Data Intensive Computing

Big-Data Content Retrieval, Storage and Foundation analysis of Data Intensive Computing

Submitted By:-

Sagar Vishwakarma

Saurabh Warvadekar

Contents

Problem Statement

- To find out the most trending words
- To find out co-occurring hash tags,
- To cluster the tweets by the number of followers they have as low, medium and high followers respectively.
- For the most popular hash tag, determine the shortest path between nodes of this network

Introduction

Apache Hadoop is an open-source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware.

The Apache Hadoop framework is composed of the following modules:

- Hadoop Common – contains libraries and utilities needed by other Hadoop modules
- Hadoop Distributed File System (HDFS) – a distributed file-system that stores data on commodity machines, providing very high aggregate bandwidth across the cluster.
- Hadoop YARN – a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications.
- Hadoop MapReduce – a programming model for large scale data processing.

Big data is the term for a collection of data sets so large and complex that it becomes difficult to process using on-hand database management tools or traditional data processing applications

Data Collection

We have collected lots of twitter data as Big data to analyze in the project. A row of data is given below:-

1689!!!!!!!1408!!!!!!!ur so lovely!!!!!!!RT @lovelykellin: my english is v bad i'm so sorry i'm german i'm allowed to make mistakes lmao bye

The data is described below:-

- We have three different entities in the data as friends, followers count and the tweet itself
- The entities are separated by seven exclamatory signs(!)
- The first entity gives the friends count
- The Second entity gives the followers count
- The third entity gives the tweet
- A tweet composes of hash tags(#), user tags(@) and the tweet.

Word Count

Problem

We want to find out the most trending words

Approach

- *First we tokenize all the words present in the input file.*
- *Then in TokenizerMapper class we will remove the stop words.*
- *Output of IntSumReducer will tokenize all the words and give them a count of 1 each.*
- *Output of IntSumReducer will serve as input for the rest of three mappers which are countmapper, hashmapper and attheratemapper.*
- *In count Mapper the sum of each of the words occurrence would be done, so ultimately its reducer will return the words which have occurred the most number of times in descending order*
- *In hash Mapper the sum of each of the trending words' occurrence would be done, so ultimately its reducer will return the words which have trended the most number of times in descending order*
- *In at the rate Mapper the sum of each of the words occurrence would be done, which contains '@' sign, so ultimately its reducer will return all such which have occurred the most number of times in descending order*
-

Pseudo Code

Results

The top 10 words obtained were:-

Word	**Count**
Love	22393
One	20704
Know	19024
People	18990
Want	17973
Se	17492
Te	17382

Via	16619
Es	16080
New	15787

Some top words as a visualiztion are:

Most Trending Hash (#) Tags

Problem

We want to find out the most occurring hash tags.

Approach

Pseudo Code

Results

The top 10 hash tags obtained are:-

Hash Tags	*Count*
#DhaniDiBlockRicaJKT	*5504*
#VoiceSave	*5336*
#iHeartAwards	*4974*
#gameinsight	*4778*
#UNSMPNYKSukses	*3184*
#NBLNabilaVoto	*2960*
#BestCollaboration	*2876*
#androidgames	*2457*
#Wicenganteng	*2328*
#android	*2180*

A visualization of some of the obtained hash tags is given below:


androidgames
gameinsight
NBLJosephine Voto
android
AgentsofSHIELD
voicesavetess
VoiceResults
PourItUp
iPadGames
GleeAlong
HipHopSOTY
IsmailSangHekerCRB
LouisIsOurPerfectAngel
flawlessfriendiefollowspree
Chelsea
Androidgames
TeamFollowBack
followmeshabazz
RihannaNavy
SpinnrTaylorSwift
mPOINTS
TANIA
iPhone
EarthDay
serenity
spring
AkuNanya
MasakitIsipinNa
TIME
win
teot
CFC
Lorde
BRAZIL
SECRETFOLLOWSPREEEYEE
TheNeonLightsTour
ipadgames
CSRClassics
RETWEET
SOTY
TweetKepo
Taurus
News
NP
tvtag
wedding
QuizIG
Scorpio
SMPNBatam
Libra
BestArtistGetWellSoonKushal
VERIFIED
SoundCloud
FaZe
RT
Cancer
Aries
MUFC
NBAPlayoffs
FuegoMejores
Stay
movie
VoiceSaveTJ
iHeartAwards
BestCollaboration
ChibiCyberCommunity
WeTheNorth
armony
hot
FanArmy
myNYPD
VirgoLittleMonsters
NBLNabila
HolyGrail
ChampionsLeague
PremiereFilm
Pisces
video
music
jobs
GameInsight
pussy
ebook
Sagittarius
Supernatural
EPNvsInternet
NoMeVaQue
Aquarius
Justin Timberlake
silence
AtletiChelsea
nowplaying
Tiaraayu
DaysUntilProblem
Leo
GlobalSelfie
Always
earthday
Gemini
free
sexy
np
life
job
SosPascal
Siminesian
GuysEggs
fashion
iPad
voicesavetj
nature
hope
zen
sex
lol

Most Trending Counts (@)

Problem

We want to find out the most occurring Counts (@)

Approach

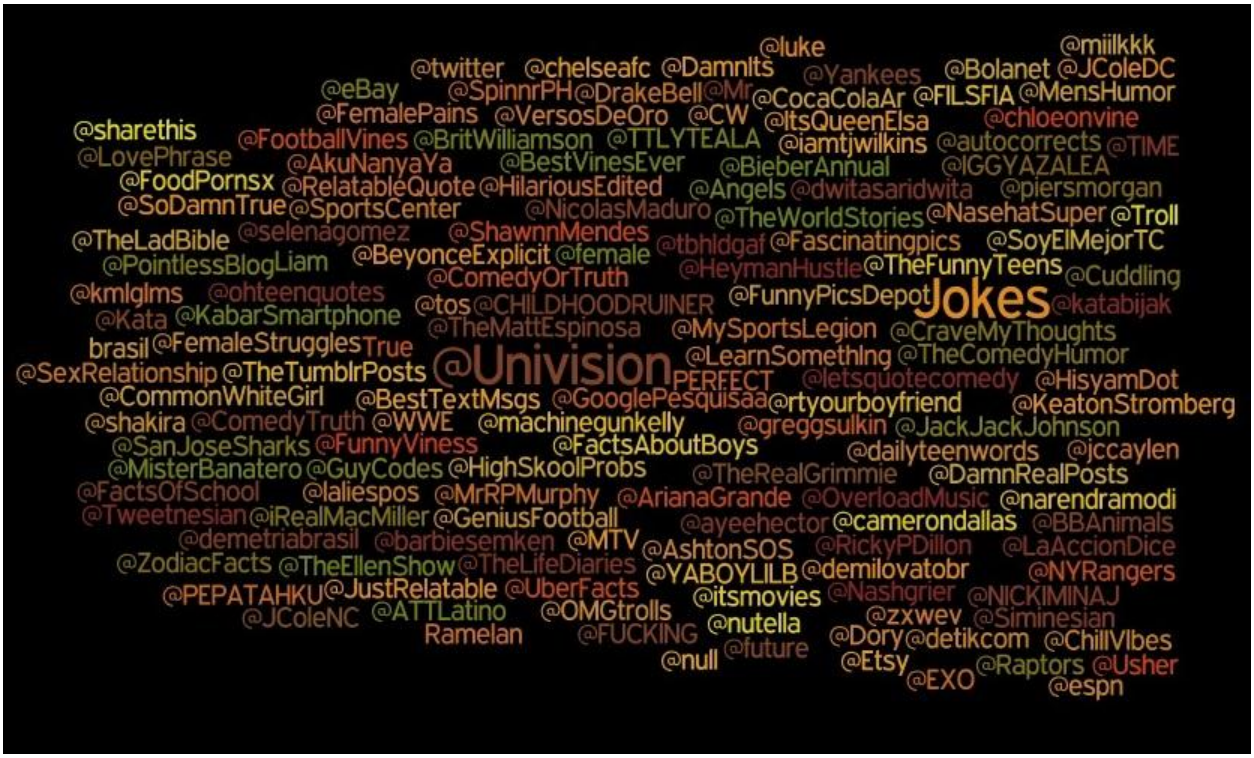
Pseudo Code

Results

The top 10 @ tags obtained are as shown below:

@justinbieber	*4583*
@YouTube	*3526*
@TheMattEspinosa	*2539*
@null	*1876*
@Real_Liam_Payne	*1837*
@LukeSOS	*1604*
@jccaylen	*1602*
@NBCTheVoice	*1535*
@BieberAnnual	*1486*
@HisyamDot_ID	*1398*

A visualization is shown below:-


@luke
@miilkkk
@twitter
@chelseafc
@DamnIts
@Yankees
@Bolanet
@JColeDC
@eBay
@SpinnrPH
@DrakeBell
@Mr
@CocaColaAr
@FILSFIA
@MensHumor
@FemalePains
@VersosDeOro
@CW
@ItsQueenElsa
@chloeonvine
@sharethis
@FootballVines
@BritWilliamson
@TTLYTEALA
@iamtjwilkins
@autocorrects
@TIME
@LovePhrase
@AkuNanyaYa
@BestVinesEver
@BieberAnnual
@IGGYAZALEA
@FoodPornsx
@RelatableQuote
@HilariousEdited
@Angels
@dwitasaridwita
@piersmorgan
@SoDamnTrue
@SportsCenter
@NicolasMaduro
@TheWorldStories
@NasehatSuper
@Troll
@TheLadBible
@selenagomez
@ShawnnMendes
@tbhIdgaf
@Fascinatingpics
@SoyElMejorTC
@PointlessBlog
Liam
@BeyonceExplicit
@female
@HeymanHustle
@TheFunnyTeens
@Cuddling
@ComedyOrTruth
@kmlglms
@ohteenquotes
@tos
@CHILDHOODRUINER
@FunnyPicsDepot
Jokes
@katabijak
@Kata
@KabarSmartphone
@TheMattEspinosa
@MySportsLegion
@CraveMyThoughts
brasil
@FemaleStrugglesTrue
@LearnSomethIng
@TheComedyHumor
@SexRelationship
@TheTumblrPosts
@Univision
PERFECT
@letsquotecomedy
@HisyamDot
@CommonWhiteGirl
@BestTextMsgs
@GooglePesquisaa
@rtyourboyfriend
@KeatonStromberg
@shakira
@ComedyTruth
@WWE
@machinegunkelly
@greggsulkin
@JackJackJohnson
@SanJoseSharks
@FunnyViness
@FactsAboutBoys
@dailyteenwords
@jccaylen
@MisterBanatero
@GuyCodes
@HighSkoolProbs
@TheRealGrimmie
@DamnRealPosts
@FactsOfSchool
@laliespos
@MrRPMurphy
@ArianaGrande
@OverloadMusic
@narendramodi
@Tweetnesian
@iRealMacMiller
@GeniusFootball
@ayeehector
@camerondallas
@BBAnimals
@demetriabrasil
@barbiesemken
@MTV
@AshtonSOS
@RickyPDillon
@LaAccionDice
@ZodiacFacts
@TheEllenShow
@TheLifeDiaries
@YABOYLILB
@demilovatobr
@NYRangers
@PEPATAHKU
@JustRelatable
@UberFacts
@itsmovies
@Nashgrier
@NICKIMINAJ
@JColeNC
@ATTLatino
@OMGtrolls
@nutella
@zxwev
@Siminesian
Ramelan
@FUCKING
@future
@Dory
@detikcom
@ChillVibes
@null
@Etsy
@Raptors
@Usher
@EXO
@espn

Co-occuring hash (#) Tags

Problem

We want to find out co-occurring hash tags

Approach

- *In Pairs approach we pass the each word with all the other words in a tweet which contain '#' and occur after the original word in the mapper .*
- *The partitioner here divides the keys emitted by the mapper by taking the hash vaue of the key and taking its modulus with the number of partitioners implemented.*
- *This hash value could some times be negative hence we used Math.abs*
- *In reducer we count the number of times a root word was sent along with its different neighboring words.*
- *In this way we have this number*
- *We divide the number of occurrence of each root-neighbor pairs occurrence with the number of times the root word occurred in the reducer using a hash map.*
- *Thus this division provides us the relative frequency of that root-neighbor pair.*

Pseudo Code

Results

Kmeans Clustering

Problem

We want to cluster the tweeters by the number of followers they have. We will need three clusters where the average number of followers is low, medium and high respectively, where the actual average value for each cluster will depend on your data size. This information may be used for marketing and for targeting the ads/messages.

