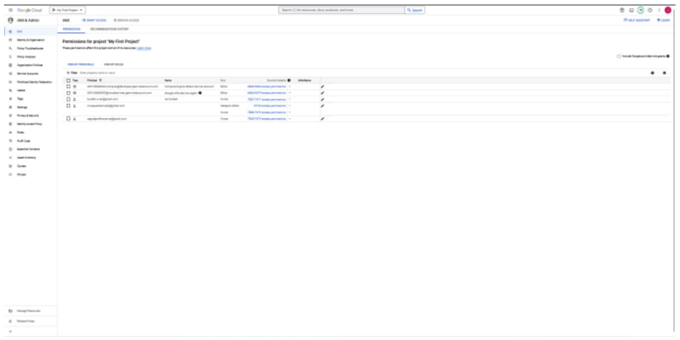
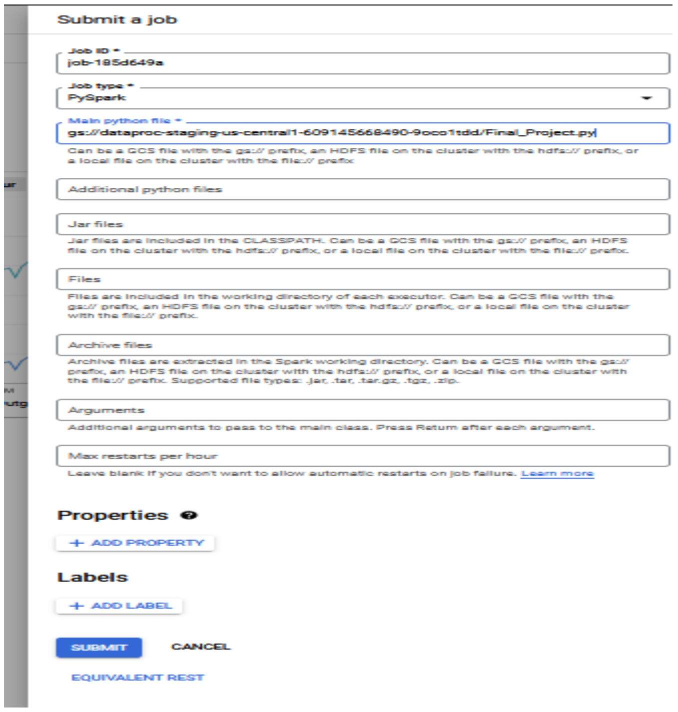


Figure 9

In **Figure 10,** we are submitting a job to run. Here we have set the job id by default. Chosen the job type as a spark and added the gs:// file. After we hit submit the job will start up, then either completed or fail. In **Figure 11** the file is running, and you can see there is a section for outputs. This will tell us what's going on with the job and if there are any issues. For us, we are looking for an accuracy score to be printed out seen in **Figure 13.**



Submit a job

Job ID *
job-185d649a

Job type *
PySpark

Main python file *
gs://dataproc-staging-us-central1-609145668490-9oco1tdd/Final_Project.py

Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix

Additional python files

Jar files

Jar files are included in the CLASSPATH. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Files

Files are included in the working directory of each executor. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix.

Archive files

Archive files are extracted in the Spark working directory. Can be a GCS file with the gs:// prefix, an HDFS file on the cluster with the hdfs:// prefix, or a local file on the cluster with the file:// prefix. Supported file types: .jar, .tar, .tar.gz, .tgz, .zip.

Arguments

Additional arguments to pass to the main class. Press Return after each argument.

