Ubiquitous Search Engine

Pooja Vyas
Department of IT
SIES GST
Navi Mumbai, India

Archith Menon
Department of IT
SIES GST
Navi Mumbai, India

Aditya Ravindran
Department of IT
SIES GST
Navi Mumbai, India

Samit Shivadekar
Professor,Department
of IT SIES GST
Navi Mumbai, India

ABSTRACT

Ubiquitous Search engine is a non-conventional search engine. It is built with an intended function of finding the most influential node in a network or given data set. The objective is to find centers of influence in social networks. It can be used as a tool for data mining and analyzing it further for optimum use of the user's benefit. The system is implemented using Hadoop and Big Data. It aims at increasing the performance of the system and rendering results in fastest possible way by implementing suitable algorithm for the same. Hadoop is used to support parallel computing whi0ch provides a base for simultaneous search on multiple machines. Big data is a large amount of data which can be analyzed and converted into useful information. The data set taken for this project is of 'Twitter', a micro blogging website as it uses a follower relationship rather than friend concept.

General Terms

Data Analysis, Search engine

Keywords

Parallel Computing, Hadoop, MapReduce, Big data, Twitter, Influence

1. INTRODUCTION

There is a theory that Kevin Bacon is the most influential person in Hollywood since any individual involved in the Hollywood film industry can be linked through his or her film roles to Kevin Bacon within six steps. The aim is to find the most influential people - the Kevin Bacon(s) of Twitter. Our objective is to find the centers of influence in a social network.

A well designed search engine to find the most relevant result to any query and well analyzed influential factors in the data set is required. Using the parallel and distributed framework of Hadoop and the principle of Centrality theory, this goal can be achieved.

Analysis was performed on 1.7 billion Twitter [7] follower relationship to find the most influential node.

2. EXISTING SYSTEM

The current search engines for data analysis make use of singular search which increases the user waiting time. Specially, if the search is to be performed from a large data set, it might lead to slower speed and longer waiting time for the user. Also, finding the relevant data from a large amount of data proves to be tedious task and reduces the overall system efficiency. Thus the main problems that need to be solved are:

1. Search speed
2. Relevance of results
3. Search from tetra bytes of data.

3. NEED FOR PROJECT

The project acts as a tool for data mining and acts as a pioneer in analyzing social network relationships. It is best suited for digital marketing. Analyzing billions relationships and finding the most accurate result for influence of each node is a very time consuming and difficult task and hence, Ubiquitous search engine provides an optimum solution for the same. The need for accurate and relevant results and faster speed is the baseline of this project

4. PROPOSED SYSTEM

The Proposed system is to find the most influential center in social network using Hadoop for parallel computing and faster results and Big data to increase the accuracy and relevance of data results. The user waiting time is reduced by approximate 8 times, making the system one of the most suitable one to use. As the analysis is performed parallel on multiple devices, the amount of data that can be analyzed is increased drastically. The system uses MapReduce function of Hadoop for its implementation. The function being one of major features of Hadoop is easy to implement. The complexity of the system is reduced.

4.1 Algorithm

1) Initialize the nodes.

   For this system, all the nodes are initialized to 1.

2) Now, for each node A_i, calculate its influence on all the nodes that it follows in 3 steps:-

   2.1) Count = number of nodes that A_i is following.

   2.2) Influence of A_i on all its direct followers = Influence of A_i/Count

   2.3) Send this value of A_i to all its direct neighbors

3) Each node receives the influence values from all its followers and calculates its own influence value.

