YouTube Data Analyzer
Final Report

Team Kongqueror

Ross Kugler, Huy Ky, Ben Bordon

December 3rd, 2025

Table of Contents

Abstract

This project presents an end-to-end YouTube Data Analyzer designed to process and analyze large-scale video network data using modern big data technologies.

We developed a YouTube Data Analyzer that processes large-scale video network data using MongoDB, Apache Spark, and a Streamlit GUI. The system supports network statistics, top-K queries, range searches, subgraph pattern detection, and PageRank-based influence analysis. Using early YouTube crawl data, we implemented scalable algorithms and visualizations to uncover trends in video categories and popularity. Future work includes cloud deployment and performance optimizations.

Problem Statement

We chose a sample project from the list provided on Canvas:

Implement a Youtube data analyzer supported by NoSQL database, Hadoop MapReduce, and Spark GraphX/GraphFrame. The analyzer provides basic data analytics functions to Youtube media datasets. The analyzer provides following functions for users:

- *Network aggregation: efficiently report the following statistics of Youtube video network:*
  - *Degree distribution (including in-degree and out-degree); average degree, maximum and minimum degree*
  - *Categorized statistics: frequency of videos partitioned by a search condition: categorization, size of videos, view count, etc.*
- *Search:*
  - *Top k queries: find top k categories in which the most number of videos are uploaded; top k rated videos; top k most popular videos.*
  - *Range queries: find all videos in categories X with duration within a range [t1, t2]; find all videos with size in range [x,y].*
  - *User identification in recommendation patterns: find all occurrence of a specified subgraph pattern connecting users and videos with specified search condition.*
- *Influence analysis:*
  - *Use PageRank algorithms over the Youtube network to compute the scores efficiently. Intuitively, a video with high PageRank score means that the video is related to many videos in the graph, thus has a high influence.*
  - *Effectively find top k most influence videos in Youtube network. Check the properties of these videos (# of views, # edges, category…). What can we find out? Present your findings.*

Note:

- *Where applicable, develop effective optimization techniques to speed up the algorithm you used, including indexing, compression, or summarization.*

Overall, the project involves efficiently storing, processing, and analyzing large-scale YouTube video network data to extract meaningful insights. The goal of the project was to create an End-to-End Application with GUI, following the description above.

Dataset

The dataset this project is based on is the *"Statistics and Social Network of YouTube Videos"* by Xu Cheng and Cameron Dale and Jiangchuan Liu.

Link: http://netsg.cs.sfu.ca/youtubedata/

Description: *"a dataset containing several early large-scale crawls of YouTube's video network. The crawler used breadth-first search up to a max of 20 levels. Each video is a node; an edge from video A → B exists if B appeared in the top 20 "related videos" list of A."*

Team

Our team is composed of three senior software engineering undergraduate students at WSU Everett. All members have taken CPTS 451 – Introduction to Database Systems, and have experience with relational databases, SQL, PostGreSQL. We have worked on several projects utilizing databases, including Microsoft Azure SQL Server+Databases, Azure Storage Account Tables, and SQLite DBs. We are experienced in several coding languages, including Python, C#, C++, and Java. Ben has unique experience in taking CPTS 489 – Web Development.

Table 1 outlines the dedicated roles the team took on during the project development, although our team collaborated extensively on all aspects of the project.

Table 1. Team Member Responsibilities

Team Member	*Responsibility*
Ross Kugler	Data Engineering and Spark Integration
Huy (Harry) Ky	Spark Cluster setup and Query Optimization
Benjamin Bordon	Frontend Development and Visualization

Related Work

Based on our research on similar projects, we have found three articles that best relate to our work. All three involve using big data tools such as Hadoop or Apache Spark to analyze YouTube data and extract insights.

1. ***"EXPLORATION OF YOUTUBE STATISTICS DATA USING HADOOP TECHNOLOGIES"***

https://www.researchgate.net/publication/334549087_Exploration_of_Youtube_Statistics_Data_using_Hadoop_Technologies

Presents a big data project analyzing factors influencing YouTube video success using 38 datasets from the YouTube Data API and Kaggle. It applies Hadoop ecosystem tools—R for cleaning, HDFS for storage, and frameworks like MapReduce, Pig, Hive, and Spark—to study upload timing, top categories by country, and high-engagement channels, with results visualized in Tableau and Power BI

2. ***"BIG DATA ANALYSIS ON YOUTUBE USING HADOOP and MAPREDUCE"***

https://ieeexplore.ieee.org/document/9012635

Combined many YouTube statistics datasets (via public datasets like Kaggle + supplemental scraping) and used big-data tools (Hadoop MapReduce, Apache Pig, Hive, Spark) for processing and analysis; output visualized via tools (e.g. Tableau, Power BI). They studied factors that correlate with likes/dislikes, comments, views depending on time of day, etc.

3. ***"Real Time Sentiment Analysis of YouTube Live Chat With Apache Spark and Kafka"***

https://ieeexplore.ieee.org/document/11218164

Presents a real-time system using Kafka and Spark to analyze YouTube Live chat sentiment. Logistic regression performed best (~99% accuracy) with low latency. The system includes monitoring/visualization and handles high-volume live events, though chat noise remains a challenge.

Big Data Solution

**All project files are located in our GitHub Repository, https://github.com/rk3026/Youtube_Analyzer.*

Architecture

Our architecture involves four main components – a parser, MongoDB database, Apache Spark algorithm set, and a Streamlit GUI app.

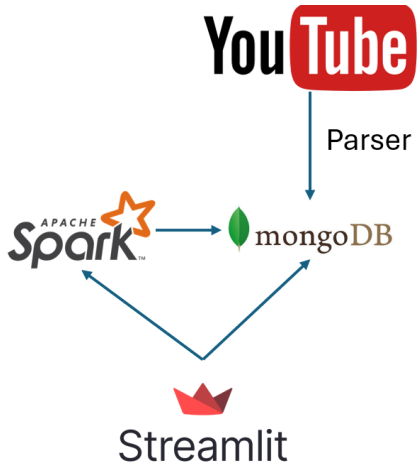


Figure 1. Architecture Overview

- Parsing Algorithm
  - Parses the raw crawl .txt files from the initial dataset
  - Cleans the data of missing/invalid values
  - Inserts into MongoDB
- MongoDB instance
  - 4 collections
    - crawls
    - edges
    - video_snapshots
    - videos
- Apache Spark
  - connects to MongoDB via the official connector
  - algorithms written in Python with PySpark
  - runs in standalone mode, requires at least 1 worker with 4 GB of memory
- Python GUI application using Streamlit framework
  - Connects to both MongoDB and Spark
  - Loads precomputed algorithm results from MongoDB
  - Can dynamically run Spark algorithms based on user queries

Data Preparation

To prepare the data for analysis, we created a python parsing algorithm ‘`process_crawl_data.py`’. This algorithm cleans and normalizes the raw crawl data and then inserts it into a MongoDB instance. The algorithm does batch insertion for efficiency.

Database System

NoSQL, document-store database: MongoDB

Non-relational schema:
4 collections in MongoDB

videos: stores static attributes for each video

```
{
 "_id": "video_id", // from the data's video id
 "uploader": "...",
 "category": "...",
 "length_sec": 213,
 “date_uploaded”: ISODate(“2007-01-01T00:00:00Z”), // calculated from age
 “seen_in_crawls”: [“0222”, “0301”, “0302”]
}
```

video_snapshots

```
{
 "_id": { "video_id": "dQw4w9WgXcQ", "crawl_id": "0222" },
 "video_id": "dQw4w9WgXcQ",
 "crawl_id": "0222",
 "age_days": 1234,              // as defined: days since YouTube establishment
 "category": "Music",
 "length_sec": 213,             // integer
 "views": 1234567,
 "rate": 4.62,                  // float
 "ratings": 9876,
 "comments": 5432
}
```

edges:

```
{
 "_id": ObjectId(),
```

```
 "crawl_id": "0222",
 "src": "dQw4w9WgXcQ",        // the video containing the "related IDs" list
 "dst": "oHg5SJYRHA0"         // one related video id
}

crawls: handy for per-crawl reporting/filters
{
 "_id": "0222",
 "date": ISODate("2007-03-01T00:00:00Z"),
 "notes": "crawled to fifth depth, but did not finish",
 "total_videos": 155513,
 "duration_sec": 93352
}
```

Indexes:

- videos
  - uploader - for filtering/grouping by channel
  - category - for category-based queries
  - date_uploaded - for temporal queries
  - length_sec - for duration-based filtering
  - seen_in_crawls - for tracking which crawls contain each video
- crawls
  - crawl_id - for looking up specific crawls
  - processed_at - for temporal queries on processing
- edges
  - src - for finding outgoing edges (related videos from a source)
  - dst - for finding incoming edges (videos that reference this one)
  - crawl_id - for filtering edges by crawl
  - (src, crawl_id) - compound index for crawl-specific edge queries
- video_snapshots
  - video_id - for looking up specific videos
  - crawl_id - for filtering by crawl
  - (video_id, crawl_id) - compound index for snapshot lookups
  - views - for view count sorting/filtering
  - rate - for rating-based queries
  - (category, views) - compound index for top videos by category

This schema is appropriate for our data because we want to analyze top-k, degree distribution, video category stats, and top video patterns. Storing videos allows for determining which videos are the most popular and their basic information. Storing edges makes the database much larger, but allows us to do graph-related queries and analysis with Spark GraphFrames. Storing the video snapshots allows us to see how videos evolve over time.

Table 2 shows statistics of our database with the first two crawls from the initial dataset.

Table 2. Database Stats

Metric	Size
Documents	19.10M
Data (uncompressed)	1.77 GB
StorageSize	0.49 GB
IndexSize	1.33 GB
Index count	21

Figure 2 shows stats about the database size after running db.stats() in the MongoDB shell.

```
PS C:\Users\rossk> mongosh mongodb://localhost:27017/youtube_analytics
Current Mongosh Log ID: 69309d0ae55456941e1e2620
Connecting to:          mongodb://localhost:27017/youtube_analytics?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.5.10
Using MongoDB:          8.2.0
Using Mongosh:          2.5.10

For mongosh info see: https://www.mongodb.com/docs/mongodb-shell/

------
   The server generated these startup warnings when booting
   2025-11-29T20:47:11.741-08:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
------

rs0 [direct: primary] youtube_analytics> db.stats()
{
  db: 'youtube_analytics',
  collections: Long('4'),
  views: Long('0'),
  objects: Long('19102596'),
  avgObjSize: 92.54287867470997,
  dataSize: 1767809224,
  storageSize: 488644608,
  indexes: Long('21'),
  indexSize: 1332555776,
  totalSize: 1821200384,
  scaleFactor: Long('1'),
  fsUsedSize: 520443277312,
  fsTotalSize: 998811254784,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1764793607, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1764793607, i: 1 })
}
```

Figure 2. Screenshot of running db.stats() on the youtube_analytics database in mongosh

Queries and Algorithms

Our application implements five analytics families over the YouTube video network: Network Aggregation, Categorized Video Statistics, Top-K Queries, Subgraph Pattern Search (Motifs), and Influence Analysis (PageRank). Additional algorithm details can be found in the appendix.

Network Aggregation

Summarizes the overall structure of the YouTube video graph. It calculates in-degree and out-degree for each video (how many videos link to it and how many it links to), then reports averages, minimums, maximums, and degree distribution. This helps us understand connectivity and identify highly linked videos.

Categorized Video Stats

Groups videos by category (e.g., Music, Entertainment, Education) and computes frequency counts. It also breaks down videos by length and view ranges to reveal trends in content type and popularity.

Top-K

Finds the "top" items based on different metrics:

- Top categories by number of videos.
- Top videos by views or ratings.
- Range queries, such as music videos within a specific duration range, ranked by views.

Subgraph Patterns

Detects small structural patterns (motifs) in the video network using Spark GraphFrames. For example, finding pairs or chains of related videos within a specific category. This uncovers hidden relationships and recommendation patterns.

Influence Analysis (PageRank)

Ranks videos by influence using the PageRank algorithm. A high PageRank score means a video is connected to many other important videos, indicating strong influence in the network. We then compare these influential videos with their views and categories.

User Interface

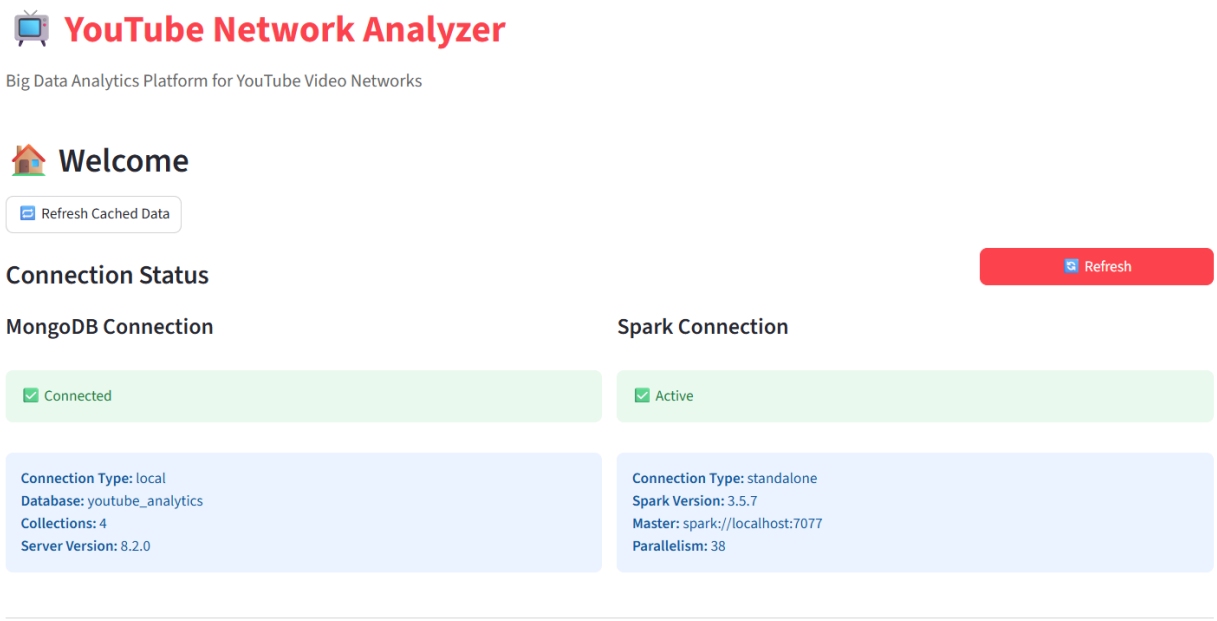
We chose to use Streamlit to build our GUI application. Streamlit is an open-source Python library used to build and share custom web apps for data science and machine learning,

using only pure Python. Overall, Streamlit allowed us to quickly prototype our application within the limited project deadlines.

Our GUI includes a home page that gives a quick overview of the data, as well as unique pages for the different algorithm categories. Within individual algorithm pages, there are different tabs for algorithm variations.

Home Page

The home page has a connection status display and dataset overview. It also has a 'getting started' section, with instructions on configuring the app and using.



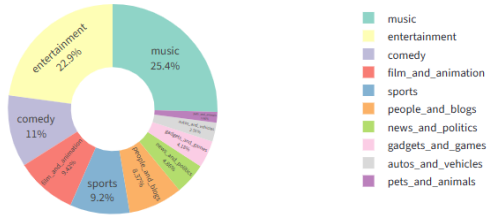
Dataset Overview

Total Videos	Total Edges	Snapshots	Avg Degree
1,844,577	30,754,839	1,886,910	N/A

Quick Analytics

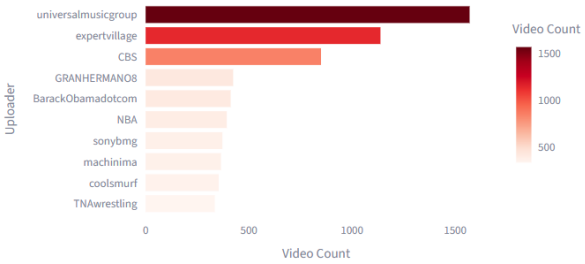
Category Distribution

Top 10 Video Categories



Top Content Creators

Top 10 Content Creators



Network Statistics Page

Network Statistics

Analyze degree distribution and categorized video statistics

Degree Distribution | Categorized Statistics

Degree Distribution Analysis

Analyze in-degree, out-degree, and total degree distributions for the video network.

Sample Size (edges) 50000

1000 100000

Run Analysis

Analysis complete!

Algorithm Performance

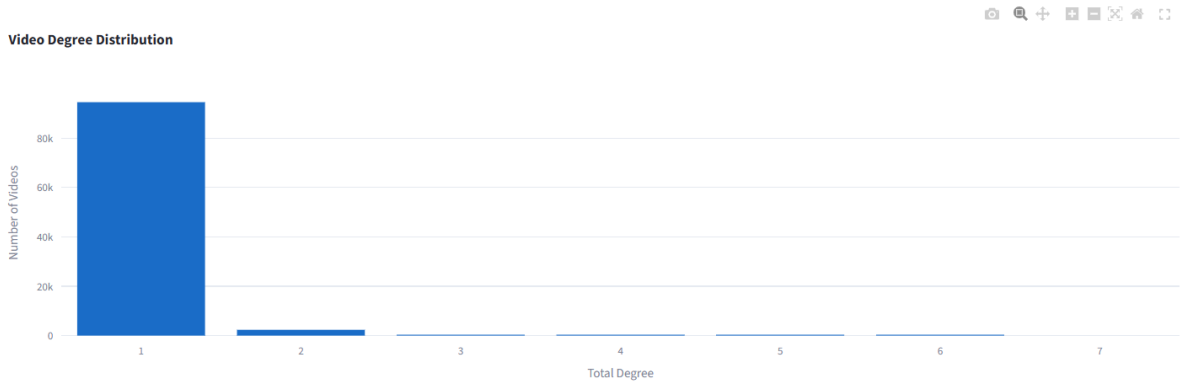
Execution Time	Rows Processed	Rows Returned	Throughput
106.656s	50,000	18	469 rows/s

Performance Details

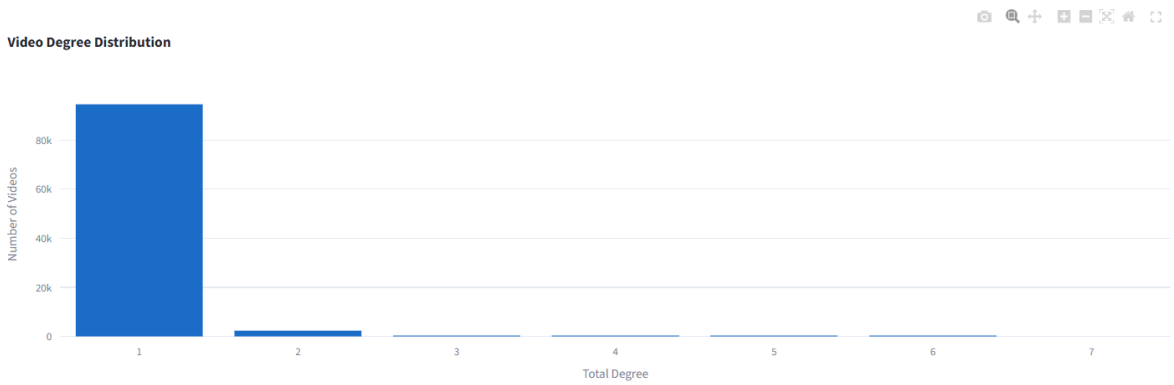
Summary Statistics

Avg Total Degree	Max Degree	Min Degree	Std Dev
1.03	7	1	0.18

Degree Distribution



Degree Distribution



Top 10 Videos by Degree

Video	Total Degree	In-Degree	Out-Degree
JrUupfFzBrA	7	7	0
dMH0bHeiRNg	6	6	0
H1BYrSk46vg	6	6	0

Top-K Queries Page

Top-K Queries

Find the most popular videos, categories, and highest rated content

Top Categories | Most Viewed Videos | Highest Rated Videos

Top K Categories by Video Count

Find the categories with the most uploaded videos.

Number of Categories (K) 50

5 50

Find Top Categories

Found top 50 categories!

Algorithm Performance

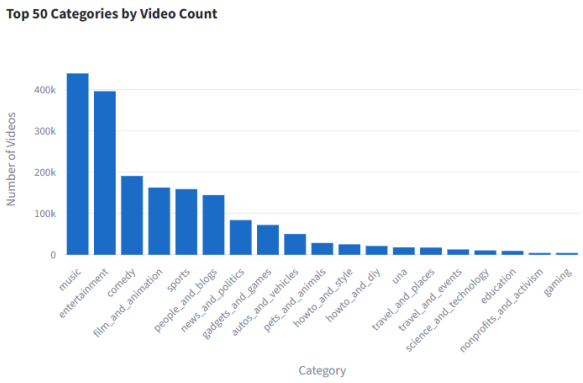
Execution Time	Rows Processed	Rows Returned	Throughput
7.907s	1,844,577	19	233,280 rows/s

Performance Details

Results Table

	Category	Video Count
1	music	438464
2	entertainment	395336
3	comedy	190471
4	film_and_animation	162443
5	sports	158718
6	people_and_blogs	144257
7	news_and_politics	83836
8	gadgets_and_games	72083
9	autos_and_vehicles	50178
10	pets_and_animals	28548

Distribution Chart



Range Queries Page

Range Queries

Filter videos by category, duration, views, and other criteria

Duration Range | Views Range

Filter Videos by Duration

Find all videos in a specific category with duration within a specified range.

Select Category: autos_and_vehicles

Duration Range (seconds): 200 – 1000

Maximum Results: 360

Search

Found 360 videos!

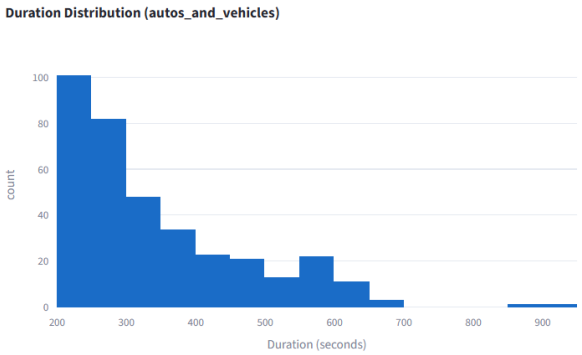
Algorithm Performance

Execution Time	Rows Processed	Rows Returned	Throughput
5.059s	1,886,910	360	372,950 rows/s

Performance Details

Results: autos_and_vehicles videos (200-1000 sec)

Video	Duration	Views	Rating
ZeBd_F2Bz5Y	4:50	8623041	4.55
ju6t-yyoU8s	5:39	7325889	4.44
uUurALr_Ckk	4:15	7258142	4.74
aCamHfJwSGU	5:24	5354163	4.11
S-ppGiwc0wQ	4:14	5039415	4.62
oPQlizRp9ck	4:58	3748272	4.69
Ggqg6q5xegE	3:28	3584892	4.73
uug6TjvRJaE	7:58	3217449	4.42
WaWoo82zNUA	9:09	2963709	4.83
_buTXh2su5U	7:43	2789746	3.54
nkfvQvc9ipI	9:50	2662782	4.78



Graph Analytics Page

Graph Analytics

Advanced network analysis using Spark GraphFrames for large-scale graph mining

PageRank Analysis | Community Detection | Centrality Metrics | Network Visualization

PageRank Analysis

Identify the most influential videos in the network based on link structure.

Sample Size (edges): 20000

Reset Probability: 0.15

Max Iterations: 10

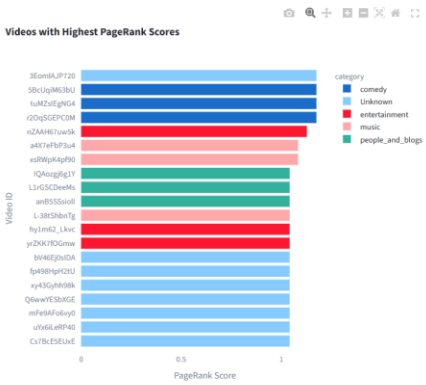
Run PageRank Analysis

Algorithm Performance

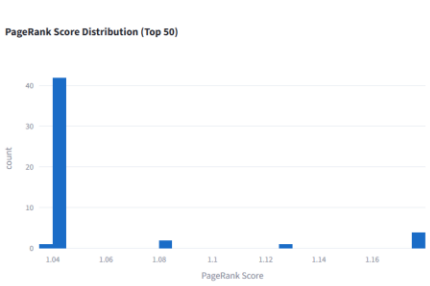
Execution Time	Rows Processed	Rows Returned	Throughput
133.655s	20,000	50	150 rows/s

Performance Details

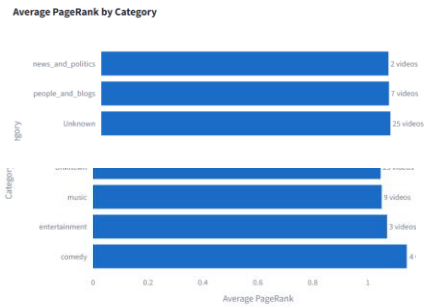
Top 20 Most Influential Videos



PageRank Distribution



PageRank by Category



Top 50 Videos by PageRank

Video	PageRank Score	Category	Uploader
r2OqSGEPC0M	1.176024	comedy	takarino0
tuMZsIEgNG4	1.176024	comedy	takarino0
3EomlAJP720	1.176024	Unknown	Unknown
5BcUqiM63bU	1.176024	comedy	TASPOclub
nZAAH67uw5k	1.128071	entertainment	takarino0
xsRWpK4pf90	1.084536	music	universalmusicgroup
a4X7eFbP3u4	1.084536	music	universalmusicgroup
Cs7BcE5EUxE	1.043128	Unknown	Unknown

Pattern Search Page

Pattern Search

Find subgraph patterns in the YouTube recommendation network

About Pattern Search

This analysis finds patterns of connected videos in the recommendation network:

- **Related Pairs:** `(a)→(b)` - Video A recommends Video B
- **Video Chains:** `(a)→(b)→(c)` - Recommendation chains
- **Common Recommendations:** `(a)→(c)←(b)` - Videos that share recommendations

These patterns help identify recommendation clusters and content relationships.

Search Configuration

Filter by Category: All Categories

Pattern Type: Related Pairs (a→b)

Maximum Results: 100

Edge Sample Size: 60000

Find Patterns

Found 100 results!

Algorithm Performance

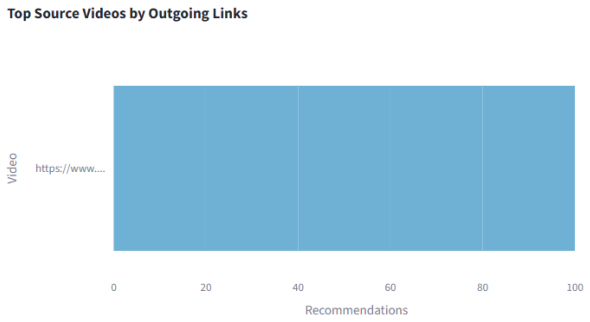
Execution Time	Rows Processed	Rows Returned	Throughput
0.338s	60,000	100	177,742 rows/s

Related Pairs (a→b) (all categories)

Results Table

	Source Video (a)	Related Video (b)
1	xUUA5xD8dJ8	x2YiYPj3tkw
2	xUUA5xD8dJ8	ZLctU3UsxFU
3	xUUA5xD8dJ8	5SAilWZxoEc
4	xUUA5xD8dJ8	94QMihIC9m8
5	xUUA5xD8dJ8	vj34U9FMtHs
6	xUUA5xD8dJ8	Db5Q8VaDSfA
7	xUUA5xD8dJ8	yNvBadXhVjA
8	xUUA5xD8dJ8	tS_wB1KoV48
9	xUUA5xD8dJ8	PgZgubuT2DM
10	xUUA5xD8dJ8	WK_HZGJoEgA

Pattern Visualization



Results and Findings

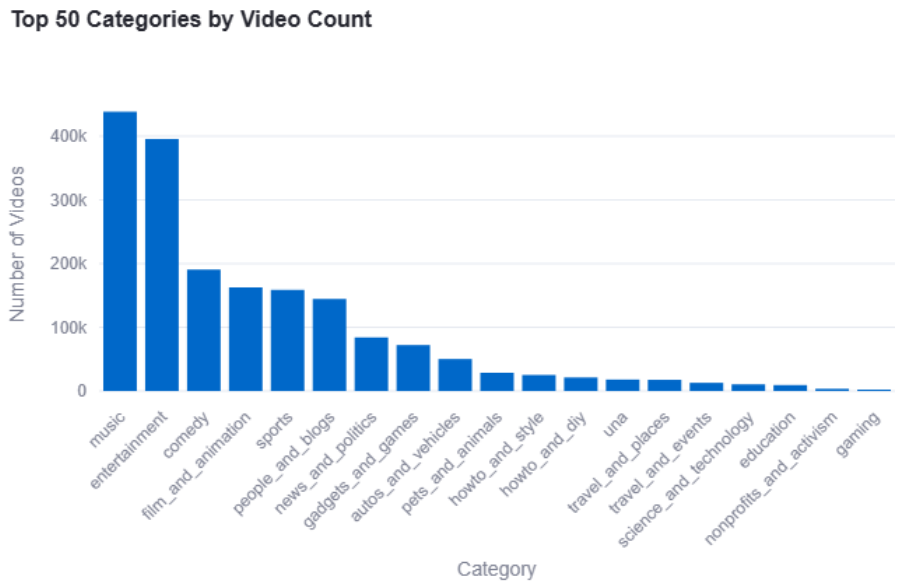
Performance

In the app, we have a performance metric of runtime execution, which is shown in the screenshots of the User Interface. We tested the app with Spark's local and standalone mode, and found the algorithms perform at a similar level, as expected when running on a single machine's resources.

Sample Results

Some insights that we extracted from the analytic results are (keep in mind this is limited to the small sample of the dataset we used):

- Music and entertainment are top categories.



- GRANHERMANO8 has most videos – Spanish Reality TV show, followed by CBS
- Education was surprisingly one of the top categories, even though online educational videos were likely not that popular in 2007-2008
- Highest viewed video is over 79 million views (Evolution of Dance), compared to current day (December 2025) data of 16 Billion views. https://en.wikipedia.org/wiki/List_of_most-viewed_YouTube_videos

Accuracy, Reliability and Scalability

Accuracy

The system's analytics are based on deterministic operations (group-by, counts, filters, and sorts) over the MongoDB video and edge collections using PySpark DataFrames. Algorithm results therefore exactly reflect the stored data, assuming the underlying collections are correctly populated. Video statistics are limited by the crawl process, which only captures a snapshot at a single point in time. As a result, view counts reflect the most recent crawl rather than real-time data, meaning many videos have outdated information.

Reliability

MongoDB and Spark are accessed through dedicated connectors that lazily initialize clients, perform simple health checks (e.g., ping and small test DataFrames), and expose structured status information. Errors during connection or query execution are logged and surfaced to the Streamlit UI via user-friendly warnings rather than raw failures. However, there is no mechanism for the Spark algorithms to retry in case of failure.

Scalability

The architecture separates storage (MongoDB), computation (PySpark), and presentation (Streamlit). Heavy analytics—such as degree distributions and Top-K queries—are implemented as Spark DataFrame transformations that can run on a local or standalone cluster with adaptive query execution, tuned parallelism, and compressed shuffles. Mongo–Spark connector options enable partitioned reads over large collections, and Streamlit's caching avoids recomputing identical queries. Unfortunately, the app's Scalability is lacking without MongoDB sharding and replica sets.

Other Insights

The dataset we worked with was old (2007-2008), just a few years after YouTube's founding (2005). Many videos are unavailable – they either violated terms or were taken down by the uploader. It would've been more beneficial to do our own crawling with current data; however, we had limited time to develop our system. Using the available dataset was a good compromise.

Exploring these older videos provided an interesting glimpse into early YouTube content.

We discovered a video category called ‘una’ from the crawl dataset. We were unsure of what this category represented, but assumed it meant unavailable or ‘no category’.

Conclusion

Overall, the project was a challenging but rewarding experience. Each of us gained hands-on experience with valuable tools like Spark, MongoDB, and Streamlit. There is room for improvement, but overall, we are satisfied with the product we were able to deliver at this point.

Lessons Learned

We created too many indexes on our data. At two crawls, our indexes took up around three-quarters of the size of data in MongoDB. Reworking the parser script would be beneficial so that we could add more data from other crawls without the database size exploding.

Getting Spark to work on Windows was more complicated compared to Linux.

Future Improvements

The next steps for our application include:

- Improve results saving and exporting system so users can save the analytic results
- Add pre-computing/caching for algorithms and job queue for Spark, so users do not need to wait for the algorithm to run as the dataset growth bigger.
- Cloud deployment
- Spark Docker container instances, distributed across different machines

References

Nima, P. (2019). *Exploration of Youtube Statistics Data using Hadoop Technologies*. https://www.researchgate.net/publication/334549087_Exploration_of_Youtube_Statistics_Data_using_Hadoop_Technologies

Poornima, S., Shashidhar, K., & Fernandes, R. (2025). Real Time Sentiment Analysis of YouTube Live Chat With Apache Spark and Kafka. *IEEE Access*, *13*, 188929–188938. https://doi.org/10.1109/ACCESS.2025.3625785

Shaikh, F., Pawaskar, D., Siddiqui, A., & Khan, U. (2018). YouTube Data Analysis using MapReduce on Hadoop. *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, 2037–2041. https://doi.org/10.1109/RTEICT42901.2018.9012635

Appendix

GitHub Repo link: https://github.com/rk3026/Youtube_Analyzer

Data Processing Log

```
2025-10-12 23:19:07,183 - INFO - [CONNECT] Connected to MongoDB:
mongodb://localhost:27017/

2025-10-12 23:19:07,183 - INFO - [MODE] Incremental processing mode - will skip
already processed crawls

2025-10-12 23:19:07,183 - INFO - [EXTRACT] Extracting crawl files...

2025-10-12 23:19:07,184 - INFO -   Extracting 0222.zip...

2025-10-12 23:19:08,099 - INFO -     Extracted 6 files to data\extracted\0222

2025-10-12 23:19:08,100 - INFO -   Extracting 0301.zip...

2025-10-12 23:19:08,314 - INFO -     Extracted 5 files to data\extracted\0301

2025-10-12 23:19:08,314 - INFO - [SUCCESS] Extracted 2 crawl directories

2025-10-12 23:19:08,319 - INFO - [INITIAL] No existing crawls found - processing
all crawls

2025-10-12 23:19:08,319 - INFO - [INDEX] Ensuring database indexes exist...

2025-10-12 23:19:08,819 - INFO - [SUCCESS] Database indexes ensured

2025-10-12 23:19:08,819 - INFO - [PROCESS] Processing crawl: 0222

2025-10-12 23:19:08,820 - INFO -   [LOG] Parsing log: log.txt

2025-10-12 23:19:08,832 - INFO -     Crawl ID: 0222

2025-10-12 23:19:08,832 - INFO -     Start: 0222 16:54:40

2025-10-12 23:19:08,832 - INFO -     Finish: 0227 17:04:29

2025-10-12 23:19:08,836 - INFO -     Processing 0.txt...

2025-10-12 23:19:08,859 - INFO -       Completed 0.txt: 189/189 valid (100.0%)

2025-10-12 23:19:08,859 - INFO -     Processing 1.txt...

2025-10-12 23:19:10,211 - INFO -       Completed 1.txt: 3,141/3,169 valid (99.1%)

2025-10-12 23:19:10,211 - INFO -     Processing 2.txt...

2025-10-12 23:19:13,061 - INFO -       Processed 10,000 records...

2025-10-12 23:19:17,044 - INFO -       Processed 20,000 records...
```

```
2025-10-12 23:19:19,623 - INFO -       Completed 2.txt: 23,454/23,543 valid
(99.6%)
2025-10-12 23:19:19,623 - INFO -     Processing 3.txt...
2025-10-12 23:19:21,195 - INFO -       Processed 30,000 records...
2025-10-12 23:19:25,494 - INFO -       Processed 40,000 records...
2025-10-12 23:19:29,792 - INFO -       Processed 50,000 records...
2025-10-12 23:19:34,725 - INFO -       Processed 60,000 records...
2025-10-12 23:19:39,512 - INFO -       Processed 70,000 records...
2025-10-12 23:19:42,441 - INFO -       Progress: 50,000 lines, 49,653 valid
(99.3%)
2025-10-12 23:19:44,367 - INFO -       Processed 80,000 records...
2025-10-12 23:19:49,029 - INFO -       Processed 90,000 records...
2025-10-12 23:19:53,575 - INFO -       Processed 100,000 records...
2025-10-12 23:19:58,346 - INFO -       Processed 110,000 records...
2025-10-12 23:20:03,209 - INFO -       Processed 120,000 records...
2025-10-12 23:20:05,651 - INFO -       Progress: 100,000 lines, 98,998 valid
(99.0%)
2025-10-12 23:20:08,086 - INFO -       Processed 130,000 records...
2025-10-12 23:20:13,103 - INFO -       Processed 140,000 records...
2025-10-12 23:20:18,454 - INFO -       Processed 150,000 records...
2025-10-12 23:20:23,436 - INFO -       Processed 160,000 records...
2025-10-12 23:20:28,450 - INFO -       Processed 170,000 records...
2025-10-12 23:20:31,770 - INFO -       Progress: 150,000 lines, 148,762 valid
(99.2%)
2025-10-12 23:20:34,623 - INFO -       Processed 180,000 records...
2025-10-12 23:20:40,174 - INFO -       Processed 190,000 records...
2025-10-12 23:20:45,630 - INFO -       Processed 200,000 records...
2025-10-12 23:20:51,009 - INFO -       Processed 210,000 records...
2025-10-12 23:20:56,570 - INFO -       Processed 220,000 records...
2025-10-12 23:20:59,742 - INFO -       Progress: 200,000 lines, 198,497 valid
(99.2%)
```

```
2025-10-12 23:21:02,323 - INFO -       Processed 230,000 records...
2025-10-12 23:21:03,939 - INFO -       Completed 3.txt: 206,784/208,331 valid
(99.3%)
2025-10-12 23:21:03,940 - INFO -       Processing 4.txt...
2025-10-12 23:21:07,912 - INFO -       Processed 240,000 records...
2025-10-12 23:21:13,322 - INFO -       Processed 250,000 records...
2025-10-12 23:21:19,555 - INFO -       Processed 260,000 records...
2025-10-12 23:21:25,429 - INFO -       Processed 270,000 records...
2025-10-12 23:21:31,008 - INFO -       Processed 280,000 records...
2025-10-12 23:21:32,632 - INFO -       Progress: 50,000 lines, 49,753 valid
(99.5%)
2025-10-12 23:21:37,056 - INFO -       Processed 290,000 records...
2025-10-12 23:21:42,859 - INFO -       Processed 300,000 records...
2025-10-12 23:21:48,443 - INFO -       Processed 310,000 records...
2025-10-12 23:21:54,203 - INFO -       Processed 320,000 records...
2025-10-12 23:22:00,188 - INFO -       Processed 330,000 records...
2025-10-12 23:22:01,306 - INFO -       Progress: 100,000 lines, 99,376 valid
(99.4%)
2025-10-12 23:22:06,013 - INFO -       Processed 340,000 records...
2025-10-12 23:22:24,106 - INFO -       Processed 350,000 records...
2025-10-12 23:22:50,630 - INFO -       Processed 360,000 records...
2025-10-12 23:23:04,334 - INFO -       Processed 370,000 records...
2025-10-12 23:23:13,216 - INFO -       Processed 380,000 records...
2025-10-12 23:23:14,342 - INFO -       Progress: 150,000 lines, 148,968 valid
(99.3%)
2025-10-12 23:23:31,722 - INFO -       Processed 390,000 records...
2025-10-12 23:23:38,344 - INFO -       Processed 400,000 records...
2025-10-12 23:23:47,466 - INFO -       Processed 410,000 records...
2025-10-12 23:23:55,489 - INFO -       Processed 420,000 records...
2025-10-12 23:24:29,944 - INFO -       Processed 430,000 records...
```

```
2025-10-12 23:24:34,791 - INFO -       Progress: 200,000 lines, 198,688 valid
(99.3%)
2025-10-12 23:24:40,276 - INFO -       Processed 440,000 records...
2025-10-12 23:24:49,098 - INFO -       Processed 450,000 records...
2025-10-12 23:25:01,711 - INFO -       Processed 460,000 records...
2025-10-12 23:25:31,491 - INFO -       Processed 470,000 records...
2025-10-12 23:25:42,500 - INFO -       Processed 480,000 records...
2025-10-12 23:25:43,561 - INFO -       Progress: 250,000 lines, 248,366 valid
(99.3%)
2025-10-12 23:25:50,265 - INFO -       Processed 490,000 records...
2025-10-12 23:26:11,714 - INFO -       Processed 500,000 records...
2025-10-12 23:26:46,340 - INFO -       Processed 510,000 records...
2025-10-12 23:26:54,751 - INFO -       Processed 520,000 records...
2025-10-12 23:27:02,891 - INFO -       Processed 530,000 records...
2025-10-12 23:27:03,713 - INFO -       Progress: 300,000 lines, 298,009 valid
(99.3%)
2025-10-12 23:27:13,579 - INFO -       Processed 540,000 records...
2025-10-12 23:28:10,941 - INFO -       Processed 550,000 records...
2025-10-12 23:28:57,533 - INFO -       Processed 560,000 records...
2025-10-12 23:29:31,562 - INFO -       Processed 570,000 records...
2025-10-12 23:30:22,728 - INFO -       Processed 580,000 records...
2025-10-12 23:30:27,682 - INFO -       Progress: 350,000 lines, 347,628 valid
(99.3%)
2025-10-12 23:31:37,335 - INFO -       Processed 590,000 records...
2025-10-12 23:32:03,843 - INFO -       Processed 600,000 records...
2025-10-12 23:32:32,004 - INFO -       Processed 610,000 records...
2025-10-12 23:33:09,297 - INFO -       Processed 620,000 records...
2025-10-12 23:33:27,460 - INFO -       Processed 630,000 records...
2025-10-12 23:33:28,734 - INFO -       Progress: 400,000 lines, 397,079 valid
(99.3%)
2025-10-12 23:33:52,417 - INFO -       Processed 640,000 records...
```

```
2025-10-12 23:34:34,535 - INFO -       Processed 650,000 records...
2025-10-12 23:34:59,146 - INFO -       Processed 660,000 records...
2025-10-12 23:35:28,632 - INFO -       Processed 670,000 records...
2025-10-12 23:35:54,545 - INFO -       Processed 680,000 records...
2025-10-12 23:35:54,574 - INFO -       Progress: 450,000 lines, 446,447 valid
(99.2%)
2025-10-12 23:36:31,600 - INFO -       Processed 690,000 records...
2025-10-12 23:36:47,474 - INFO -       Processed 700,000 records...
2025-10-12 23:37:02,125 - INFO -       Processed 710,000 records...
2025-10-12 23:37:10,228 - INFO -       Processed 720,000 records...
2025-10-12 23:37:18,001 - INFO -       Progress: 500,000 lines, 495,951 valid
(99.2%)
2025-10-12 23:37:19,121 - INFO -       Processed 730,000 records...
2025-10-12 23:37:28,652 - INFO -       Processed 740,000 records...
2025-10-12 23:37:38,164 - INFO -       Completed 4.txt: 509,894/514,129 valid
(99.2%)
2025-10-12 23:37:38,164 - INFO -    📊 Crawl 0222 summary:
2025-10-12 23:37:38,164 - INFO -     Total lines: 749,361
2025-10-12 23:37:38,164 - INFO -     Valid records: 743,462
2025-10-12 23:37:38,165 - INFO -     Malformed removed: 5,899
2025-10-12 23:37:38,165 - INFO -     Duplicates removed: 0
2025-10-12 23:37:38,165 - INFO - [SUCCESS] Completed crawl: 0222
2025-10-12 23:37:38,165 - INFO - [PROCESS] Processing crawl: 0301
2025-10-12 23:37:38,166 - INFO -   [LOG] Parsing log: log.txt
2025-10-12 23:37:38,414 - INFO -     Crawl ID: 0301
2025-10-12 23:37:38,414 - INFO -     Start: 0228 23:53:41
2025-10-12 23:37:38,415 - INFO -     Finish: 0302 1:49:33
2025-10-12 23:37:38,417 - INFO -     📄 Processing 0.txt...
2025-10-12 23:37:38,457 - INFO -       Completed 0.txt: 353/359 valid (98.3%)
2025-10-12 23:37:38,457 - INFO -     📄 Processing 1.txt...
```

```
2025-10-12 23:37:41,311 - INFO -       Completed 1.txt: 3,614/3,636 valid (99.4%)
2025-10-12 23:37:41,311 - INFO -     📄 Processing 2.txt...
2025-10-12 23:37:45,937 - INFO -       Processed 10,000 records...
2025-10-12 23:37:54,045 - INFO -       Processed 20,000 records...
2025-10-12 23:37:57,393 - INFO -       Completed 2.txt: 19,695/19,817 valid
(99.4%)
2025-10-12 23:37:57,394 - INFO -     📄 Processing 3.txt...
2025-10-12 23:38:02,260 - INFO -       Processed 30,000 records...
2025-10-12 23:38:17,248 - INFO -       Processed 40,000 records...
2025-10-12 23:38:24,839 - INFO -       Processed 50,000 records...
2025-10-12 23:38:32,780 - INFO -       Processed 60,000 records...
2025-10-12 23:38:40,565 - INFO -       Processed 70,000 records...
2025-10-12 23:38:43,075 - INFO -       Progress: 50,000 lines, 49,851 valid
(99.7%)
2025-10-12 23:38:56,965 - INFO -       Processed 80,000 records...
2025-10-12 23:39:05,580 - INFO -       Processed 90,000 records...
2025-10-12 23:39:15,331 - INFO -       Processed 100,000 records...
2025-10-12 23:39:25,152 - INFO -       Processed 110,000 records...
2025-10-12 23:39:59,373 - INFO -       Processed 120,000 records...
2025-10-12 23:40:08,650 - INFO -       Progress: 100,000 lines, 99,592 valid
(99.6%)
2025-10-12 23:40:14,833 - INFO -       Processed 130,000 records...
2025-10-12 23:40:24,190 - INFO -       Processed 140,000 records...
2025-10-12 23:40:45,828 - INFO -       Processed 150,000 records...
2025-10-12 23:40:59,797 - INFO -       Completed 3.txt: 131,130/131,701 valid
(99.6%)
2025-10-12 23:40:59,797 - INFO -   📊 Crawl 0301 summary:
2025-10-12 23:40:59,797 - INFO -     Total lines: 155,513
2025-10-12 23:40:59,797 - INFO -     Valid records: 154,792
2025-10-12 23:40:59,797 - INFO -     Malformed removed: 721
```

```
2025-10-12 23:40:59,797 - INFO -     Duplicates removed: 0

2025-10-12 23:40:59,798 - INFO - [SUCCESS] Completed crawl: 0301

2025-10-12 23:41:00,002 - INFO - [CLEANUP] Cleaned up extracted files

2025-10-12 23:41:00,005 - INFO - [COMPLETE] Crawl data processing completed
successfully!

2025-10-12 23:41:00,005 - INFO - Processing time: 1312.8 seconds

2025-10-12 23:41:00,005 - INFO - Records processed: 904,874

2025-10-12 23:41:00,005 - INFO - Valid records: 898,254

2025-10-12 23:41:00,006 - INFO - Crawls processed: 2

2025-10-12 23:41:00,006 - INFO - Run validate_crawl_processing.py for detailed
validation and reporting
```

Algorithm Details

Algorithm 1: Network Aggregation / Degree Distribution Computation

Input:

- edges collection from MongoDB containing directed edge relationships (src, dst)
- videos collection containing vertex metadata (_id, category, length_sec, etc.)

Output:

- Per-vertex degree statistics: (video_id, in_degree, out_degree, total_degree)
- Aggregate network statistics: average, minimum, and maximum degrees
- Degree distribution histogram

Computing Operations:

1. **Load edges** from MongoDB into Spark DataFrame
2. **Compute out-degrees**: Group edges by src, count occurrences per vertex
3. **Compute in-degrees**: Group edges by dst, count occurrences per vertex
4. **Join and combine**: Outer join in-degrees and out-degrees on video_id, fill nulls with 0
5. **Calculate total degree**: total_degree = in_degree + out_degree
6. **Aggregate statistics**: Compute avg(), min(), max() over all degrees
7. **Generate histogram**: Group by total_degree, count frequency of each degree value

---

Algorithm 2: Categorized Video Statistics

Input:

- videos collection from MongoDB containing video attributes (category, length_sec, view_count)

Output:

- Frequency distributions partitioned by:
  - Category (e.g., music, entertainment, education)
  - Length buckets (0-2m, 2-5m, 5-10m, 10-20m, >20m)
  - View count buckets (0-1K, 1K-10K, 10K-100K, 100K-1M, 1M-10M, >10M)

Computing Operations:

1. **Load videos** from MongoDB into Spark DataFrame
2. **Category frequency**: Group by category, count videos per category, sort descending
3. **Length bucketing**: Apply conditional logic to bin length_sec into ranges, group and count
4. **View count bucketing**: Apply conditional logic to bin view_count into ranges, group and count

5. **Return results** as separate DataFrames for each partitioning dimension

---

Algorithm 3a: Top-K Categories by Number of Videos

Input

- videos_df: Spark DataFrame containing metadata for each video.
  Relevant attributes:
  - category (string)

Output

- A ranked list (size *K*) of categories with the highest number of videos.
- Columns: category, count

Computing Operations

1. **Group** all videos by the category attribute.
2. **Aggregate** using count() to compute the number of videos per category.
3. **Sort** the aggregated result in descending order of the count.
4. **Return** the top *K* categories.

Algorithm 3b: Top-K Videos by Views (Snapshot-Based)

Input

- snapshots_df: Spark DataFrame containing time-stamped snapshots for each video.
  Relevant attributes:
  - video_id
  - views
  - category
  - crawl_id

Output

- A ranked list (size *K*) of videos with the highest number of views across all snapshots.
- Columns: video_id, views, category, crawl_id

Computing Operations

1. **Sort** the snapshot records by the views attribute in descending order.
2. **Select** relevant fields describing the video and snapshot.
3. **Return** the top *K* records.

Algorithm 3c: Top-K Videos by Rating (Snapshot-Based)

Input

- snapshots_df
  Relevant attributes:
  - video_id
  - rate
  - views
  - category
  - crawl_id

Output

- A ranked list (size *K*) of videos with the highest rating.
- Columns: video_id, rate, views, category, crawl_id

Computing Operations

1. **Sort** the snapshot records by rate in descending order.
2. **Select** fields showing each video's rating, views, and snapshot source.
3. **Return** the top *K* results.

Algorithm 3d: Top-K Range Query: Music Videos With Duration 200–500 Seconds

Input

- snapshots_df
  Relevant attributes:
  - video_id
  - category
  - length_sec
  - views

Output

- A list (size *K*) of music videos whose duration falls between 200 and 500 seconds, ordered by view count.
- Columns: video_id, length_sec, views, category

Computing Operations

1. **Filter** all snapshot records to include only:
   - category == 'Music'
   - length_sec between 200 and 500
2. **Sort** the filtered results by views in descending order.
3. **Select** the relevant columns for reporting.
4. **Return** the top *K* matching videos.

Algorithm 4: Subgraph Pattern Search (Motifs)

Input

- graph: Spark GraphFrame representing the video network.
  Relevant attributes:
  - Vertex: id (video ID), other metadata if needed
  - Edge: relationships between videos (e.g., recommendation, related videos)
- videos_df: Spark DataFrame containing video metadata.
  Relevant attribute: category

Output

- A list of pairs of related videos (edges) that match a subgraph pattern (motif) in the graph.
- Example columns: a.id, b.id, e (edge attributes)

Computing Operations

1. **Define** a subgraph pattern to search, e.g., (a)-[e]->(b), representing a directed edge from video a to video b.
2. **Filter** vertices to only include videos from a specific category (e.g., 'Music').
3. **Execute** the motif search using GraphFrame's find() method, which efficiently searches the graph for all occurrences of the pattern.
4. **Return** the top matches (e.g., first 5) for inspection.

Algorithm 5: Influence Analysis Using PageRank

Input

- graph: Spark GraphFrame representing the video network.
- snapshots_df: Spark DataFrame containing snapshot data.
  Relevant attributes: video_id, views, category, age_days

Output

- A ranked list (size *K*) of videos with the highest influence according to PageRank.
- Columns: video_id, pagerank, views, category

Computing Operations

1. **Compute PageRank** on the video graph using Spark GraphFrames' pageRank() method:
   - resetProbability=0.15 (teleport probability)
   - maxIter=10 (iterations)
2. **Identify the latest snapshot** for each video using a window function over age_days to select the most recent snapshot.
3. **Join** PageRank results with the latest snapshot data to associate each video's influence score with its view count and category.
4. **Sort** the resulting table by pagerank in descending order.
5. **Return** the top *K* videos with the highest influence.

Algorithm Pseudocode

```
Algorithm 1: Network Aggregation / Degree Distribution Computation
FUNCTION compute_degree_distribution(edges, vertices):
    // Step 1: Compute out-degrees
    out_degrees = GROUP edges BY src
                  COUNT(*) AS outDegree

    // Step 2: Compute in-degrees
    in_degrees = GROUP edges BY dst
                 COUNT(*) AS inDegree

    // Step 3: Combine degrees
    degree_stats = OUTER_JOIN(out_degrees, in_degrees) ON id
                   FILL_NULL(outDegree, inDegree) WITH 0
                   ADD_COLUMN total_degree = outDegree + inDegree

    // Step 4: Compute aggregate statistics
    agg_stats = AGGREGATE degree_stats:
                    AVG(total_degree), MIN(total_degree),
MAX(total_degree)
                    AVG(outDegree), MAX(outDegree)
                    AVG(inDegree), MAX(inDegree)

    // Step 5: Generate histogram
    histogram = GROUP degree_stats BY total_degree
                COUNT(*) AS num_videos
                ORDER BY total_degree

    RETURN degree_stats, agg_stats, histogram
END FUNCTION
```

Algorithm 2: Categorized Video Statistics

```
FUNCTION compute_categorized_statistics(vertices):
    // Step 1: Category frequency
    category_freq = GROUP vertices BY category
                    COUNT(*) AS num_videos
                    ORDER BY num_videos DESC

    // Step 2: Length bucketing
    length_buckets = MAP vertices:
                         IF length_sec <= 120 THEN "0-2m"
                         ELSE IF length_sec <= 300 THEN "2-5m"
                         ELSE IF length_sec <= 600 THEN "5-10m"
                         ELSE IF length_sec <= 1200 THEN "10-20m"
                         ELSE ">20m"
                     GROUP BY length_bucket
                     COUNT(*) AS num_videos

    // Step 3: View count bucketing
    view_buckets = MAP vertices:
                       IF view_count <= 1000 THEN "0-1K"
                       ELSE IF view_count <= 10000 THEN "1K-10K"
                       ELSE IF view_count <= 100000 THEN "10K-100K"
                       ELSE IF view_count <= 1000000 THEN "100K-1M"
                       ELSE IF view_count <= 10000000 THEN "1M-10M"
                       ELSE ">10M"
                   GROUP BY view_bucket
                   COUNT(*) AS num_videos

    RETURN category_freq, length_buckets, view_buckets
END FUNCTION
```

```
ALGORITHM 3: Top-K Queries
INPUT: videos_df, snapshots_df, K (number of results)
OUTPUT: Top-K results for various metrics

1. K ← 10

2. // Query 1: Top categories by video count
3. category_counts ← GROUP_BY(videos_df, category)
4. category_counts ← COUNT(category_counts)
5. category_counts ← ORDER_BY(category_counts, count DESC)
6. DISPLAY TOP K(category_counts)

7. // Query 2: Top videos by views
8. top_views ← ORDER_BY(snapshots_df, views DESC)
9. top_views ← SELECT video_id, views, category, crawl_id FROM
top_views
10. DISPLAY TOP K(top_views)

11. // Query 3: Top videos by rating
12. top_ratings ← ORDER_BY(snapshots_df, rate DESC)
13. top_ratings ← SELECT video_id, rate, views, category, crawl_id
FROM top_ratings
14. DISPLAY TOP K(top_ratings)

15. // Query 4: Range query for Music videos
16. music_videos ← FILTER(snapshots_df, category = 'Music' AND
                          length_sec BETWEEN 200 AND 500)
17. music_videos ← SELECT video_id, length_sec, views, category FROM
music_videos
18. music_videos ← ORDER_BY(music_videos, views DESC)
19. DISPLAY TOP K(music_videos)
END ALGORITHM
```

```
ALGORITHM 4: Subgraph Pattern Search (Motif Finding)
INPUT: graph (GraphFrame), videos_df
OUTPUT: Video relationship patterns

1. CREATE_TEMP_VIEW(videos_df, name="videos")
2. // Find motif pattern: (a)-[e]->(b)
3. motifs ← FIND_PATTERN(graph, pattern='(a)-[e]->(b)')
4. // Filter for music category
5. music_filter ← "a.id IN (SELECT _id FROM videos WHERE
category='music')"
6. motifs ← FILTER(motifs, condition=music_filter)
7. DISPLAY LIMIT(motifs, 5)
END ALGORITHM
```

```
ALGORITHM 5: Influence Analysis Using PageRank
INPUT: graph (GraphFrame), snapshots_df, K
OUTPUT: Top-K influential videos

1. // Compute PageRank scores
2. pagerank_results ← PAGERANK(graph, resetProbability=0.15,
maxIter=10)

3. // Get latest snapshot per video
4. windowSpec ← WINDOW(partitionBy=video_id, orderBy=age_days DESC)
5. latest_snapshots ← ADD_ROW_NUMBER(snapshots_df, windowSpec)
6. latest_snapshots ← FILTER(latest_snapshots, row_number = 1)
7. latest_snapshots ← DROP_COLUMN(latest_snapshots, row_number)

8. // Join PageRank with video metadata
9. influence_results ← INNER_JOIN(pagerank_results.vertices,
latest_snapshots, pagerank_results.id = latest_snapshots.video_id)

10. influence_results ← SELECT video_id, pagerank, views, category
                        FROM influence_results

11. influence_results ← ORDER_BY(influence_results, pagerank DESC)

12. DISPLAY TOP K(influence_results)
END ALGORITHM
```

Algorithm Results

Network Aggregation / Degree Distribution

```
============================================================
2025-11-16 22:51:40,324 - INFO - DEGREE DISTRIBUTION STATISTICS
2025-11-16 22:51:40,324 - INFO - ============================================================
==========
2025-11-16 22:51:40,325 - INFO -
[AGGREGATE STATISTICS]
2025-11-16 22:51:40,325 - INFO -   Average Total Degree:    8.02
2025-11-16 22:51:40,326 - INFO -   Minimum Total Degree:    1
2025-11-16 22:51:40,326 - INFO -   Maximum Total Degree:    2224
2025-11-16 22:51:40,327 - INFO -
  Average Out-Degree:      4.03
2025-11-16 22:51:40,327 - INFO -   Maximum Out-Degree:      20
2025-11-16 22:51:40,327 - INFO -
  Average In-Degree:       4.00
2025-11-16 22:51:40,328 - INFO -   Maximum In-Degree:       2204
```

```
[Stage 4:================>(22 + 1) / 23][Stage 5:================>(22 + 1) / 23]
+-------+-----------+------------------+------------------+
|summary|         id|          inDegree|         outDegree|
+-------+-----------+------------------+------------------+
|  count|      22036|             22036|             22036|
|   mean|       NULL|2.2690143401706298|2.2690143401706298|
| stddev|       NULL|3.4225807045249566| 6.328498087331983|
|    min|--aApGlDZGA|                 0|                 0|
|    max|zzuddQOyuW8|                46|                20|
+-------+-----------+------------------+------------------+
```

```
[TOP 10 VIDEOS BY TOTAL DEGREE]
2025-11-16 22:51:41,951 - INFO -   Video ID: 22uYDgZXoJQ
2025-11-16 22:51:41,954 - INFO -     Total Degree: 2224 (In: 2204, Out: 20)
2025-11-16 22:51:41,955 - INFO -   Video ID: H1BYrSk46vg
2025-11-16 22:51:41,955 - INFO -     Total Degree: 2141 (In: 2121, Out: 20)
2025-11-16 22:51:41,956 - INFO -   Video ID: nypYVhnGbcQ
2025-11-16 22:51:41,957 - INFO -     Total Degree: 2139 (In: 2119, Out: 20)
2025-11-16 22:51:41,958 - INFO -   Video ID: 9nbeqmXEbj8
2025-11-16 22:51:41,958 - INFO -     Total Degree: 2124 (In: 2104, Out: 20)
2025-11-16 22:51:41,959 - INFO -   Video ID: oeGjJ-joZdI
2025-11-16 22:51:41,959 - INFO -     Total Degree: 1902 (In: 1882, Out: 20)
2025-11-16 22:51:41,960 - INFO -   Video ID: 5FQN_MYRm4U
2025-11-16 22:51:41,960 - INFO -     Total Degree: 1892 (In: 1872, Out: 20)
2025-11-16 22:51:41,961 - INFO -   Video ID: QLrEcMq_nPQ
2025-11-16 22:51:41,961 - INFO -     Total Degree: 1882 (In: 1862, Out: 20)
2025-11-16 22:51:41,962 - INFO -   Video ID: 13n8OpQS9Dg
2025-11-16 22:51:41,962 - INFO -     Total Degree: 1878 (In: 1858, Out: 20)
2025-11-16 22:51:41,963 - INFO -   Video ID: y2jUOABWK1A
2025-11-16 22:51:41,963 - INFO -     Total Degree: 1873 (In: 1853, Out: 20)
2025-11-16 22:51:41,964 - INFO -   Video ID: JrUupfFzBrA
2025-11-16 22:51:41,964 - INFO -     Total Degree: 1861 (In: 1841, Out: 20)
```

```
[DEGREE DISTRIBUTION HISTOGRAM]
2025-11-16 22:51:42,029 - INFO -   Degree | Number of Videos
2025-11-16 22:51:42,030 - INFO -   ----------------------------
2025-11-16 22:51:56,588 - INFO -        1 | 2,001,992
2025-11-16 22:51:56,588 - INFO -        2 | 654,777
2025-11-16 22:51:56,589 - INFO -        3 | 297,089
2025-11-16 22:51:56,590 - INFO -        4 | 158,974
2025-11-16 22:51:56,591 - INFO -        5 | 95,763
2025-11-16 22:51:56,591 - INFO -        6 | 61,154
2025-11-16 22:51:56,591 - INFO -        7 | 40,932
2025-11-16 22:51:56,592 - INFO -        8 | 28,580
2025-11-16 22:51:56,593 - INFO -        9 | 20,575
2025-11-16 22:51:56,593 - INFO -       10 | 15,337
2025-11-16 22:51:56,594 - INFO -       11 | 11,480
2025-11-16 22:51:56,594 - INFO -       12 | 8,881
2025-11-16 22:51:56,595 - INFO -       13 | 7,020
2025-11-16 22:51:56,595 - INFO -       14 | 5,578
2025-11-16 22:51:56,596 - INFO -       15 | 4,515
2025-11-16 22:51:56,596 - INFO -       16 | 3,810
2025-11-16 22:51:56,597 - INFO -       17 | 3,185
2025-11-16 22:51:56,597 - INFO -       18 | 2,739
2025-11-16 22:51:56,598 - INFO -       19 | 2,673
2025-11-16 22:51:56,598 - INFO -       20 | 5,182
2025-11-16 22:51:56,599 - INFO -
```

Categorized Statistics

```
======================================================================
2025-11-16 22:51:56,731 - INFO - CATEGORIZED VIDEO STATISTICS
2025-11-16 22:51:56,731 - INFO - ============================================================
==========
2025-11-16 22:51:56,732 - INFO -
[BY CATEGORY] Top 15
2025-11-16 22:52:04,462 - INFO -   music: 227,689
2025-11-16 22:52:04,462 - INFO -   entertainment: 167,071
2025-11-16 22:52:04,463 - INFO -   comedy: 112,852
2025-11-16 22:52:04,463 - INFO -   sports: 93,119
2025-11-16 22:52:04,463 - INFO -   film_and_animation: 89,306
2025-11-16 22:52:04,464 - INFO -   gadgets_and_games: 76,361
2025-11-16 22:52:04,464 - INFO -   people_and_blogs: 66,054
2025-11-16 22:52:04,464 - INFO -   news_and_politics: 45,812
2025-11-16 22:52:04,465 - INFO -   howto_and_diy: 22,787
2025-11-16 22:52:04,465 - INFO -   autos_and_vehicles: 21,776
2025-11-16 22:52:04,466 - INFO -   travel_and_places: 18,972
2025-11-16 22:52:04,466 - INFO -   pets_and_animals: 13,900
2025-11-16 22:52:04,467 - INFO -   una: 8,199
2025-11-16 22:52:04,467 - INFO -
[BY LENGTH BUCKET]
2025-11-16 22:52:11,087 - INFO -     0-2m: 328,809
2025-11-16 22:52:11,088 - INFO -   10-20m: 17,420
2025-11-16 22:52:11,089 - INFO -     2-5m: 418,836
2025-11-16 22:52:11,090 - INFO -    5-10m: 193,206
2025-11-16 22:52:11,091 - INFO -     >20m: 5,627
2025-11-16 22:52:11,102 - INFO -
```

Top-K queries

```
+------------------+------+
|          category| count|
+------------------+------+
|             music|212605|
|     entertainment|154820|
|            comedy|105659|
|            sports| 85578|
|film_and_animation| 85185|
| gadgets_and_games| 72083|
|  people_and_blogs| 60546|
| news_and_politics| 42741|
|     howto_and_diy| 21282|
|autos_and_vehicles| 19833|
+------------------+------+
only showing top 10 rows


+-----------+--------+---------------+--------+
|   video_id|   views|       category|crawl_id|
+-----------+--------+---------------+--------+
|dMH0bHeiRNg|42513417|         comedy|    0222|
|4c_Grdrx7t0|24133454|            una|    0301|
|0XxI-hvPRRA|20282464|         comedy|    0222|
|1dmVU08zVpA|16087899|  entertainment|    0222|
|RB-wUgnyGv0|15712924|  entertainment|    0222|
|QjA5faZF1A8|15256922|          music|    0222|
|-_CSo1gOd48|13199833|people_and_blogs|    0222|
|49IDp76kjPw|11970018|         comedy|    0222|
|tYnn51C3X_w|11823701|          music|    0222|
|pv5zWaTEVkI|11672017|          music|    0222|
+-----------+--------+---------------+--------+
only showing top 10 rows
```

```
[Stage 39:=====================================>                   (2 + 1) / 3]
+-----------+----+-----+-----------------+--------+
|   video_id|rate|views|         category|crawl_id|
+-----------+----+-----+-----------------+--------+
|feioiPwxt98| 5.0| 1632|           sports|    0301|
|LqNUMgSzWP4| 5.0|  167|            music|    0222|
|--B1obbH1ck| 5.0|  120|            music|    0222|
|Lq_pEG5dYBo| 5.0|  558|           comedy|    0222|
|fenxoc-IWb0| 5.0| 1590|autos_and_vehicles|    0222|
|LqPW591Xfaw| 5.0| 1583|    entertainment|    0222|
|--Qm8HF07BM| 5.0|  214|            music|    0222|
|LqGGf5Go3Jk| 5.0|  362|            music|    0222|
|fekHbMHXXcQ| 5.0|  101|film_and_animation|    0222|
|LqTK-3wsH1E| 5.0| 7735|    entertainment|    0222|
+-----------+----+-----+-----------------+--------+
only showing top 10 rows

+--------+----------+-----+--------+
|video_id|length_sec|views|category|
+--------+----------+-----+--------+
+--------+----------+-----+--------+
```

Subgraph pattern search (motifs)

```
+--------------+--------------------+--------------+
|             a|                   e|             b|
+--------------+--------------------+--------------+
|{-boOvAGNKUc}|{-boOvAGNKUc, sf-...|{sf-Ym_pFP6U}|
|{-boOvAGNKUc}|{-boOvAGNKUc, OUe...|{OUeN4DhCIFw}|
|{-boOvAGNKUc}|{-boOvAGNKUc, Jsd...|{JsdCu9T47iY}|
|{-boOvAGNKUc}|{-boOvAGNKUc, 7wj...|{7wj8-HkZ0XQ}|
|{-boOvAGNKUc}|{-boOvAGNKUc, MG1...|{MG1Xv99426g}|
+--------------+--------------------+--------------+
only showing top 5 rows
```

Influence Analysis (Pagerank)

```
25/11/16 00:04:29 WARN BlockManager: Block rdd_312_1 already exists on this machine; not re-adding it
[Stage 389:=====================================>                 (2 + 1) / 3]
+-----------+-----------------+-----+----------------+
|   video_id|         pagerank|views|        category|
+-----------+-----------------+-----+----------------+
|-fTO_SYoFCM|6.246195832075682|   64| people_and_blogs|
|FN9ZOOImjCg|6.246195832075682|  182| people_and_blogs|
|KNdj3ae2Tlk|6.246195832075682|  164| people_and_blogs|
|pazSPUYZYVE|6.101569604111737|  350|news_and_politics|
|TFvum08n0nI|6.101569604111737| 1391|   entertainment|
|Ud7KYmEOWic|6.101569604111737|  232|news_and_politics|
|6OZ0ruq0DH4|6.101569604111737|  837|   entertainment|
|CESHloI5or8|6.101569604111737|  552|news_and_politics|
|QezUyZ7pKQc|6.101569604111737| 8635|          sports|
|D7k8Ni_oEsE|6.101569604111737|  519|          sports|
+-----------+-----------------+-----+----------------+
only showing top 10 rows
```