Max restarts per hour

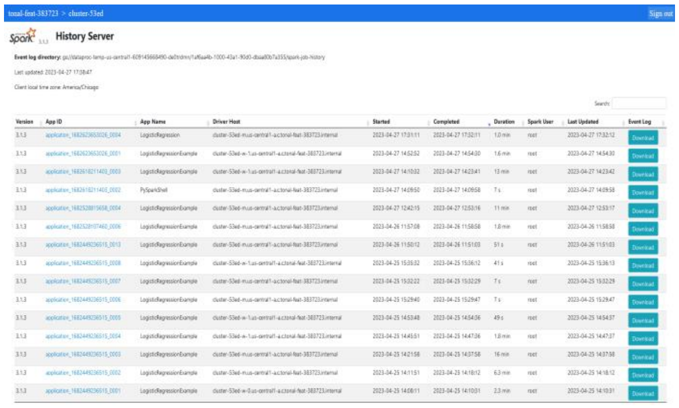
Leave blank if you don't want to allow automatic restarts on job failure. Learn more

Properties

Labels

Figure 10

Figure 11



History Server

Event log directory: gs://dataproc-temp-us-central1-609145668490-de0trdmn/1af6aa4b-1000-43a1-90d0-dbaa80b7a355/spark-job-history

Last updated: 2023-04-27 17:38:47

Client local time zone: America/Chicago

Version	App ID	App Name	Driver Host	Started	Completed	Duration	Spark User	Last Updated	Event Log
3.1.3	application_1682623653026_0004	LogisticRegression	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-27 17:31:11	2023-04-27 17:32:11	1.0 min	root	2023-04-27 17:32:12	Download
3.1.3	application_1682623653026_0001	LogisticRegressionExample	cluster-53ed-w-1.us-central1-a.c.tonal-feat-383723.internal	2023-04-27 14:52:52	2023-04-27 14:54:30	1.6 min	root	2023-04-27 14:54:30	Download
3.1.3	application_1682618211403_0003	LogisticRegressionExample	cluster-53ed-w-1.us-central1-a.c.tonal-feat-383723.internal	2023-04-27 14:10:32	2023-04-27 14:23:41	13 min	root	2023-04-27 14:23:42	Download
3.1.3	application_1682618211403_0002	PySparkShell	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-27 14:09:50	2023-04-27 14:09:58	7 s	root	2023-04-27 14:09:58	Download
3.1.3	application_1682528815658_0004	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-27 12:42:15	2023-04-27 12:53:16	11 min	root	2023-04-27 12:53:17	Download
3.1.3	application_1682528107460_0006	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-26 11:57:08	2023-04-26 11:58:58	1.8 min	root	2023-04-26 11:58:58	Download
3.1.3	application_1682449236515_0013	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-26 11:50:12	2023-04-26 11:51:03	51 s	root	2023-04-26 11:51:03	Download
3.1.3	application_1682449236515_0008	LogisticRegressionExample	cluster-53ed-w-1.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 15:35:32	2023-04-25 15:36:12	41 s	root	2023-04-25 15:36:13	Download
3.1.3	application_1682449236515_0007	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 15:32:22	2023-04-25 15:32:29	7 s	root	2023-04-25 15:32:29	Download
3.1.3	application_1682449236515_0006	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 15:29:40	2023-04-25 15:29:47	7 s	root	2023-04-25 15:29:47	Download
3.1.3	application_1682449236515_0005	LogisticRegressionExample	cluster-53ed-w-1.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 14:53:48	2023-04-25 14:54:06	49 s	root	2023-04-25 14:54:37	Download
3.1.3	application_1682449236515_0004	LogisticRegressionExample	cluster-53ed-w-1.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 14:45:51	2023-04-25 14:47:36	1.8 min	root	2023-04-25 14:47:37	Download
3.1.3	application_1682449236515_0003	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 14:21:58	2023-04-25 14:37:58	16 min	root	2023-04-25 14:37:58	Download
3.1.3	application_1682449236515_0002	LogisticRegressionExample	cluster-53ed-m.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 14:11:51	2023-04-25 14:18:12	6.3 min	root	2023-04-25 14:18:12	Download
3.1.3	application_1682449236515_0001	LogisticRegressionExample	cluster-53ed-w-0.us-central1-a.c.tonal-feat-383723.internal	2023-04-25 14:08:11	2023-04-25 14:10:31	2.3 min	root	2023-04-25 14:10:31	Download