Approach

- Take two input files: one having the followers list and other an initial centers list have three numbers:-

  1500
  100000
  3500000

- Map each point to a nearest center from the list and send it to the reducer
- Aggregate all the points for a particular center and calculate the average for each
- Replace the centers list with the new centers
- Run till convergence reached
- Since, Convergence may require a lot of iterations. For convenience we have used 12 iterations. Although convergence was still reached

Pseudo Code

```
Kmeansmapper class
{
  void config()
    { //take centers from the distributed cache file centerslist.txt and store it in a list
    }

  Void map()
    { //map the points to the nearest center in the array list
    }
}

Kmeansreducer class
{
```

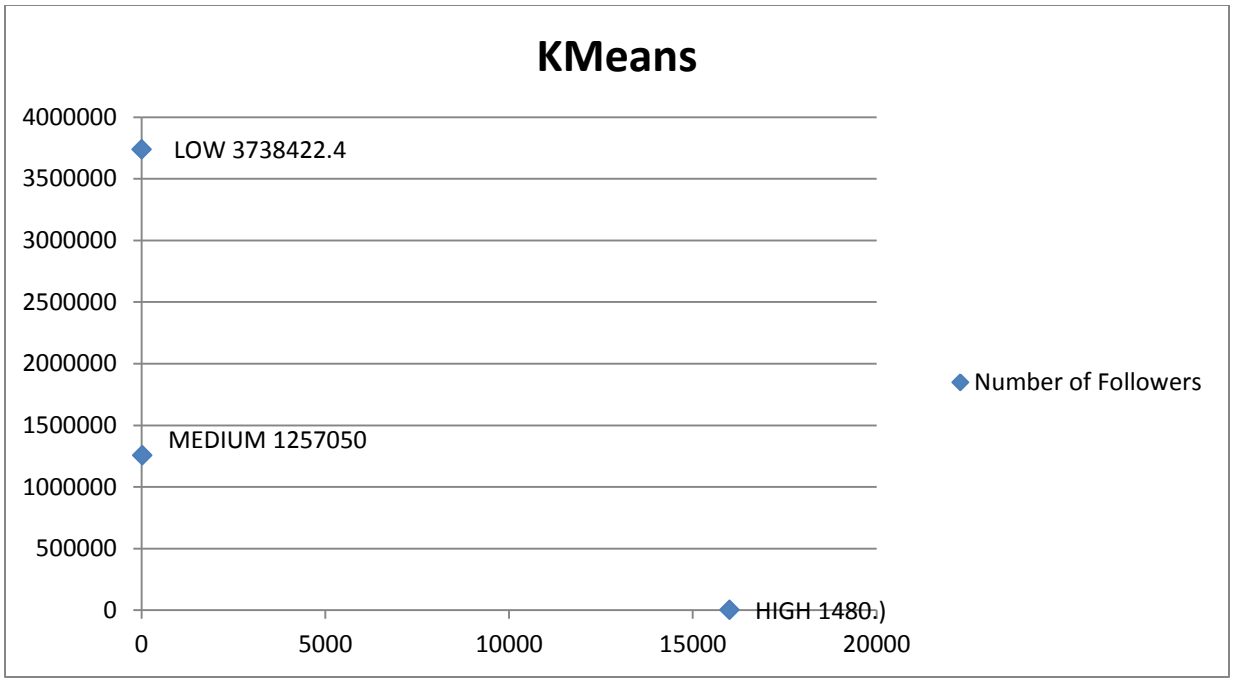
```
  Void reduce()
    { //collect data for each centre and calculate the average and store in the outut file
    }
}

Kmeansmain class
{
   void main()
    { counter=0
     while(counter<12)
       {//put the centerslist file in the distributed cache
        //run job
        //open outut file  and fetch the new centers
       //Update the centerslist file with the new values
      counter++;
      }
  }
}
```

