# Big Data Report

# Team F3

**Yizhi Gu (466768)**

**Moksh Singh (469249)**

**Yiqing Zhang (474084)**

**Christopher Naegele (468960)**

## Introduction

Youtube is one of the most popular video-sharing platforms, which allows users to upload, view, rate, share, comment on videos, and subscribe to the channels they like. It offers a wide variety of user-generated and corporate media videos. As of May 2019, more than 500 hours of video content are uploaded to YouTube every minute. Therefore, we are curious about what are the most popular videos and channels. Is there any relationship between different attributes of Youtube?

We applied our analytic skills on the dataset of Youtube Statistics, which has been extracted from Kaggle. In this report, we use several Hadoop tools, including impala, Pyspark, and we also use MapReduce to complete the word-counting of some attributes in this dataset. Additionally, to present our results, we use multiple tools, such as tableau and python, to visualize the exciting results we get.

This report will include five parts, introducing our problem statement, dataset we use, what methods and results we get, and the summary of the whole report. Besides, we also add our code at last.

## DataSet

The DataSet here comprises Youtube video related data from 10 countries - Canada, France, Great Britain, India, Japan, South Korea, Russia, the US, Germany, and Mexico. Each table has 17 columns from Likes, Dislikes, ID, Trending Date, Views, Comment Count to Description. The data is comprised of the top 200 trending videos in a particular region for each day and is structured.

In order for the Youtube dataset to be considered as a big data dataset. The Youtube dataset has to fulfill the 3Vs requirement, which is volume, variety, and velocity. As for volume, the aggregated Youtube dataset has data from 10 countries with 2 million rows of data for each country, and a total of more than 20 Million records with a total of 515 MB. As for variety, the aggregated Youtube data set has 16 columns of different data with six unique types of data recorded. For velocity, The Youtube dataset records new publishing video data from every second. Thereby, the Youtube dataset is considered to be big data that fulfills the 3Vs.

**Problem Statement**

With a bird's eye view of the whole dataset, we are trying to find some patterns and trends among various attributes of the Youtube containing the entire world data. We are curious about what are the most popular videos and channels, and what is the relationship between different attributes. Additionally, to dig more in-depth about the data, we compare the characteristics between different countries and find out what is common and what is different.

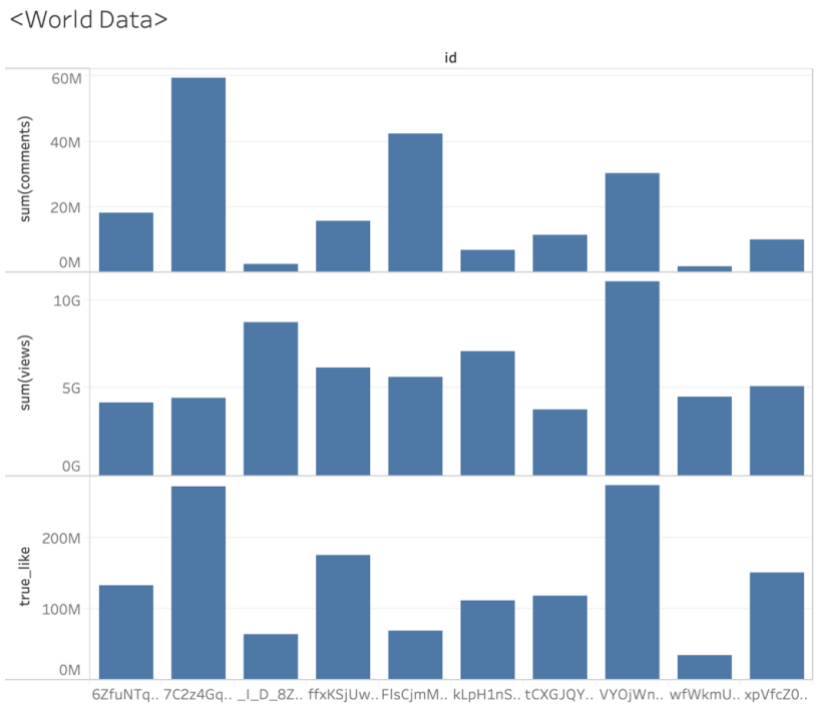
While researching the patterns and trends, we are also interested to see if some "urban Myths" about YouTube is correct or not. These myths include: everyone likes "crazy cat" video on YouTube; The more extended the video title is, the more views it got; and people want to "Eat and Run," which means they watch video without like, dislike, or comment.

**Methods and Results**

Since we have 10 datasets from 10 different countries, the first step is to aggregate all the data into one. The easiest way to do aggregation is to use bash code. By using "sed 1d" and "cat" command,

we got rid of the title row from 9 out of 10 datasets and converted all datasets into one aggregated data file called “youtube.csv”. Then, we used Hadoop and Python to do Big Data Analytics on the Data. The data was in CSV format so we loaded it into our Hadoop environment, formatting the delimited fields by the delimiter. We created a table with our required attributes of views, likes, dislikes, and comments. As the graph shows below:

| | id | sum(views) | sum(likes) | sum(dislikes) | sum(comments) | true_like |
|---|---|---|---|---|---|---|
| 1 | VYOjWnS4cMY | 11014979242 | 290396935 | 17571281 | 30373217 | 272825654 |
| 2 | _I_D_8Z4sJE | 8732892233 | 66141556 | 3105679 | 2488744 | 63035877 |
| 3 | kLpH1nSLJSs | 7048799863 | 116054986 | 5540628 | 6751136 | 110514358 |
| 4 | ffxKSjUwKdU | 6153508814 | 182749328 | 6941462 | 15587662 | 175807866 |
| 5 | FlsCjmMhFmw | 5606930213 | 141474753 | 71923276 | 42450281 | 69551477 |
| 6 | xpVfcZ0ZcFM | 5097369107 | 152643873 | 2709607 | 9899016 | 149934266 |
| 7 | wfWkmURBNv8 | 4495150978 | 38175496 | 3062839 | 1725929 | 35112657 |
| 8 | 7C2z4GqqS5E | 4449219508 | 279897557 | 8528976 | 59289892 | 271368581 |
| 9 | 6ZfuNTqbHE8 | 4185267042 | 135529203 | 2507805 | 18317019 | 133021398 |
| 10 | tCXGJQYZ9JA | 3759256931 | 125175069 | 6979629 | 11287659 | 118195440 |

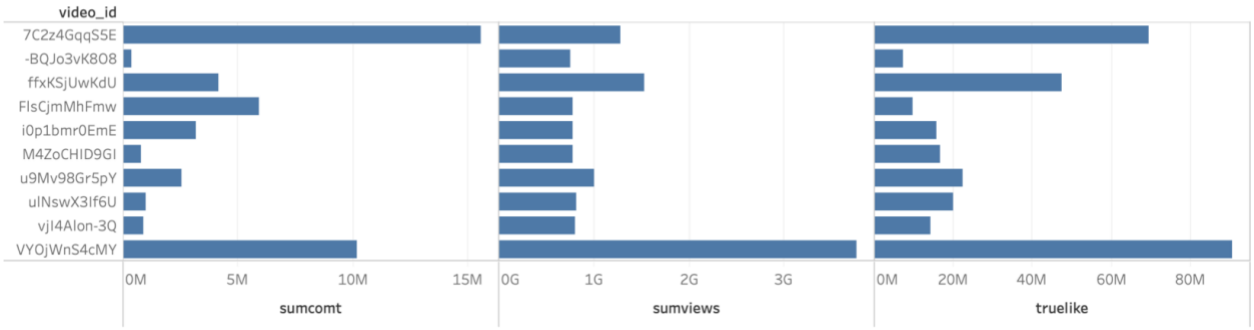



(Attributes from 10 countries)

Query History  Saved Queries  Results (10)

| | video_id | sumviews | sum(likes) | sum(dislikes) | truelike | sumcomt |
|---|---|---|---|---|---|---|
| 1 | VYOjWnS4cMY | 3758488765 | 96700818 | 6054434 | 90646384 | 10151289 |
| 2 | ffxKSjUwKdU | 1529291326 | 49451353 | 1877988 | 47573365 | 4143608 |
| 3 | 7C2z4GqqS5E | 1283188291 | 71835050 | 2389776 | 69445274 | 15568561 |
| 4 | u9Mv98Gr5pY | 1003151226 | 23339807 | 831784 | 22508023 | 2533352 |
| 5 | ulNswX3lf6U | 818792483 | 20165850 | 346079 | 19819771 | 1002366 |
| 6 | vjI4Alon-3Q | 803455479 | 15314079 | 1065209 | 14248870 | 864473 |
| 7 | FlsCjmMhFmw | 780801040 | 19781372 | 10111153 | 9670219 | 5898529 |
| 8 | M4ZoCHID9GI | 778810304 | 16996782 | 369107 | 16627675 | 774444 |
| 9 | i0p1bmr0EmE | 774320575 | 17322894 | 1534546 | 15788348 | 3172060 |
| 10 | -BQJo3vK8O8 | 748060352 | 7964301 | 692739 | 7271562 | 363161 |

<US data >



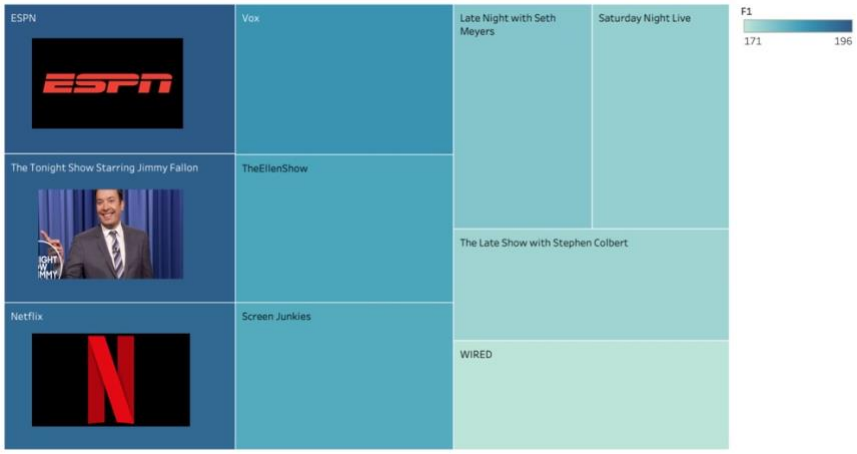

(Attributes from US only)

We tried to find the TOP 10 viewed videos from only US data and aggregated data. By sorting in total views, we found that there are only two videos appear in both lists, one is "7C2z4GqqS5E", the other one is "VYOJWnS4cMY", and it is also the winner of both list. Also, the percentage of viewers that likes, dislikes, and true likes are relatively close on these two videos.

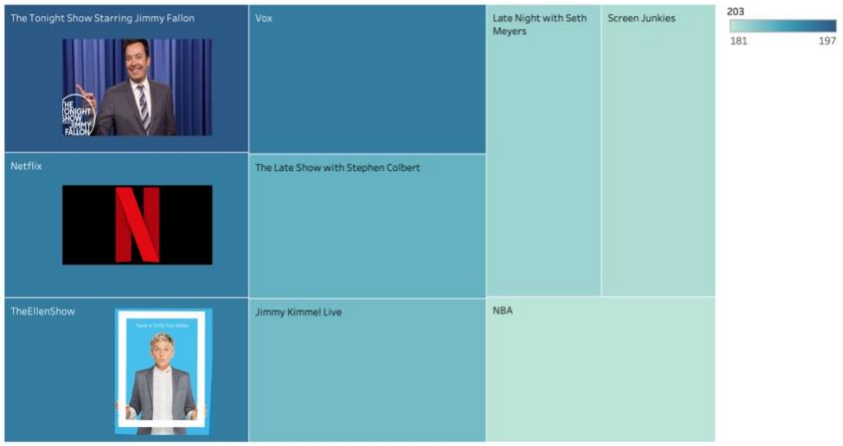
We also use the MapReduce to count the words of channel_title in the Youtube dataset. First, we create two python files, including map.py and reduce.py as the mapper and reducer respectively. Second, we put the original dataset into HDFS and write the bash code to do the MapReduce.

Moreover, we run the MapReduce process both on the whole dataset, which contains the entire world data and the US dataset, in order to compare the difference of the popular channels.

<US Data>




<World Data>



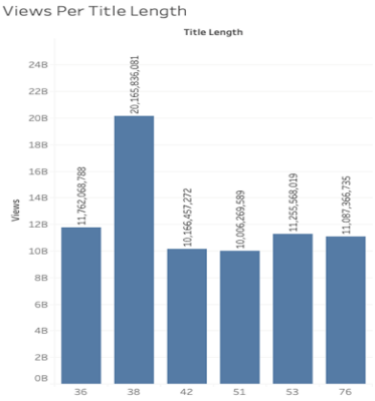

As of the “crazy cat” myth, we tried to find the top 15 tags from the aggregated data to see if there are any cat related tags are in the top 15 list. Since it requires package “nltk”, “stopwords” and “wordcloud” to count the actual tags excluding stopwords. As a result:

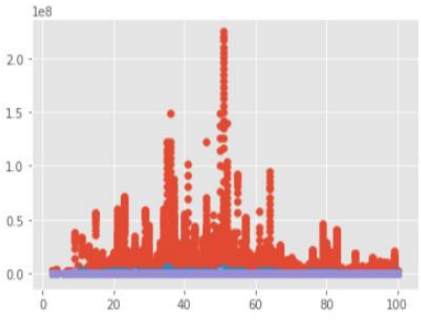
| Tags | Counts |
|---|---|
| "funny" | 3737 |
| "comedy" | 2742 |
| "news" | 1914 |
| "politics" | 1380 |
| "donald trump" | 1211 |
| "video" | 1091 |
| 'entertainment" | 1083 |
| "music" | 1069 |
| "food" | 1053 |
| "humor" | 1033 |
| "trump" | 1031 |
| "2018" | 967 |
| "talk show" | 946 |
| "reaction" | 907 |
| "review" | 876 |


There are not any words in the TOP 15 list that related to cat. The Urban myth of people love to watch “crazy cat” on Youtube is groundless.

The second Urban myth we are interested in is “The longer the video title is, the more views it got”. In order to find the result, we used Impala to find the relation between title length and total views, and we used python and Tableau to create a plot chart as well as the correlation between title length and views, likes, dislikes.

| | title_length | sum(views) |
|---|---|---|
| 1 | 38 | 20165836081 |
| 2 | 36 | 11762068788 |
| 3 | 53 | 11255568019 |
| 4 | 76 | 11087366735 |
| 5 | 42 | 10166457272 |
| 6 | 51 | 10006269589 |




The correlation between title length and views, likes, dislikes are

```
Correlation between TITLE LENGTH and VIEWS
[[ 1.         -0.03560854]
 [-0.03560854  1.        ]]
Correlation between TITLE LENGTH and LIKES
[[ 1.         -0.07816254]
 [-0.07816254  1.        ]]
Correlation between TITLE LENGTH and dislikes
[[ 1.         -0.03123342]
 [-0.03123342  1.        ]]
```

As a result, there are no correlation between title length and views, or dislikes, but a tiny negative correlation between title length and likes.

The third Urban myth is that people like to “Eat and Run”. We used Impala to find the “Eat and Run” ratio, which is the ratio of the summation of total likes, dislikes, comment count and total views. The lower this ratio is, the more people tend to “Eat and Run”.

| | category | eat_and_run_rate |
|---|---|---|
| 1 | 44 | 0.99620296858819468 |
| 2 | 30 | 0.98420054776826349 |
| 3 | 19 | 0.97853995452320608 |
| 4 | 43 | 0.97815700713053066 |
| 5 | 25 | 0.97751978628420066 |

As a result, the average “Eat and Run” ratio is 0.98069, which means 98% of people do not leave any comments or like/dislike after watching a video. The myth of people like to “Eat and Run” is proved to be true, and the category of video that people like to “Eat and Run” the most is the trailers.

**Conclusion**

After doing all the research, we conclude that the US and other countries have some attributes in common, for example, and every country shares similar tastes on categories. Although users from the US have similar tastes on video “VYOJWnS4cMY” as the other countries, users from different countries tend not to share common with the users from the US on any other videos.

As of the Urban Myth, most of the Myth cannot be backed by data, only the Myth of “people like to “Eat and Run” has been proven right with a solid evidence that 98% of people don’t like/dislike or comment after watching videos. It is interesting using big data to research user’s habits and behavior. With the help of using big data, it is easier to understand the user’s habits and behaviors, as well as making business decisions.

**Code**

1.To combine all files into one CSV and put it onto HDFS.

```
[cloudera@quickstart youtube]$ sed 1d CAvideos.csv >nhCAvideos.csv
[cloudera@quickstart youtube]$ sed 1d FRvideos.csv >nhFRvideos.csv
[cloudera@quickstart youtube]$ sed 1d INvideos.csv >nhINvideos.csv
[cloudera@quickstart youtube]$ sed 1d KRvideos.csv >nhKRvideos.csv
[cloudera@quickstart youtube]$ sed 1d RUvideos.csv >nhRUvideos.csv
[cloudera@quickstart youtube]$ sed 1d DEvideos.csv >nhDEvideos.csv
[cloudera@quickstart youtube]$ sed 1d GBvideos.csv >nhGBvideos.csv
[cloudera@quickstart youtube]$ sed 1d JPvideos.csv >nhJPvideos.csv
[cloudera@quickstart youtube]$ sed 1d MXvideos.csv >nhMXvideos.csv
[cloudera@quickstart youtube]$ sed 1d USvideos.csv >nhUSvideos.csv
[cloudera@quickstart youtube]$ cat USvideos.csv nhCAvideos.csv nhFRvideos.csv nh
INvideos.csv nhKRvideos.csv nhRUvideos.csv nhDEvideos.csv nhGBvideos.csv nhJPvid
eos.csv nhMXvideos.csv > youtube.csv

[cloudera@quickstart youtube]$ hdfs dfs -put youtube.csv
[cloudera@quickstart youtube]$ hdfs dfs -ls
Found 1 items
-rw-r--r--   1 cloudera cloudera  540424408 2019-12-06 22:12 youtube.csv
```

2. To make a MapReduce to count words of the channel_title in the dataset.

Mapper:

```
#!/usr/bin/env python
import sys
for line in sys.stdin:
    lines=line.strip().split(",")
    title=lines[3]
    print '%s\t%s' %(title,"1")
```

**Reducer:**

```
#!/usr/bin/env python

import sys

titlesum={}

for line in sys.stdin:

        title,count=line.strip().split("\t")

        try:

                count=int(count)

        except ValueError:

           continue

         try:

                titlesum[title]=titlesum[title]+count

        except:

               titlesum[title]=count

for title in titlesum.keys():

        print "%s\t%s"% (title,titlesum[title])
```

**Bash file:**

(1)

```
#!/bin/bash

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.13.0.jar\

        -input /user/cloudera/USvideos.csv\

        -output /user/cloudera/channel\

        -file map.py\
```

-file reduce.py\

-mapper "python map.py"\

-reducer "python reduce.py"

(2)

#!/bin/bash

hadoop jar /usr/lib/hadoop-mapreduce/hadoop-streaming-2.6.0-cdh5.13.0.jar\

-input /user/cloudera/youtube.csv\

-output /user/cloudera/channel2\

-file map.py\

-file reduce.py\

-mapper "python map.py"\

-reducer "python reduce.py"

Result:

hdfs dfs -cat channel2/part-00000|sort -n|head

```
1	494ta
1	90s Commercials
1	Absolut
1	Aerie
1	Alicia Keys
1	American Bridge 21st Century
1	Anaki Abo
1	BeyondSlowMotion
1	BillyCrammer
1	Bini the Bunny
```

hdfs dfs -cat channel2/part-00000|sort -rn|head

```
203     ESPN
197     The Tonight Show Starring Jimmy Fallon
193     Vox
193     TheEllenShow
193     Netflix
187     The Late Show with Stephen Colbert
186     Jimmy Kimmel Live
183     Late Night with Seth Meyers
182     Screen Junkies
181     NBA
```

3. To import data to impala to make some queries.

First, create the table youtube from the CSV data.

```
CREATE table Youtube(ID STRING,trending string, title string,
channel string, category int, publishDate string, tags string,
views int, likes int, dislikes int, comments int, url string,
CD int, RD int, VE int, description string) Row FORMAT DELIMITED FIELDS TERMINATED BY',' STORED AS TEXTFILE;
LOAD DATA INPATH '/user/cloudera/youtube.csv' into table youtube
```

4. Getting the views list sorted along with ID.

```
select ID, sum(views),sum(likes),sum(dislikes),sum(comments)
from Youtube
where views is not NULL
Group by ID
order by sum(views) desc limit 10
```

Select video_id, sum(views), sum(likes), sum(dislikes), sum(comments)

Form table_name

Where views is not NULL

Group by video_od

Order by sum(views) desc limit 10；

Results of the query：

| | id | sum(views) |
|---|---|---|
| 1 | VYOjWnS4cMY | 11014979242 |
| 2 | _I_D_8Z4sJE | 8732892233 |
| 3 | kLpH1nSLJSs | 7048799863 |
| 4 | ffxKSjUwKdU | 6153508814 |
| 5 | FlsCjmMhFmw | 5606930213 |
| 6 | xpVfcZ0ZcFM | 5097369107 |
| 7 | wfWkmURBNv8 | 4495150978 |
| 8 | 7C2z4GqqS5E | 4449219508 |
| 9 | 6ZfuNTqbHE8 | 4185267042 |
| 10 | tCXGJQYZ9JA | 3759256931 |

5. Getting the Like/Views for the list, ordered by highest like percentage

```
select ID,round(sum(likes)/sum(views),2) as LikePrecentage,sum(dislikes),sum(comments)
from Youtube
where views is not NULL and likes is not NULL
Group by ID
order by likePrecentage desc limit 10
```

Results：

| | id | likeprecentage |
|---|---|---|
| 1 | 5mNSOQO-bRA | 0.93999999999999995 |
| 2 | 18HPwrJuSIM | 0.69999999999999996 |
| 3 | bRnJjFb4D5U | 0.48999999999999999 |
| 4 | nCAKMs5Pjf4 | 0.42999999999999999 |
| 5 | tuvDpOE44ko | 0.41999999999999998 |
| 6 | jp1yOtQN2ZA | 0.41999999999999998 |
| 7 | wKAFBEhErTo | 0.40999999999999998 |
| 8 | BHK9BSr9LEA | 0.40999999999999998 |
| 9 | Ay7RapOcTow | 0.40999999999999998 |
| 10 | ODjB1sL_D2E | 0.39000000000000001 |

6. Category with Like percentage order

```
select category,round(sum(likes)/sum(views),2) as LikePrecentage,sum(dislikes),sum(comments)
from Youtube
where views is not NULL and likes is not NULL
Group by category
order by likePrecentage desc limit 10
```

Result：

| | category | likeprecentage |
|---|---|---|
| | 29 | 0.080000000000000002 |
| 2 | 23 | 0.050000000000000003 |
| 3 | 26 | 0.040000000000000001 |
| 4 | 20 | 0.040000000000000001 |
| 5 | 27 | 0.040000000000000001 |

8. Eat and Run rate

select avg(a.Eat_And_Run_Rate)

from(

select category_id, (1-(sum(dislikes)+sum(likes)+sum(comment_count))/sum(views))as Eat_And_Run_Rate

from youtube

where views is not null and likes is not null and dislikes is not null and comment_count is not null

group by category_id

order by Eat_And_Run_Rate desc limit 5) as a

| | avg(a.eat_and_run_rate) |
|---|---|
| 1 | 0.980695056438049 |

```
select category,1-(sum(dislikes)+sum(likes)+sum(comments))/sum(views) as EAT_AND_RUN_RATE
from Youtube
where views is not NULL and likes is not NULL and dislikes is not NULL and comments is not NULL
Group by category
order by EAT_AND_RUN_RATE desc limit 5
```

Query History | Saved Queries | Results (5)

| | category | eat_and_run_rate |
|---|---|---|
| 1 | 44 | 0.99620296858819468 |
| 2 | 30 | 0.98420054776826349 |
| 3 | 19 | 0.97853995452320608 |
| 4 | 43 | 0.97815700713053066 |
| 5 | 25 | 0.97751978628420066 |

9. Title length of the most viewed videos

```
select length(title) as TITLE_LENGTH, sum(views)
from Youtube
where views is not NULL and likes is not NULL and dislikes is not NULL and comments is not NULL
Group by TITLE_LENGTH
order by sum(views) desc
```

Query History | Saved Queries | Results (290)

| | title_length | sum(views) |
|---|---|---|
| 1 | 38 | 20165836081 |
| 2 | 36 | 11762068788 |
| 3 | 53 | 11255568019 |
| 4 | 76 | 11087366735 |
| 5 | 42 | 10166457272 |
| 6 | 51 | 10006269589 |

**Reference**


1.Loke Hale, James (May 7, 2019). "More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute". TubeFilter. Los Angeles, CA. Retrieved June 10, 2019.

2. Wikipedia: https://en.wikipedia.org/wiki/YouTube