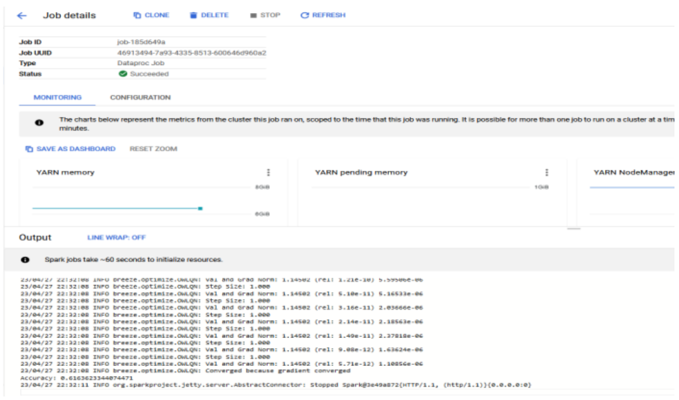
Figure 12

VIII. Challenges Faced

During the coding and testing of our project, we encountered certain resource-related issues like the inadequacy of resources in the yarn scheduler. These issues forced us to halt and restart the cluster every time we encountered such problems. This process helped us release all the previously utilized jobs and resources, which subsequently became available for future usage.

Additionally, while we were in the process of training our model, we started by converting the data in the Title column with the help of word embedding (vectorizer) technique. However, the outcome was not satisfactory in terms of efficiency. After that, we decided to switch to a different method called TF-IDF, and the results were quite remarkable as it resulted in significant enhancement in the accuracy of the model.

IX. Results



Job ID job-185d649a
Job UUID 46913494-7a93-4335-8513-600646d960a2
Type Dataproc Job
Status Succeeded

The charts below represent the metrics from the cluster this job ran on, scoped to the time that this job was running. It is possible for more than one job to run on a cluster at a tim
minutes.

Spark jobs take ~60 seconds to initialize resources.

```
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 1.21e-10) 5.59506e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Step Size: 1.000
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 5.10e-11) 5.16533e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Step Size: 1.000
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 3.16e-11) 2.03666e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Step Size: 1.000
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 2.14e-11) 2.18563e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Step Size: 1.000
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 1.49e-11) 2.37818e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Step Size: 1.000
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 9.08e-12) 1.63624e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Step Size: 1.000
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Val and Grad Norm: 1.14502 (rel: 5.71e-12) 1.10856e-06
23/04/27 22:32:08 INFO breeze.optimize.OWLQN: Converged because gradient converged
Accuracy: 0.6163623344074471
23/04/27 22:32:11 INFO org.sparkproject.jetty.server.AbstractConnector: Stopped Spark@3e49a872{HTTP/1.1, (http/1.1)}{0.0.0.0:0}
```

Figure 13

We submitted a job on spark cluster as shown in **Figure 10.** In **figure 13**, The job was successful, and the accuracy of the model was printed in the output, indicating an accuracy of **0.61**. Initially, the model's accuracy was determined to be 0.44, however, through further analysis of the dataset and subsequent changes made to the code, the accuracy was ultimately improved to 0.61.

X. Individual Contributions

For this project, our team engaged in numerous synchronous and asynchronous meetings on google meet. Together, we collaborated to identify and select a suitable project idea, a process that required several brainstorming sessions.

Additionally, we held multiple discussions to establish the best course of action for our project, and collectively set up a Google Cloud account, created clusters and various other things required for our project during a group session on Google Meets. As we encountered technical difficulties, we banded together to troubleshoot and resolve the issues. In terms of the coding process, we all conducted extensive research to find relevant examples and methods for implementing our project. Overall, we worked collaboratively and effectively, with each team member making valuable contributions to the project.

XI. Conclusion

The main objective of our project was to design and execute a machine learning model by utilizing the Google Cloud platform, with the purpose of accurately predicting the most suitable tags for queries posed on Stack Overflow. Through this project, we acquired knowledge and hands-on experience on several aspects of the Google Cloud platform, such as creating spark clusters, writing spark code, and training a machine learning model. As the field of technology continues to evolve, cloud computing is widely regarded as the future of this industry, and our project provided us with a valuable opportunity to gain a firsthand understanding of this exciting technology.

REFERENCES:

[1] Facebook Recruiting III - Keyword Extraction | Kaggle

[2]https://www.youtube.com/watch?v=psnbsRSSq7U&ab_channel=ianburkett