Final Project Report

Project: YouTube Analyzer

Team: BitsN'Bytes
Team Members: Pinaki Prasad, Vishnu Priya, Priyanka, Jennifer

1. Introduction:

The emergence of Big Data (structured, unstructured, semi-structured) has created a need for businesses to develop new ways of managing and analyzing this data in order to gain insights and make better decisions. With the help of advanced technologies such as artificial intelligence, machine learning, and predictive analytics, companies are now able to process large amounts of data quickly and accurately. This has enabled them to gain valuable insights into customer behavior, market trends, and other important business metrics. YouTube is a widely used and engaging social media platform that allows users to give feedback on videos through comments, likes, dislikes, and the number of subscribers for a particular channel. This data can be used to gain insight into user behavior, video content, categories, and community interests. View Counts, Likes, Votes, and Comments are all traditional data points that YouTube collects and analyzes.

Many businesses are now uploading their product launches to YouTube, eagerly awaiting feedback from their subscribers. Movie studios are also releasing trailers and people are giving their initial reactions and reviews, which helps to generate hype and anticipation for the movie. This makes the data collected from these sources very important for companies so they can analyze it and gain an understanding of how customers feel about what they offer. So, our project aims at building a YouTube Analyzer, which can be used by any business or a content creator who wants to derive insights about their channel or the business and improve their content and products in the future. A back story to, how the model of YouTube analyzer works- There was a video that was created and produced by YouTube annually from 2010-2019, which sums up all the top liked videos, top trended videos, top-most viewed videos, topmost influential videos in the form of 'YOUTUBE ЯEWIND'. Later, summary of all the videos mentioned above are made and put on to a trends page on their official site.

2. Problem Statement:

Implement a 'YouTube Analyzer', using Python and Spark Framework, and the following functions are provided in the Analyzer:

1. Find Top k categories in which the most number of videos are uploaded.
2. Find Top k rated videos and top k most popular videos.
3. Range queries:
    a. Find all videos in categories X with duration within a range [t1, t2]
    b. Find all videos in categories X with duration within a size range [x,y]

4. Degree Distribution (in degree, out degree)
5. Visualizations mapping different variables
6. Influence Analysis- Page Rank Algorithm.

3. Related Work:

This model draws on the same principles as those used by major tech companies to make recommendations [1]. It considers the user's past activities and trends to suggest items they may be interested in. By analyzing data and identifying patterns, it can provide personalized recommendations tailored to the individual user.

In [2], the authors presented the architecture and utility of Apache Spark to the community. They discussed its programming model, which includes RDDs and parallel computing, as well as a few implementations in the environment. In [3], they introduced Apache Sparks machine learning library, MLlib, and its core features and components. [4] analyzed Sparks primary framework by running a sample ML instance on it, and presented comprehensive results based on their evaluations that highlighted Sparks advantages. Similarly, [5] used Spark to analyze twitter data, while [6] performed twitter sentiment analysis using Apache Spark. These works demonstrate the significance of using a tool like Spark for large datasets such as Twitter Data Analysis.

4. Models, Algorithm, and Methods:

The analyzer has all the following functions implemented-

a. Search Algorithms
b. Visualizations
c. Network Aggregation
d. Page Rank

The Search Algorithms are implemented both in Python and Spark and the data pre-processing for the implementation in Python is carried out using R programming and its libraries, and for the Spark framework, it was carried out using Pyspark library supported by Apache Spark to implement in Python. The network aggregation, PageRank algorithms are executed in Spark.

To summarize, the following are the languages and frameworks used in building the analyzer:

1. Python
2. Apache Spark- Pyspark library in Python
3. Tableau for visualizations.

Apache Spark:

Apache Spark is an open-source distributed computing framework used for big data processing. It is designed to provide high-level APIs in Java, Scala, Python and R. It also supports a wide range

of data sources including HDFS, Cassandra, HBase and S3. Spark provides an optimized engine for large-scale data processing with in-memory computing capabilities and advanced analytics such as machine learning and graph processing. It is also highly scalable and can be deployed on-premise, in the cloud or in a hybrid environment.