Results obtained

The following final Kmeans were obtained in the centers file:

1480.6342307426119
1257050.0
3738422.4



The result shows:

- Center with value 1480.6342307426119 has an extremely high followers as compared to other centers
- There is not much difference between the number of followers in low and medium clusters

Shortest Path Algorithm (Dijsktra)

Problem

We want to find the minimum distance between a starting and ending node, when the distance between intermediate nodes is given

Approach

- *We provide the input file to our mapper to be mapped line by line*
- *We send three types of keys to the reducer*
- *If the line read from the input file is of the node itself, we send the distance so far from the origin, which is the middle value in the input file.*
- *If the line read from the input file is not of the node itself , we send the distance+1 so far from the origin, of all the adjacent nodes.*
- *Else we send the infinite distance.*
- *In Reducer, if any of the values is of infinite distance then we increment the counter by 1, which means the iteration has to repeat itself.*
- *In reducer we check if the value passed from the mapper is lesser than the infinite value, we write that value corresponding to that node in the output*
- *Next time when the iteration runs, the input to this iteration would be the output to previous iteration, and it will keep running untill no infinite values exist it the output.*

Pseudo Code

Result

1 0 2:
2 1 3:
3 2 2:13:
4 5 14:
5 5 14:
6 10 7:
7 9 6:8:17:
8 10 7:20:
9 13 10:
10 12 9:20:
11 5 12:
12 4 11:13:22:
13 3 3:12:14:
14 4 4:5:13:15:
15 5 14:16:

16 6 15:27:
17 8 7:27:
18 13 19:35:
19 12 18:20:21:
20 11 8:10:19:
21 13 19:
22 5 12:24:
23 8 32:
24 6 22:32:
25 8 26:32:
26 8 25:27:
27 7 16:17:26:28:33:
28 8 27:44:
29 17 37:38:
30 16 45:
31 8 32:
32 7 23:24:25:31:40:41:46:
33 8 27:42:43:
34 17 49:
35 14 18:50:
36 16 50:
37 18 29:
38 16 29:45:53:
39 16 45:
40 8 32:55:
41 8 32:47:
42 9 33:
43 9 33:44:48:
44 9 28:43:52:
45 15 30:38:39:61:
46 8 32:56:60:
47 9 41:
48 10 43:
49 16 34:50:71:
50 15 35:36:49:58:
51 17 58:
52 10 44:
53 17 38:62:
54 11 66:
55 9 40:
56 9 46:68:
57 13 43:69:
58 16 50:51:72:
59 13 69:
60 9 46:66:67:
61 14 45:65:
62 18 53:63:64:
63 19 62:

64 19 62:
65 13 61:80:
66 10 54:60:79:
67 10 60:
68 10 56:77:78:
69 12 57:59:77:
70 13 82:102:
71 17 49:72:
72 17 58:71:73:74:75:
73 18 72:
74 18 72:
75 17 72:76:104:
76 18 75:
77 11 68:69:82:98:
78 11 68:81:
79 11 66:80:81:
80 12 65:79:89:
81 12 78:79:95:
82 12 70:77:101:
83 18 85:
84 18 85:
85 17 83:84:86:
86 16 85:87:
87 15 86:88:
88 14 87:89:91:
89 13 80:88:90:
90 14 89:93:
91 15 88:92:
92 16 91:93:
93 15 90:92:94:
94 14 93:95:96:
95 13 81:94:
96 15 94:
97 14 99:
98 12 77:99:
99 13 97:98:100:
100 14 99:107:
101 13 82:105:
102 14 70:103:
103 15 102:104:105:
104 16 75:103:106:
105 14 101:103:107:
106 17 104:108:
107 15 100:105:
108 18 106:109:110:
109 19 108:
110 19 108: