



## Analyzing Social Trends Data from Google Trends and YouTube

4th Aug, 2015

In today's world dominated by technology and gadgets, we often wonder how well a particular product or technology is being perceived by society and if it is truly going to leave a long lasting impression. We regularly see fan wars breaking out between Apple and Android fanatics (no offence to Windows mobile lovers!) on social media where each claim that they are better than the rest. Another thing we notice quite often is that whenever a new product is launched or in the process of being launched, it starts trending on different social media websites.

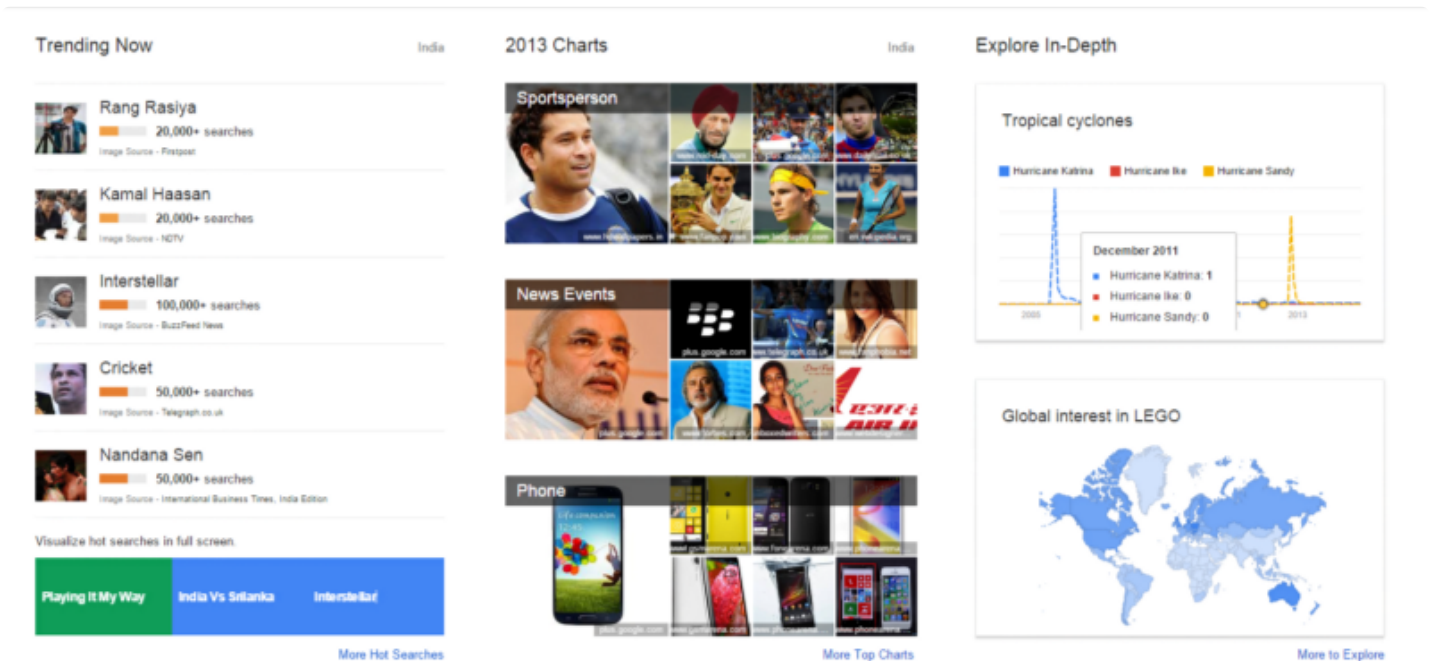
This made me wonder if there was a way to see some of these trends and the impact it is causing on social media. There are different social media channels where people post a wide variety of content ranging from personal opinions to videos and pictures. A few of the popular ones are listed below.

- Facebook
- Twitter
- YouTube
- Instagram
- Pinterest

Today, I will discuss two such ways we can do this, namely **Google Trends** and **YouTube**. If you want to know how we can perform data mining

using Twitter, refer to my earlier post “[Building a Twitter Sentiment Analysis App using R](#)” which deals with getting data from Twitter and analyzing it.

Now, we will be looking at how easy it is to visualize trending topics on Google Trends without writing a single line of code. For this, you need to go to the [Google Trends](#) website. On opening it, you will be greeted by an interactive dashboard, showing the current trending topics summarized briefly just like the snapshot shown below. You can also click on any particular panel to explore it in detail.



This is not all that Google Trends has to offer. We can also customize visualizations to see how specific topics are trending across the internet by specifying them in the interface and the results are shown in the form of a beautiful visualization. Google gets the data based on the number of times people have searched for it online. A typical comparison of people’s interest in different mobile operating systems over time is shown below.

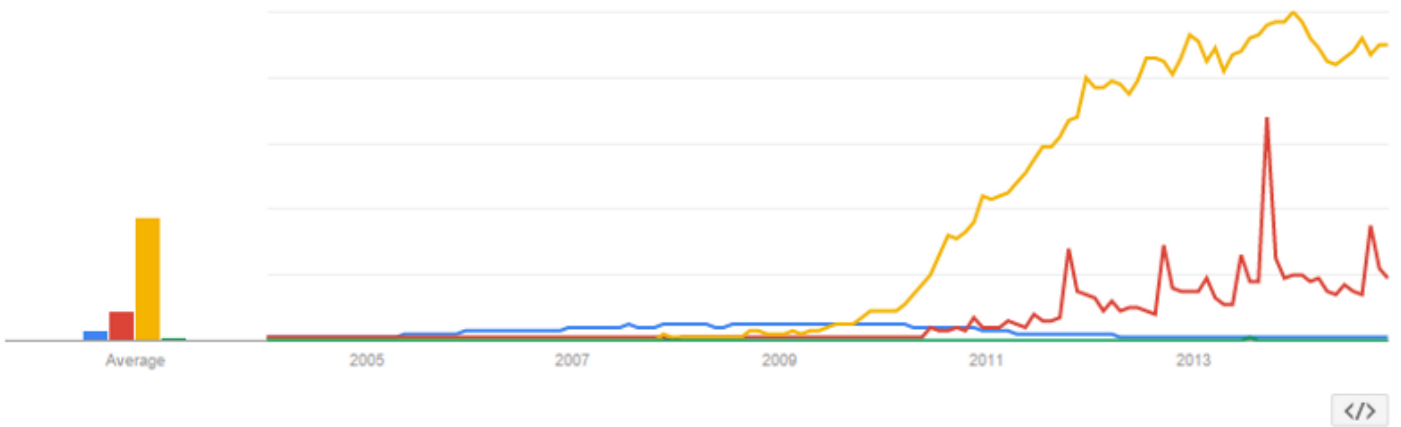
Topics

Subscribe

- windows ...**  
Search term
- ios**  
Search term
- android**  
Search term
- firefox os**  
Search term
- [+ Add term](#)

Interest over time

News headlines  Forecast



Regional interest

windows mobile ios android firefox os



Region	City	Interest Score
Ghana		100
Madagascar		93
Czech Republic		89
Maldives		83
India		82
Cameroon		75
Nepal		71

View change over time

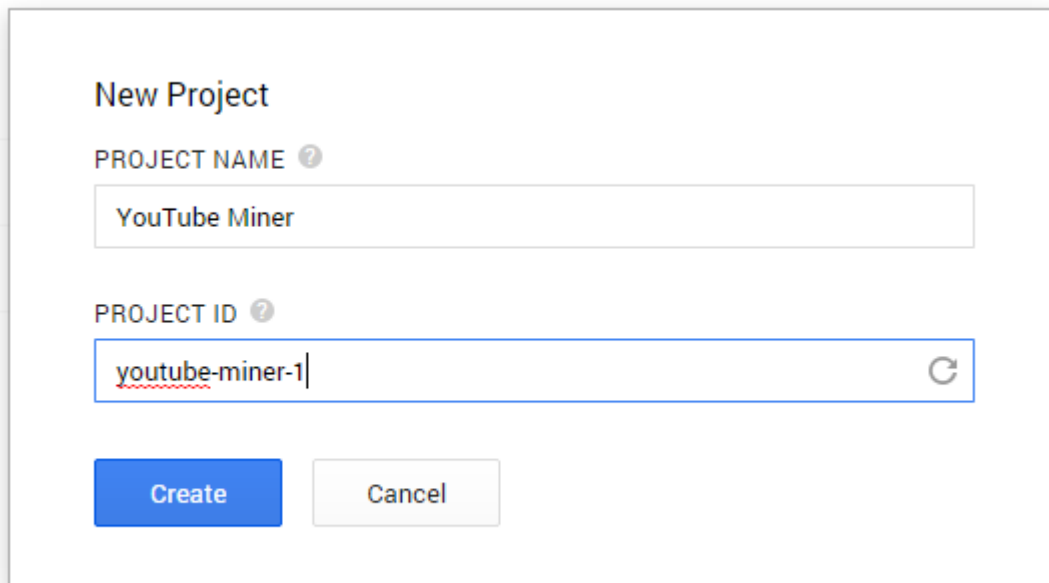
Interestingly, from the above visualization, we see that 'Windows Mobile' was quite popular from 2007 till mid 2009 when the popularity of 'Android' just skyrocketed. Apple's 'iOS' gained popularity sometime around 2010. One must remember however that this data is purely based on data tracked by Google searches.

Coming to YouTube, it is perhaps the most popular video sharing website and I am sure all of you have at least watched a video on YouTube. Interestingly,

we can also get a lot of interesting statistics from these videos besides just watching them, thanks to some great APIs provided by Google.

In the next part, I will discuss how to get interesting statistics from YouTube based on a search keyword and do some basic analysis. I won't be delving into the depths of data analytics here but I will provide you enough information to get started with data mining from YouTube. We will be using Google's [YouTube Data API](#), some Python wrappers for the same and the [pandasframework](#) to analyze the data.

First, we would need to go to the [Google Developers Console](#) and create a new project just like the snapshot shown below.



The image shows a 'New Project' dialog box from the Google Developers Console. It has a title 'New Project' and two input fields. The first field is labeled 'PROJECT NAME' with a help icon and contains the text 'YouTube Miner'. The second field is labeled 'PROJECT ID' with a help icon and contains the text 'youtube-miner-1'. A red dashed underline is visible under the 'y' in 'youtube'. To the right of the second field is a refresh icon. At the bottom of the dialog are two buttons: a blue 'Create' button and a grey 'Cancel' button.

Once the project is created, you will be automatically re-directed to the dashboard for the project: There you can choose to enable the APIs you want for your application. Go to the APIs section on the left and enable the YouTube Data API v3 just like it is depicted in the snapshot below (click it if you are unable to make out the text in the image).

**Enabled APIs**  
Some APIs are enabled automatically. You can disable them if you're not using their services.

NAME	QUOTA	STATUS
BigQuery API	0%	ON
Google Cloud SQL		ON
Google Cloud Storage		ON
Google Cloud Storage JSON API		ON
YouTube Analytics API		ON
YouTube Data API v3		ON

**Browse APIs**  
youtube

No results found

Now, we will create a new API key for public API access. For this, go to the Credentials section and click on Create new key and choose the Server key option and create a new API key which is shown in the snapshot below (click to zoom the image).

**OAuth**  
OAuth 2.0 allows users to share specific data with you (for example, contact lists) while keeping their usernames, passwords, and other information private.  
[Learn more](#)  
[Create new Client ID](#)

**Public API access**  
Use of this key does not require any user action or consent, does not grant access to any account information, and is not used for authorization.  
[Learn more](#)  
[Create new Key](#)

**Create a new key**  
The APIs represented in the Google Developers Console require that requests include a unique project identifier. This enables the Console to tie a request to a specific project in order to monitor traffic, enforce quotas, and handle billing.

[Server key](#) [Browser key](#) [Android key](#) [iOS key](#)

We will be using some Python libraries so open up your terminal or command prompt and install the following necessary libraries if you don't have them.

```
[root@dip]# pip install google-api-python-client [root@dip]# pip install pandas
```

Now that the initial setup is complete, we can start writing some code! Head over to your favorite Python IDE or console and use the following code segment to build a YouTube resource object.

```

from apiclient.discovery import build
from apiclient.errors import HttpError
import pandas as pd
DEVELOPER_KEY = "REPLACE WITH YOUR KEY"
YOUTUBE_API_SERVICE_NAME = "youtube"
YOUTUBE_API_VERSION = "v3"
youtube = build(YOUTUBE_API_SERVICE_NAME, YOUTUBE_API_VERSION, developerKey=DEVELOPER_KEY)

```

Once this is complete, we will be using this YouTube resource object to search for videos with the android keyword. For this we will be using the [search method](#) to query YouTube and after getting back a list of results, we will be storing each result to its appropriate list, and then display the lists of matching videos, channels, and playlists using the following code segment.

```

search_response = youtube.search().list( q="android", part="id,snippet", maxResults=50 ).execute()
videos = []
channels = []
playlists = []
for search_result in search_response.get("items", []):
    if search_result["id"]["kind"] == "youtube#video":
        videos.append("%s (%s)" % (search_result["snippet"]["title"], search_result["id"]["videoId"]))
    elif search_result["id"]["kind"] == "youtube#channel":
        channels.append("%s (%s)" % (search_result["snippet"]["title"], search_result["id"]["channelId"]))
    elif search_result["id"]["kind"] == "youtube#playlist":
        playlists.append("%s (%s)" % (search_result["snippet"]["title"], search_result["id"]["playlistId"]))

```

Based on the query I ran, most of the results seemed to be videos. The output I obtained is shown in the snapshot below.

```

In [96]: videos[:5]
Out[96]:
[u'Android 5.0 Lollipop Feature Review! (pEDQ1z1-PvU)',
 u'#080 - Os 5 melhores aplicativos para Android - #A19-118 (OdotwjuRRxg)',
 u'Moto X (2nd gen) Android 5.0 "Lollipop" Tour! (EOfCq_bR7uw)',
 u'Android Lollipop Arrival - Google Calendar Update - Samsung Galaxy S6 Rumor (GEr_TMoQYII)',
 u'5 True Facts about Android! (1gUQhN5joSQ)']

In [97]: channels
Out[97]: [u'Android Authority (UCgyqtNWZmIxTx3b60xTSALw)']

In [98]: playlists
Out[98]: []

```

Since the playlists and channels we obtained are very less in number, I decided to go ahead and analyze the videos obtained. For that, we create a dict of video identifiers and video names. Then we pass a query to the YouTube API's [videos method](#), to get the relevant statistics for each video.

```

videos = {}
for search_result in search_response.get("items", []):
    if search_result["id"]["kind"] == "youtube#video":
        videos[search_result["id"]["videoId"]] = search_result["snippet"]["title"]
video_ids_list = ','.join(videos.keys())
video_list_stats = youtube.videos().list( id=video_ids_list, part='id,statistics' ).execute()

```

I know you must be interested by now to see what kind of data is present in `video_list_stats`. So for that, I will show you the relevant statistics obtained for a video from the API in the following snapshot.

```
In [107]: video_list_stats['items'][0]
Out[107]:
{u'etag': u'"PSjn-HSKiX6orvNhGZvg1LI21vk/QkLJVeOvyOh-PedHBMNgpnKRPmY"',
 u'id': u'Dh3hKaG_Nqc',
 u'kind': u'youtube#video',
 u'statistics': {u'commentCount': u'222',
 u'dislikeCount': u'38',
 u'favoriteCount': u'0',
 u'likeCount': u'1187',
 u'viewCount': u'99172'}}
```

Now we will be using pandas to analyze this data. For that, the following code segment is used, to get this data into a [pandas data frame](#).

```
df = []
for item in videos_list_stats['items']:
    video_dict = dict(video_id = item['id'],
                      video_title = videos[item['id']])
    video_dict.update(item['statistics'])
    df.append(video_dict)
df = pd.DataFrame.from_dict(df)
```

Now, we can view the contents of this data frame. I will be showing the output of the first few rows with the relevant columns in the snapshot below. I have considered only the important data points which include `viewCount`, `likeCount`, `dislikeCount`, `commentCount` indicating the number of views, likes, dislikes and comments on the videos respectively.

```
In [169]: df[['video_id', 'video_title', 'viewCount', 'likeCount', 'dislikeCount', 'commentCount']].head(5)
Out[169]:
```

	video_id	video_title	viewCount	likeCount	dislikeCount	commentCount
0	Dh3hKaG_Nqc	Android 5.0 Lollipop Developer Preview Review	99173	1187	38	222
1	t5XL50IuGlg	An Android Enthusiast's Apple iPhone 6 Challenge	78620	3706	596	1837
2	q-hq2Zx_ojw	Como tener Internet gratis en Android - BIEN e...	805564	28446	1089	2680
3	edXWZSTOgKo	Android 5.0 "Lollipop" Tour!	218600	1845	42	386
4	-KGz5L8p6Mc	8 APLICACIONES Y JUEGOS IMPRESCINDIBLES de And...	35891	1527	34	94

Once we have this table of clean and formatted data, we can do all sorts of analytics on it, like getting the mean and median for number of views, seeing which are really popular videos and so on. Some examples with required code segments are depicted below. I have used the [IPython](#) shell for analyzing the data.

### Mean and Median of different counts

```
In [48]: print '\n\tMEAN\t\t\tMEDIAN\n\nViews: %s\t\tViews: %s \
...: \nLikes: %s\t\tLikes: %s\nDislikes: %s\t\tDislikes: %s \
...: \nComments: %s\t\tComments: %s' \
...: %(df['viewCount'].mean(), df['viewCount'].median(),
...: df['likeCount'].mean(), df['likeCount'].median(),
...: df['dislikeCount'].mean(), df['dislikeCount'].median(),
...: df['commentCount'].mean(), df['commentCount'].median())
```

MEAN	MEDIAN
Views: 232225.541667	Views: 80863.0
Likes: 5165.60416667	Likes: 1978.5
Dislikes: 191.666666667	Dislikes: 58.0
Comments: 699.875	Comments: 262.5

### Top ten most viewed videos

```
In [50]: df.sort(['viewCount'], ascending=False)[['video_title', 'viewCount']].head(10)
Out[50]:
```

	video_title	viewCount
38	Top 20 Best Android Apps 2014	1493817
37	70 Mejores Juegos Android 2014	1144793
18	Android 5.0 Lollipop Feature Review!	842629
2	Como tener Internet gratis en Android - BIEN e...	805564
36	Android: And you	791182
23	5 True Facts about Android!	659782
34	Como Actualizar Android a la Ultima Version	608113
44	Android L vs Android KitKat	523379
25	Exclusive Preview - Android 5.0 Lollipop on Sa...	486591
35	8 cool alternate uses for an Android smart phone!	459750

### Top ten most liked videos



```
In [51]: df.sort(['likeCount'], ascending=False)[['video_title', 'likeCount']].head(10)
Out[51]:
```

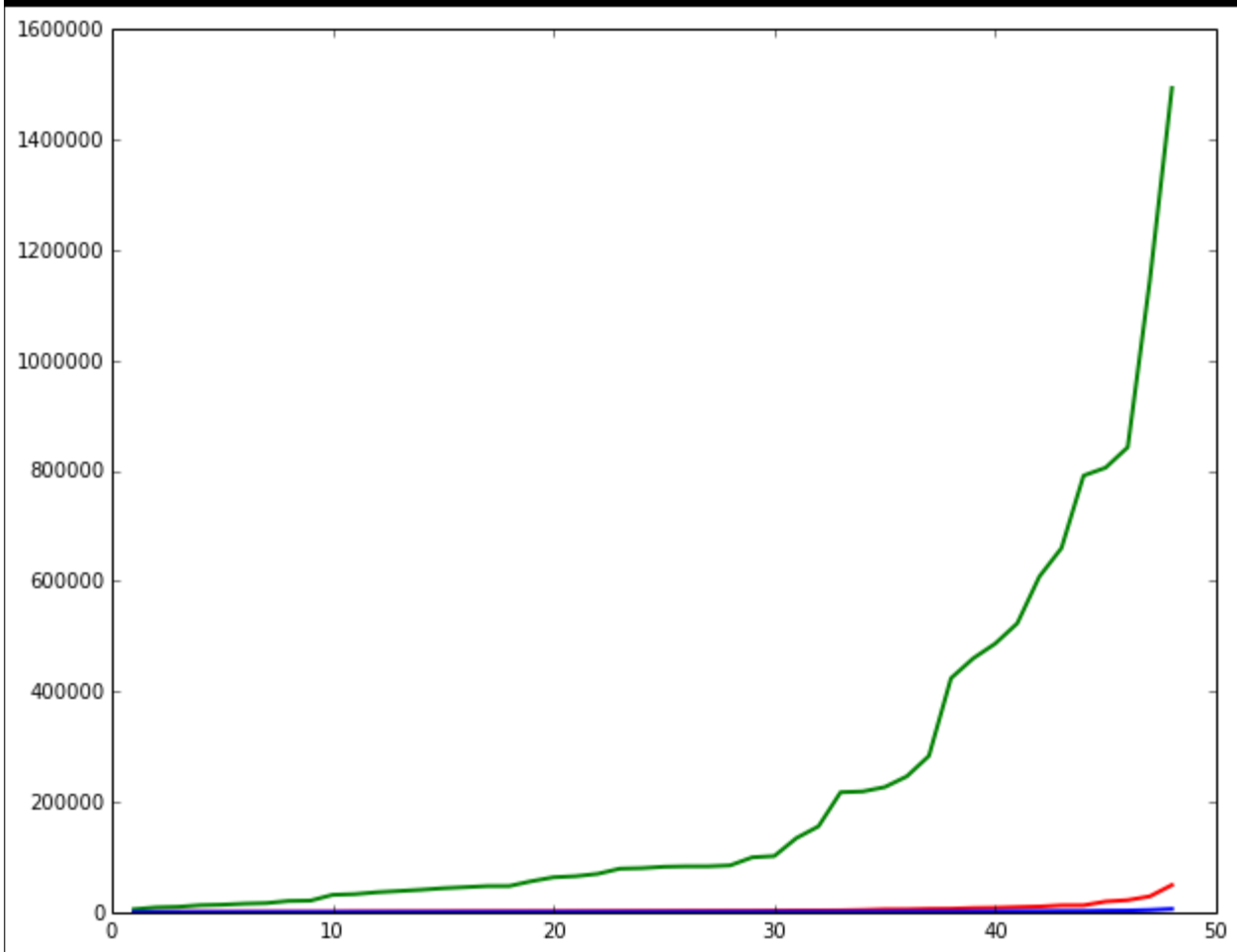
	video_title	likeCount
18	Android 5.0 Lollipop Feature Review!	49028
2	Como tener Internet gratis en Android - BIEN e...	28446
37	70 Mejores Juegos Android 2014	21654
23	5 True Facts about Android!	19086
35	8 cool alternate uses for an Android smart phone!	12234
34	Como Actualizar Android a la Ultima Version	12135
38	Top 20 Best Android Apps 2014	9964
17	Полный обзор Android 5.0 Lollipop. Лучшая моби...	8964
36	Android: And you	7813
44	Android L vs Android KitKat	7305

Line chart showing counts of likes, views and comments

```
In [58]: import matplotlib.pyplot as plt

In [59]: plt.plot(
...: range(1,49),df.sort(['likeCount'],
...: ascending=[0])[['likeCount']], 'r',
...: range(1,49),df.sort(['viewCount'],
...: ascending=[0])[['viewCount']], 'g',
...: range(1,49),df.sort(['commentCount'],
...: ascending=[0])[['commentCount']], 'b',
...: linewidth=2.0)
```

Out[59]:



Thus you can see by now that a lot of interesting analysis and visualizations can be built on top of this data. For more details on how to customize and use the Youtube API with Python, you can refer to this page for sample code segments.

Originally published at [blog.dataweave.in](http://blog.dataweave.in).

- [DataWeave Marketing](#)

4th Aug, 2015

E COMMERCE