Data Preprocessing:

```
data = read.csv ("data.csv", fill = TRUE)
str(data)
library(tidyverse)
data <-data %>%unite ("related_IDs",10:29, sep =',')
data <-subset (data, select = -c (11, 29))colnames(data) <-
c("video_ID","uploader","age","category","length","views","rate","ratings","comments"
,"relatedIds")
```

For implementing the Network aggregation algorithm for efficiently reporting the statistics of YouTube video network for degree distribution (including in-degree and out-degree); average degree, maximum and minimum degree, the following algorithm has been followed:

Algorithm 1:

Input: The .txt data provided by the professor is analyzed and the data is stored in the form of a list of the video IDs only.

Output: The indegree and outdegree of the nodes (represented by the video IDs)

Goal of the Solution: Analyzing the degree distribution (including in-degree and out-degree); average degree, maximum and minimum degree.

Pseudo-code:

Step 1: Build the Spark Session
Step 2: Convert the txt file to a graph format
Step 3: Call the following function InDegreeOutDegree

```
function InDegreeOutDegree (input ID lists, n):
        initialize indegree ← 0 for all IDs in the network
        initialize outdegree ← 0 for all IDs in the network
        for id1 ∈ ID of the YouTube network do
            l ← list of the YouTube IDs connected to id
            outdegree[id1] ← length(l)
            for each id2 ∈ the list of outdegree[id1] do
```

```
                    indegree[id2] ← indegree[id2] + 1
        initialize min_deg ← 0
        initialize max_deg ← 0
        initialize avg_deg ← 0
        initialize tot_deg ← 0
        for id ∈ ID of the YouTube network do
            temp_deg ← indegree[id] + outdegree[id]
            if temp_deg < min_deg do
                min_deg ← temp_deg
            if temp_deg > max_deg do
                max_deg ← temp_deg
            tot_deg← tot_deg+ temp_deg
        avg_deg← tot_deg / number of IDs
```

Step 4: spark.stop()

Algorithm 2:

Input: The .txt data provided by the professor

Output: The top k categories of data.

Goal of the Solution: Analyzing and find out the top k categories of the data in the given dataset.

Pseudo-code:

```
# importing module
from pandas import *
import collections
from operator import itemgetter
from itertools import chain
import operator

# reading CSV file
data = read_csv("/Users/priyanka/Desktop/Big Data/Project/csv_youtube.csv")

# converting column data to list
category = data['category'].tolist()

# printing list data
list_category = [ ]
```

```
for cat in category:
   val = str(cat)
   list_category.append(val.replace(u'\xa0', u' '))
#print('Category:', list_category)
print("Please enter the value of k :")
k = input()
temp = []
dict_cat = {}
# memoizing count
for sub in list_category:
      if sub != 'nan':
         if sub in dict_cat:
            val = dict_cat[sub]
            val = val+1
            dict_cat[sub]= val
         else:
            dict_cat[sub]= 1
#def sort_dict_by_value(dict_cat, reverse = False):
#  return dict(sorted(dict_cat.items(), key = lambda x: x[1], reverse = reverse))
sorted_d = dict( sorted(dict_cat.items(), key=operator.itemgetter(1),reverse=True))
print("Category with frequencies :")
print(sorted_d)
print("Top K frequent categories: ")
val = 1
for key in sorted_d.keys():
   val = val+1
   if val<=int(k):
      print(key)
```

Algorithm 3:

Input: The .txt data is analyzed and processed using spark, and the data is stored in the form of a table.

Output: frequency of videos partitioned by a search condition

Goal of the Solution: Categorized statistics: frequency of videos partitioned by a search condition such as categorization, size of videos, view count, etc.

Pseudo-code:
Step 1: Import the required libraries
Step 2: Build the Spark Session
Step 3: Convert the txt file to tabular format

Step 4: Use the proper libraries to write queries with the desired condition, which will return the output.
Step 5: spark.stop()

Algorithm 4:

Input: The input for the YouTube Analyzer project is the YouTube data from https://netsg.cs.sfu.ca/youtubedata/ The data is pre-processed in python and then used for the implementation of algorithms for the project.

