International Journal of Current Engineering and Technology
©2017 INPRESSCO®, All Rights Reserved

E-ISSN 2277 – 4106, P-ISSN 2347 – 5161
Available at http://inpressco.com/category/ijcet

General Article

Infotube-A Study of Youtube Trends

Alisha^, Mamta Mittal*#, Sanjot Kaur^, Tavleen Kaur^ and Avneet Singh Malhotra^

^G. T. B. I. T., GGSIPU, New Delhi, India
#G. B. Pant Govt. Engineering College, GGSIPU, New Delhi, India

Received 30 Oct. 2017, Accepted 10 Nov. 2017, Available online 23 Nov. 2017, **Vol.7, No.6 (Nov/Dec 2017)**

Abstract

In today's technological world, social media serves as a mirror for analyzing the behaviour of people, especially the youth. The analysis of choices and likes becomes the foundation for industries for further product developments and helps them to survive competition. In this paper, we present a detailed procedure for analyzing YouTube trends by the application of machine learning algorithms on trends, and by comparing the results produced by two machine learning algorithms. In this paper, the various YouTube trends have been analyzed and hidden information has been extracted based on user statistics.

Keywords*: K-means Clustering, Agglomerative Clustering, Python, YouTube, Social media, YouTube data API.*

1. Introduction

In the present scenario, due to the increase in the amount of data, the analysis of data has become essential and the process of extracting valuable information has become a challenging task. Social media is a platform where numbers of products start trending even before products are launched. You Tube is one such media where you may stream anything with the provision of searching and sorting also. YouTube data analytics is used as a driver for improving operational performance and supporting decision making for better matching supply and demand. Then, the challenge is to identify analytics techniques or methods chain for analyzing YouTube trends and creating business value. In this paper, a comprehensive overview of YouTube trends use in supply chain and underline its potential role in supply chain transformation is performed. For this to happen we conducted a study of: - Which Machine Learning algorithms should be incorporated to analyses trends? -How to perform data visualization? The objective of this paper is to extract data from YouTube using Google's YouTube Data API based on search keyword and use Python to analyze the YouTube data (Cheng *et al* 2008).

2. Background Work

Data clustering is technique for unsupervised learning where the target variables need not be defined. Clustering divides the data into number of clusters or groups based on the similarity of data. It finds application in number of fields like pattern matching and data mining. The data in a cluster is similar as compared to data outside the cluster (Äyrämö *et al* 2006). There are a number of clustering algorithms. In the following section, K-means and Agglomerative Hierarchical Clustering have been described.

2.1 K-means Clustering

K –means Clustering is a type of clustering where the number of clusters is fixed. It defines a cluster centre for each cluster in an iterative fashion such that Euclidean distance is minimized. Initially the algorithm determines the cluster centres randomly. Then the membership function is determined and the cost function is calculated by the following equation. Given a set of *n* vectors a_k , k =1,……, *n* , that need to be partitioned into *m* groups , g_l , l=1,…,m . The cost function can be defined by:

$$J(v) = \sum_{k=1}^{n} \sum_{l=1}^{ck} (||a_k - v_l||)^2$$

where,

$||x_i - v_l||$ is the Euclidean distance.
C_k is the number of data points in kth cluster.
C is the number of cluster centres (Kanungo *et al* 2002).

The K-Means Clustering algorithm stops if the value of the cost function is below a threshold value or the improvement from the earlier iteration is less than threshold. The cluster centers are further updated and

*Corresponding author's ORCID ID: 0000-0003-0490-4413 the same steps in the iteration are repeated (Ortega *et al*).

2.2 Agglomerative Hierarchical Clustering

Agglomerative Hierarchical Clustering algorithm finds the distance between all the pairs of data points. It is based on bottom to up approach where the data points are initially randomly assigned to any cluster and then the clusters are combined. The metric used for merge approach is based on minimum sum of square of distance of the data points. The linkage function minimizes the distance between data points in two clusters (Fung *et al*, 2003). It begins with a cluster with the minimum sequence number and finds the minimum distance between pair of clusters. Then the sequence number of cluster is incremented and the clusters are merged. Finally the distance matrix is updated. If all the data points are in the same cluster, then the process halts. Otherwise the same process is repeated (Borgatti, 1994).

3. Methodology

In this paper, we compare the results obtained on performing clustering on the top 50 trending videos. The data about the videos has been obtained from the YouTube data API (V3) (https:// developers.google.com/ youtube/ v3/) and the analysis has been performed using python as the coding language. The following steps have been followed in order to extract the data for this analysis:

1) Create an account on the Google Developer Console
2) Create a new project on the GDC
3) Add the YouTube data API to your project
4) Create a developer's key for the API
5) Enter the developer's key in your python script
6) Set the search parameters to obtain the results.
7) Gather the statistics on the videos in python lists
8) Form a pandas data frame from these lists

The pandas data frames are used for obtaining the data in the required format for performing clustering. Clustering is performed on the number of views and the number of likes on the top 50 trending videos. Two clustering algorithms have been implemented in python for comparing the results and for obtaining useful information about YouTube trends. The number of clusters for both the algorithms has been selected as 10.

The first algorithm that has been implemented is k-means clustering. The following python code implements k-means on the data and the data is divided into 10 clusters:

```
>>> f1=df['Views'].values
>>>f2=df['Likes'].values
>>>X=np.matrix(list(zip(f1,f2)))
>>>kmeans = KMeans(n_clusters=10).fit(X)
>>>print(kmeans.labels_)
```

The second algorithm that has been implemented is agglomerative clustering. The following python code implements agglomerative clustering on the data and the data is divided into 10 clusters:

```
>>> f1=df['Views'].values
>>>f2=df['Likes'].values
>>>X=np.matrix(list(zip(f1,f2)))
>>>Hclustering=AgglomerativeClustering(n_clusters=1
0, affinity='manhattan', linkage='average')
>>>Hclustering.fit(X)
>>>print(Hclustering.labels_)
```

The affinity value for the Agglomerative Clustering function has been changed to Euclidean as well as cosine and the results have been compared. The above steps have also been performed on the number of views and the difference between the number of views and the number of likes on each video. For applying K-Means on the new data, following changes have been made:

```
>>> f1=df['Views'].values
>>>f2=df['Likes'].values
>>>f3=[]
>>>for i in range(0,50):
f3.append(df['Views'][i]-df['Likes'][i])
>>>X=np.matrix(list(zip(f1,f3)))
>>>kmeans = KMeans(n_clusters=10).fit(X)
>>>print(kmeans.labels_)
```

For applying agglomerative clustering on the new data, following changes have been made:

```
>>> f1=df['Views'].values
>>>f2=df['Likes'].values
>>>f3=[]
>>>for i in range(0,50):
f3.append(df['Views'][i]-df['Likes'][i])
>>>X=np.matrix(list(zip(f1,f3)))
>>>>Hclustering=AgglomerativeClustering(n_clusters=
10, affinity='manhattan', linkage='average')
>>>Hclustering.fit(X)
>>>print(Hclustering.labels_)
```

The results have been visualized using matplotlib.pyplot module and each cluster has been represented with a different color. The following code is used to visualize the results of both these algorithms:

```
>>>a1=kmeans.labels_
>>>res1=np.array(['blue','red','yellow','green','black','
magenta',' cyan','brown','m','navy'])
>>>plt.scatter(f1,f2,c=res1[a1],s=50)
>>>plt.xlabel('Views')
>>>plt.ylabel('Likes')
>>>plt.title('Clustering')
>>>plt.show()
```

4. Results

The following figures visualize the clusters formed by implementing kmeans as well as agglomerative clustering on the YouTube data.

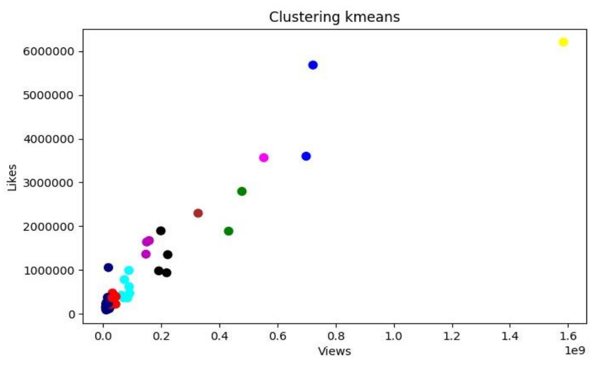

Figure 1: K-Means Clustering on views and likes with n=10

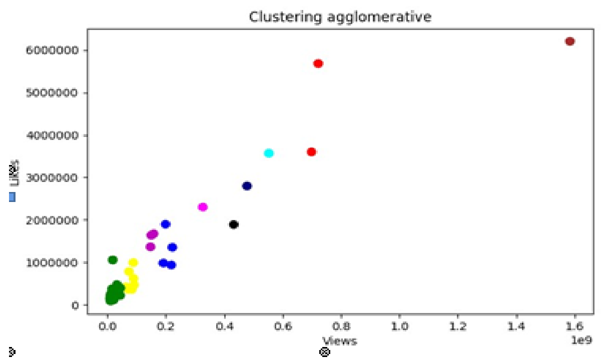

Figure 2: Agglomerative Clustering on views and likes with n=10 and affinity='euclidean'

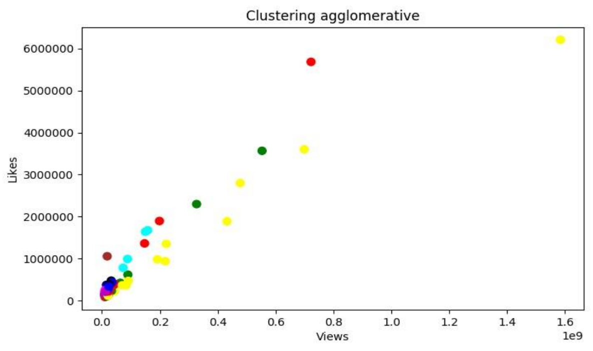

Figure 3: Agglomerative Clustering on views and likes with n=10 and affinity='cosine'

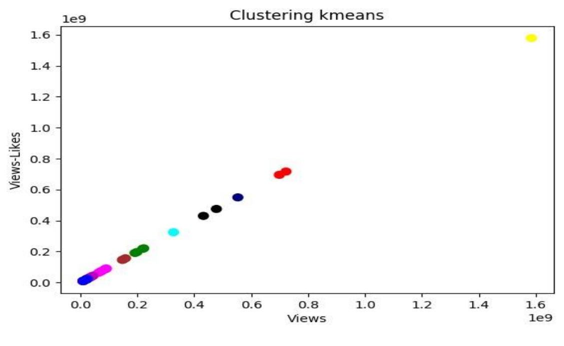

Figure 4: K -Means Clustering on views and difference between views and likes with n=10

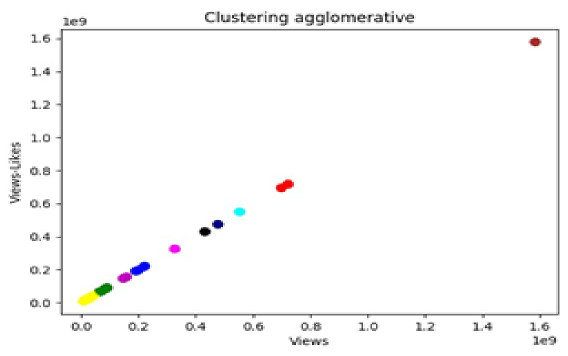

Figure 5: Agglomerative Clustering on views and difference between views and likes with n=10 and affinity='euclidean'

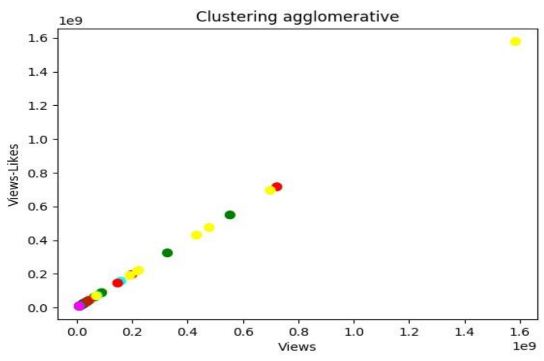

Figure 6: Agglomerative clustering on views and difference between views and likes with n=10 and affinity='cosine'

Conclusion

Based on the above visual results, we conclude that both Agglomerative Clustering as well as clustering using K -means produce similar results. This may be due to the limited nature of the dataset chosen. However, there are some noticeable differences which can be summarized as follows:

- Agglomerative clustering with affinity='euclidean' produces highly distinguishable clusters
- Kmeans clustering produces somewhat distinguishable clusters but some clusters overlap.
- Agglomerative clustering with affinity='cosine' produces overlapping clusters.

Thus, agglomerative clustering with Euclidean distance produces the best results out of the three since there is minimum overlapping between clusters.

Another conclusion that we draw in this paper is that the clusters drawn on the basis of number of views and the difference between number of views and number of likes produce a straight line. Thus, there is a direct proportionality between the viewers who view the top trending videos and the viewers who like these videos. Therefore, an equal proportion of people tend to like the trending videos out of all the people who view these videos. The final conclusion drawn on the basis of these visual results is that only a minority of the top 50 trending videos gets a large number of views and likes since the majority of the clusters plotted are present in the beginning of the graphs or in the fourth quarter of the graphs. Thus, only a few videos gain enormous popularity and rest of the trending videos gain statistically similar popularities.

References

Cheng, X., Dale, C., & Liu, J. (2008, June). Statistics and social network of youtube videos. In Quality of Service, 2008. IWQoS 2008. 16th International Workshop on (pp. 229-238). IEEE.

Äyrämö, S., & Kärkkäinen, T. (2006). Introduction to partitioning-based clustering methods with a robust example. Reports of the Department of Mathematical Information Technology. Series C, Software engineering and computational intelligence 1/2006.

Kanungo, T., Mount, D. M., Netanyahu, N. S., Piatko, C. D., Silverman, R., & Wu, A. Y. (2002). An efficient k-means clustering algorithm: Analysis and implementation. IEEE transactions on pattern analysis and machine intelligence, 24(7), 881-892.

Ortega, J. P., Del, M., Rojas, R. B., & Somodevilla, M. J. Research issues on k-means algorithm: An experimental trial using matlab. In CEUR Workshop Proceedings: Semantic Web and New Technologies.

Fung, B. C., Wang, K., & Ester, M. (2003, May). Hierarchical document clustering using frequent itemsets. In Proceedings of the 2003 SIAM International Conference on Data Mining (pp. 59-70). Society for Industrial and Applied Mathematics.

Borgatti, S. P. (1994). How to explain hierarchical clustering.

https://developers.google.com/youtube/v3/