Output: The PageRank values of al the ID nodes and also can find out top k most influential videos in the YouTube Network.

Goal of the Solution: To develop a PageRank algorithm which can help us finding out the most influential YouTube videos.

Pseudo-code for PageRank:
Step 1: Import the required libraries(pyspark,graphFrame)
Step 2: Build the Spark Session
Step 3: Convert the txt file to a graph format
Step 4: Use the proper libraries and functions to implement page rank
Step 5: pagerank(resetProbability, tol) # pagerank function in graphFrame package
Step 6: vertices.show()
Step 7: edges.show()
Step 8: spark.stop()

Algorithm 5:

Input: The user id and the search conditions such as categorization, size of videos, view count, etc.

Output: The list of videos that match the user's search conditions

Goal of the Solution: Find all occurrences of a specified subgraph pattern connecting users and videos with specified search conditions.

Pseudo-code for user identification in recommendation patterns:
Step 1: Import the required libraries
Step 2: Read the file and then convert the txt file to a graph format
Step 3: Prompt user for user id and search condition
Step 4: Split the line into different fields, assign a variable to each field
Step 5: Check if the user id is valid and if the search condition is true
Step 6: Traverse through the graph to find the matching entries

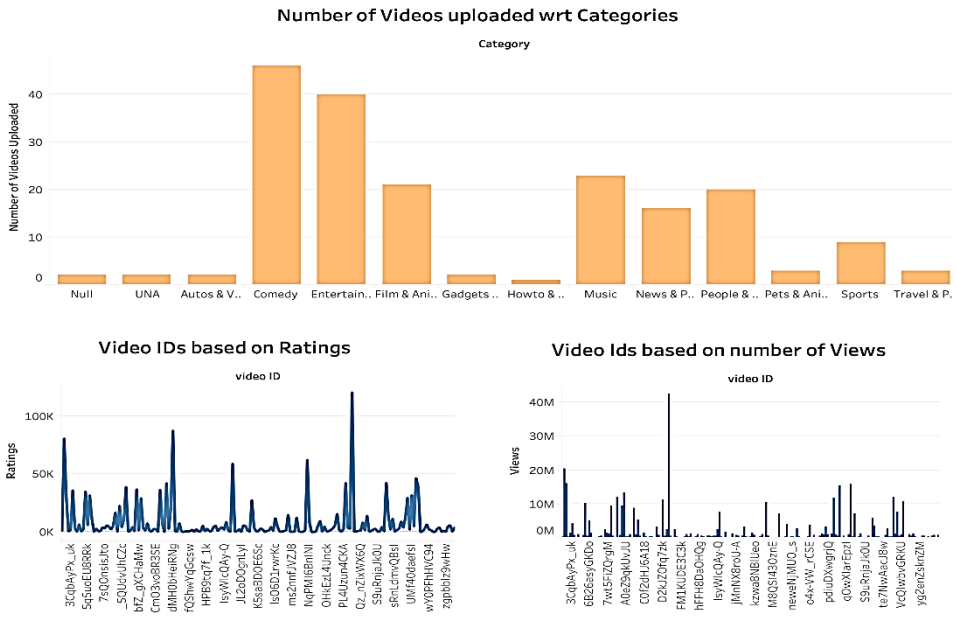
Step 7: Return the result

Optimization Techniques:

It is possible to implement the Pagerank algorithms with just python and using libraries like networkx, but to ensure that the algorithms work for larger datasets and deliver results at a highspeed, a combination of python and spark frameworks would provide us better results. So, implementing the algorithms using frameworks like Hadoop or spark gives us better results and at a faster pace.

5. Results and findings:

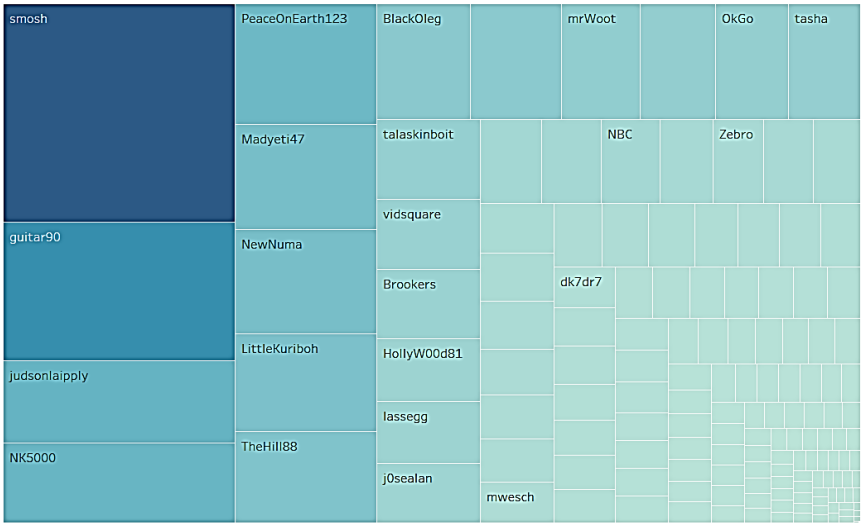
Visualizations: For visualization, we have used Tableau for its easy drag and drop mechanism. The visualization below is done in accordance with the findings of the algorithm. To align with what the algorithm has found, the visualizations support the same.



In order to make our data easier to understand and interpret, we made some visualizations. This allows us to get a higher-level understanding of what the data represents. These visualizations would be useful to the user. These visualizations can make it easier to see why certain videos are trending and recommend trending videos to users. The tree map below represents the number of comments that a video uploader received for their videos. The bigger the size of the block, the

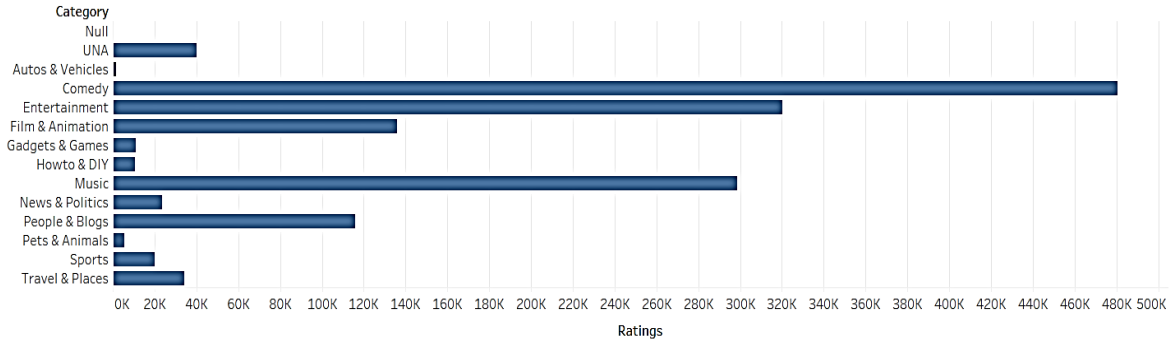
Video Uploader and Comments:

Statistical Analysis



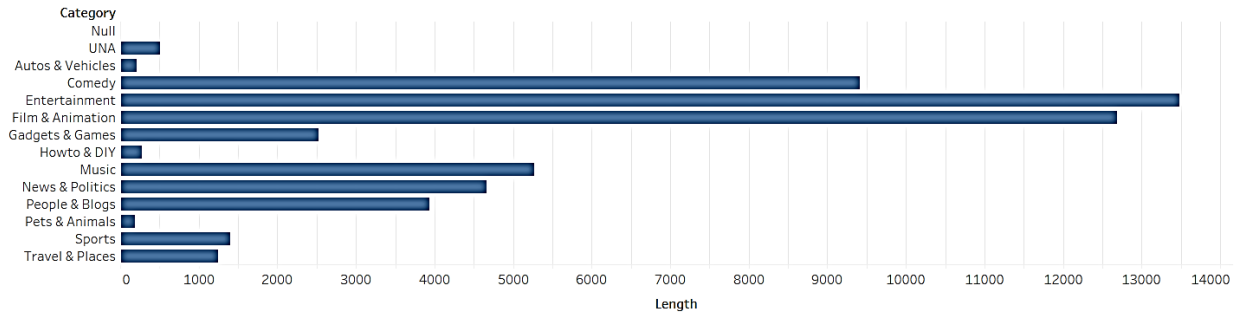
greater the number of comments they got and vice versa. Thus, the top left corner shows that Smosh had the highest number of comments amongst the other video uploaders.

Category vs Rating



The category vs rating horizontal bar chart above represents the relationship between categories and ratings. The comedy category had the highest ratings, while auto & vehicles had the lowest ratings.

Category vs Video Length



Spark Results showing the indegrees of the nodes:

```
g.inDegrees.show()
```

```
+-----------+--------+
|         id|inDegree|
+-----------+--------+
|71agSyF_TYo|       4|
|MydHXSjZgp4|      19|
|CVqSmBSEe0I|       1|
|z-fQEgmQFXE|       1|
|NBSQWuujn-I|       1|
|82nR5p-XDs8|       2|
|OdiwzztQtTE|       1|
|pSnaNtvfTqg|       1|
|grZIe1wKEH4|       1|
|9iS97D0Rkko|       2|
|8vuIsp5J-bU|       1|
|zopZmUhKduI|       1|
|OKiXkQYatfA|       1|
|JaOoaE8h7n8|       1|
|-6TUKuLXaJU|       1|
|SxxBsNbdhhY|       1|
|Sr5p09qkG34|       1|
|gd2qoz5eL8k|       6|
|BCQvVTGng28|       1|
|bhVvC3zuvSY|       2|
+-----------+--------+
only showing top 20 rows
```

```
# MAX indegree
g.inDegrees.agg({"inDegree": "max"}).show()
```

```
+-------------+
|max(inDegree)|
+-------------+
|         2906|
+-------------+
```

```
# MIN indegree
g.inDegrees.agg({"inDegree": "min"}).show()
```

```
+-------------+
|min(inDegree)|
+-------------+
|            1|
+-------------+
```

```
# AVG indegree
g.inDegrees.agg({"inDegree": "avg"}).show()
```

```
+------------------+
|     avg(inDegree)|
+------------------+
|2.1884406983744733|
+------------------+
```

Spark Results showing the outdegrees of the nodes:

```
g.outDegrees.show()
```

```
+-----------+---------+
|         id|outDegree|
+-----------+---------+
|MydHXSjZgp4|       20|
|WkY4gKaPNBw|       20|
|gd2qoz5eL8k|       20|
|_9fZtJsq1gI|       20|
|jKxg40VySI4|       20|
|ZyWgo4QMrfQ|       20|
|zMSpTvv9tg8|       20|
|2o3lFsf3fkQ|       20|
|3866CS6QVC8|       20|
|8XrLxo15i48|       20|
|hmh1xt7yiVs|       20|
|_hDsHDjERA0|       20|
|rQDqbpq3lpU|       20|
|yUWD2vZbO84|       20|
|GcIEpyem6XY|       20|
|ZqF5_OD-4Xw|       20|
|0Kc4_YpOHlM|       20|
|dATGTZm3Ypc|       20|
|7ZCjnSkpyZc|       20|
|ntuFXBvwIl8|       20|
+-----------+---------+
only showing top 20 rows
```

```
# MAX outdegree
g.outDegrees.agg({"outDegree": "max"}).show()
```

```
+-------------+
|max(outDegree)|
+-------------+
|           20|
+-------------+
```

```
# MIN outdegree
g.outDegrees.agg({"outDegree": "min"}).show()
```

```
+-------------+
|min(outDegree)|
+-------------+
|           20|
+-------------+
```

```
# AVG outdegree
g.outDegrees.agg({"outDegree": "avg"}).show()
```

```
+-------------+
|avg(outDegree)|
+-------------+
|         20.0|
+-------------+
```

Results showing the Top k Search Queries:

```
# B. Search
# - top k queries
g.vertices.groupBy("category").count().sort(f.col("count").desc()).show()
```

```
+----------------+-----+
|        category|count|
+----------------+-----+
|           Music|   22|
|Autos & Vehicles|   17|
| News & Politics|   14|
|     Howto & DIY|   10|
|  People & Blogs|    9|
|   Entertainment|    7|
|          Sports|    7|
| Gadgets & Games|    6|
|          Comedy|    5|
|            UNA |    2|
|Film & Animation|    2|
| Travel & Places|    1|
+----------------+-----+
```

Results for the Range Queries – Videos belonging to Category Sports between a length range:

```
# Range queries: find all videos in categories X with duration within a range [t1, t2]; find all
g.vertices.filter((f.col('length').between(0, 9999999999)) & (f.col('category') == 'Sports')).show()
```

```
+-----------+-------+--------+-----------+-----------+---+--------+------+-----+----+-------+--------+
|         id|length_|    size|   video_id|   uploader|age|category|length|views|rate|ratings|comments|
+-----------+-------+--------+-----------+-----------+---+--------+------+-----+----+-------+--------+
|vy74CnRaQgs|    420|17390492|vy74CnRaQgs|  supreme84|697|  Sports|   420|35186|4.68|     44|      51|
|3qNVLn0rNYk|    311|12966459|3qNVLn0rNYk|     jhsieh|502|  Sports|   311|10683|4.53|     15|       3|
|8CCSF0G84HQ|    219| 7661791|8CCSF0G84HQ|  truckitup|517|  Sports|   219| 2276|4.33|      3|       0|
|vyiVL0zPpaE|    254|10490566|vyiVL0zPpaE|glidinclyde|457|  Sports|   254| 2459| 5.0|      8|       7|
|S1imSSkqOB8|    242| 8730888|S1imSSkqOB8|    leasky8|671|  Sports|   242| 4113|4.35|     17|      15|
|28gYtVPOm5U|    216| 8953391|28gYtVPOm5U| xlcpuvirus|646|  Sports|   216| 4019|4.62|     16|       2|
|KzbYR_Ri3ww|     96| 3708444|KzbYR_Ri3ww|     Jlinck|658|  Sports|    96| 2573|3.33|      3|       0|
+-----------+-------+--------+-----------+-----------+---+--------+------+-----+----+-------+--------+
```

Results for Range Queries – No. of videos uploaded by an Uploader:

```
# videos with size in range [x,y].
g.vertices.groupBy("uploader").count().sort(f.col("count").desc()).show()
```

```
+---------------+-----+
|       uploader|count|
+---------------+-----+
|MyCadillacStory|   11|
|     lilscrappy|    9|
|       heniadir|    3|
|  bigwallypants|    2|
|        thirdd3|    2|
|          reeuh|    1|
|      truckitup|    1|
|       yatucamp|    1|
|     xlcpuvirus|    1|
|        jensyao|    1|
|       kylaalee|    1|
|     bishop8000|    1|
|    glidinclyde|    1|
|       gweebage|    1|
|     liamtipton|    1|
| DanielSundgren|    1|
|  coyoteboy1983|    1|
|     YagoCooper|    1|
|          VVasp|    1|
|        erkut89|    1|
+---------------+-----+
only showing top 20 rows
```

Results of the PageRank Algorithm – Showing PageRank values of Each Node:

id	length_	size	video_id	uploader	age	category	length	views	rate	ratings	comments	pagerank
vy74CnRaQgs	420	17390492	vy74CnRaQgs	supreme84	697	Sports	420	35186	4.68	44	51	0.4049279991415213
6hJCfXllrqE	309	13497109	6hJCfXllrqE	DanielSundgren	613	Music	309	1230	2.67	3	0	0.4049279991415213
GvIFh3por14	538	22236234	GvIFh3por14	CharlesCoburn	690	Entertainment	538	12788	4.56	25	41	0.4049279991415213
Eykcs8v77N0	9	352917	Eykcs8v77N0	gweebage	600	People & Blogs	9	12176	3.67	9	0	0.4049279991415213
LqiLaTpIhD4	149	5994499	LqiLaTpIhD4	MyCadillacStory	722	Autos & Vehicles	149	5178	4.2	20	16	2.554700438385539
uObTAvVz2iY	27	1005034	uObTAvVz2iY	MyCadillacStory	716	Autos & Vehicles	27	2914	4.5	8	3	2.554700438385539
Q6KmM2_jsR0	48	1895868	Q6KmM2_jsR0	MyCadillacStory	698	Autos & Vehicles	48	16020	4.42	31	22	2.554700438385539
6qc9bMuffoE	30	1219883	6qc9bMuffoE	MyCadillacStory	687	Autos & Vehicles	30	2996	4.3	10	4	2.554700438385539
0ZH7NBiruqk	52	2102878	0ZH7NBiruqk	MyCadillacStory	675	Autos & Vehicles	52	3320	4.3	10	5	2.554700438385539
CycZagvqJps	49	1898621	CycZagvqJps	MyCadillacStory	675	Autos & Vehicles	49	2299	4.09	11	2	2.554700438385539
g9m-Mjk9a44	96	3942324	g9m-Mjk9a44	MyCadillacStory	675	Autos & Vehicles	96	11306	4.17	24	19	2.554700438385539
CMNaxTCbgJU	9	311645	CMNaxTCbgJU	MyCadillacStory	675	Autos & Vehicles	9	4125	4.29	7	3	2.554700438385539
qsrnHkeKtqs	32	1335841	qsrnHkeKtqs	MyCadillacStory	675	Autos & Vehicles	32	1020	4.0	4	1	2.554700438385539
vXZEPt7mbgY	64	2585631	vXZEPt7mbgY	MyCadillacStory	675	Autos & Vehicles	64	2939	5.0	10	6	2.554700438385539
wo8ceXX21oc	124	4764966	wo8ceXX21oc	iPhoneusers	695	News & Politics	124	805	4.6	5	0	0.7491167984118142
Am56liCVcaw	587	17231193	Am56liCVcaw	matthewisthebest	693	Howto & DIY	587	13980	3.81	16	16	1.0416772777915633
tyYUJUl3lNs	237	9750888	tyYUJUl3lNs	phatteningphilms	703	News & Politics	237	237	5.0	2	1	0.4049279991415213
EqNBeSp1MRw	594	24660894	EqNBeSp1MRw	bucqui	551	News & Politics	594	2671	3.03	38	8	0.4049279991415213
r92ZCONFStk	270	10808997	r92ZCONFStk	thewinekone	520	Comedy	270	179877	4.71	4632	2256	0.4049279991415213
jUGBWD6Xo44	241	9995583	jUGBWD6Xo44	HeartacheXTears	627	Music	241	890	4.56	9	7	0.4049279991415213

only showing top 20 rows

Results of the PageRank Algorithm – Weights of the Edges:

src	dst	weight
TFWNpS4E4fM	gSP9EnBHI10	0.25
BU7L9OawuMI	gSP9EnBHI10	0.25
q8u2TSrw_Pc	r76T2EG90MA	0.2
_yUo9crgm3U	r76T2EG90MA	0.16666666666666666
LqiLaTpIhD4	6qc9bMuffoE	0.09090909090909091
uObTAvVz2iY	6qc9bMuffoE	0.09090909090909091
Q6KmM2_jsR0	6qc9bMuffoE	0.09090909090909091
6qc9bMuffoE	6qc9bMuffoE	0.09090909090909091
0ZH7NBiruqk	6qc9bMuffoE	0.09090909090909091
CycZagvqJps	6qc9bMuffoE	0.09090909090909091
g9m-Mjk9a44	6qc9bMuffoE	0.09090909090909091
CMNaxTCbgJU	6qc9bMuffoE	0.09090909090909091
qsrnHkeKtqs	6qc9bMuffoE	0.09090909090909091
vXZEPt7mbgY	6qc9bMuffoE	0.09090909090909091
v28pkP3WCLw	6qc9bMuffoE	0.09090909090909091
AQXj7ANHP-Q	40NRsg0UsA0	0.1111111111111111
Hyfmg9zrbYI	40NRsg0UsA0	0.1111111111111111
40NRsg0UsA0	40NRsg0UsA0	0.1111111111111111
abkj32lgDKc	40NRsg0UsA0	0.1111111111111111
1FTwiGn-Jd4	40NRsg0UsA0	0.1111111111111111

only showing top 20 rows

Result Analysis and Conclusion:

Apache Spark is compatible with Python through its PySpark API. PySpark is a Python API for Spark that allows developers and data scientists to access the power of Apache Spark from Python. It provides a simple programming interface that makes it easy to write big data programs in Python, and it supports a wide range of libraries including NumPy, Pandas, SciPy, and scikit-learn. PySpark also provides a powerful distributed computing platform that can be used to process large datasets in parallel. With PySpark, developers can quickly and easily build distributed applications that can scale to handle large amounts of data. Additionally, PySpark is compatible with popular

Python libraries such as TensorFlow and Keras, making it easy to integrate machine learning models into Spark applications. Apache Spark's compatibility with Python makes it an ideal choice for data scientists and developers who want to build powerful distributed applications quickly and easily. In particular, we used the GraphFrames package, to perform the network aggregation, and implement the page rank algorithm. GraphFrames is a package for Apache Spark that provides DataFrame-based Graphs. It enables users to manipulate and analyze graph data using the familiar DataFrame API. It also provides a set of algorithms for graph analysis, such as PageRank, connected components, and shortest paths. GraphFrames also supports user-defined algorithms, allowing users to extend the library with their own custom algorithms. With GraphFrames, users can easily combine graph analysis with other data processing tasks in Spark, making it an ideal tool for large-scale graph analytics.

We have used it to find out the indegrees and outdegrees of each node (represented by a video). And from the result, we can see that the maximum indegree is 2906, the minimum outdegree is 1, and the average value is 2.188. This indicates that most of the videos have indegree 1, which makes the average less. Whereas, for outdegree, we see that the maximum, minimum, and average outdegree is 20. That means, all the nodes have an outdegree of 20. This makes sense, because in the dataset itself, contains 20 reference videos for each video.

The search algorithms work efficiently with the usage of Pyspark library, compared to Python. From the top k search algorithm, we can see that the category Music contains the maximum number of videos (22), followed by Autos and Vehicles (17). Whereas, the category Travel and Places ranks the lowest with just 1 video. Similarly, for the search query ranking the number of videos uploaded by a user, we see that MyCadillacStory uploaded 11 videos, which was the highest. Whereas, Erkut89 uploaded just 1 video, which was one of the lowest.

PageRank algorithm results show that the top result that we got is the most influential video with a PageRank value of 2.55, whereas, the lowest PageRank value we obtained for our network is 0.40. Also, from the second table we can see the weights assigned to each edge connecting these videos. The maximum weight is 0.25, whereas the lowest weight is 0.09.

References:

[1] Lu, Zhongqi, Zhicheng Dou, Jianxun Lian, Xing Xie, and Qiang Yang. "Content-based collaborative filtering for news topic recommendation." In *Twenty-ninth AAAI conference on artificial intelligence*. 2015.

[2] Zaharia, Matei, Mosharaf Chowdhury, Michael J. Franklin, Scott Shenker, and Ion Stoica. "Spark: Cluster computing with working sets." In *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*. 2010.

[3] Meng, Xiangrui, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman et al. "Mllib: Machine learning in apache spark." *The Journal of Machine Learning Research* 17, no. 1 (2016): 1235-1241.

[4] Fu, Jian, Junwei Sun, and Kaiyuan Wang. "Spark–a big data processing platform for machine learning." In *2016 International Conference on Industrial Informatics-Computing Technology, Intelligent Technology, Industrial Information Integration (ICIICII)*, pp. 48-51. IEEE, 2016.

[5] Nair, Lekha R., and DR Sujala D. Shetty. "STREAMING TWITTER DATA ANALYSIS USING SPARK FOR EFFECTIVE JOB SEARCH." *Journal of Theoretical & Applied Information Technology* 80, no. 2 (2015).

[6] Nodarakis, Nikolaos, Spyros Sioutas, Athanasios K. Tsakalidis, and Giannis Tzimas. "Large Scale Sentiment Analysis on Twitter with Spark." In *EDBT/ICDT Workshops*, pp. 1-8. 2016.