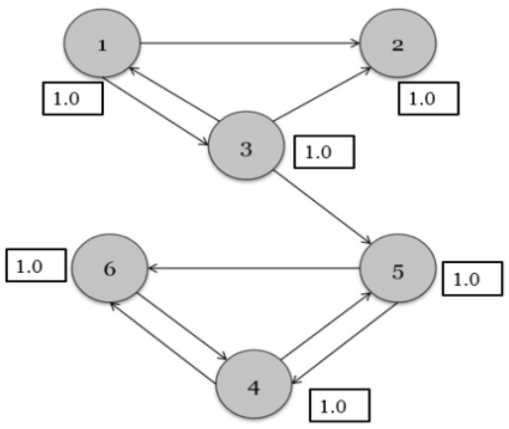


Fig 1. Initialize the nodes

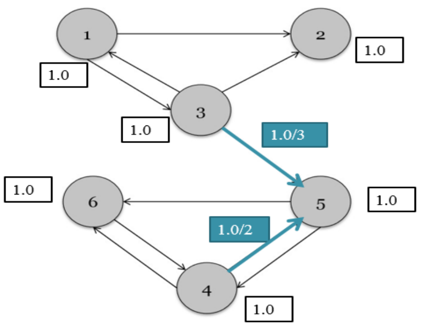
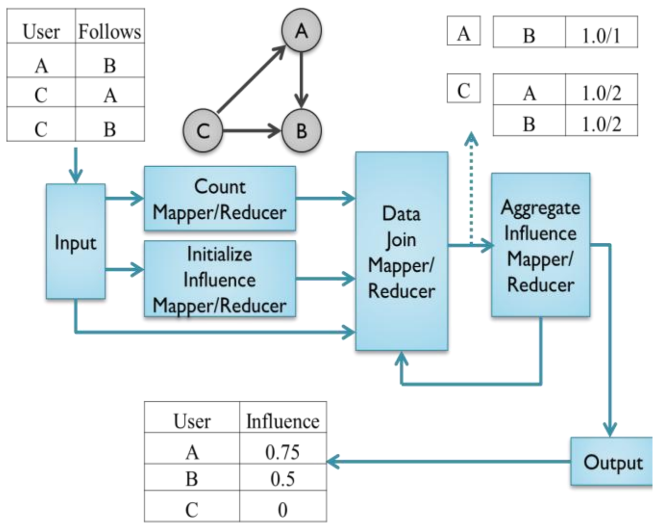


Fig 2. Count influence

4.2 Working

The input data is provided in a format of large text file by specifying the path of the file. The data is stored in format of followed and follower. The system inverts the input. The format now changes to follower - followed. The Mapper class then divides the data into parts. The influence count is calculated and each node calculates its own value on basis of number of iterations involved. On basis of influence factor, most influential node is found and stored in an output folder created by the system.



User	Follows
A	B
C	A
C	B

A	B	1.0/1
C	A	1.0/2
	B	1.0/2

User	Influence
A	0.75
B	0.5
C	0

A
B
C
Input
Count Mapper/Reducer
Initialize Influence Mapper/Reducer
Data Join Mapper/ Reducer
Aggregate Influence Mapper/ Reducer
Output

Fig 3. Implementation

5. FEATURES

- Parallel computing using Hadoop on multiple devices.
- Analysis of 1.7 billion relationships in minimum possible time
- Highly scalable
- Better Speed and accuracy
- Low cost for implementation.
- Easy to use for users

6. PRINCIPLES

The system uses 2 main principles for its implementation:

6.1 Centrality Theory

Centrality theory is the fundamental principle used to search the influential node. Centrality can be used to measure the traffic flow in any given system. This theory lays out a typology of network flows based on two dimensions of variation, namely the kinds of trajectories that traffic may follow (geodesics, paths, trails, or walks) and the method of spread (broadcast, serial replication, or transfer). [3]

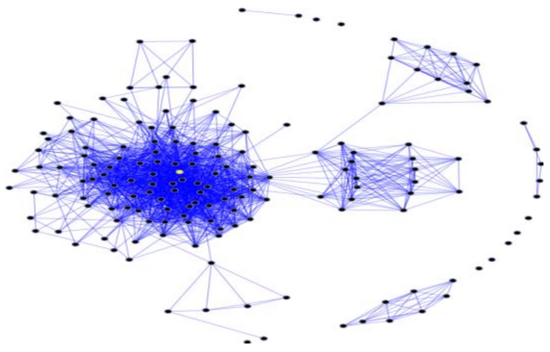


Fig 4. Centrality Theory

The above diagram explains the concept of centrality theory. The dark blue region indicates the most crowded and central node of the network.[3]

6.2 MapReduce

Hadoop MapReduce is a software framework for easily writing applications which process vast amounts of data (multi-terabyte data-sets) in-parallel on large clusters (thousands of nodes) of commodity hardware in a reliable, fault-tolerant manner. A MapReduce job usually splits the input data-set into independent chunks which are processed by the map tasks in a completely parallel manner. The framework sorts the outputs of the maps, which are then input to the reduce tasks. Typically both the input and the output of the job are stored in a file-system. The framework takes care of scheduling tasks, monitoring them and re-executes the failed tasks. Typically the compute nodes and the storage nodes are the same, that is, the MapReduce framework and the Hadoop Distributed File System are running on the same set of nodes. This configuration allows the framework to effectively schedule tasks on the nodes where data is already present, resulting in very high aggregate bandwidth across the cluster. The MapReduce framework consists of a single master

JobTracker and one slave TaskTracker per cluster-node. The master is responsible for scheduling the jobs' component tasks on the slaves, monitoring them and re-executing the failed tasks. The slaves execute the tasks as directed by the master. Minimally, applications specify the input/output locations and supply map and reduce functions via implementations of appropriate interfaces and/or abstract-classes. These, and other job parameters, comprise the job configuration. The Hadoop job client then submits the job (jar/executable etc.) and configuration to the JobTracker which then assumes the responsibility of distributing the software/configuration to the slaves, scheduling tasks and monitoring them, providing status and diagnostic information to the job-client. [1], [2]

Inputs and Outputs:

The MapReduce framework operates exclusively on <key, value> pairs, that is, the framework views the input to the job as a set of <key, value> pairs and produces a set of <key, value> pairs as the output of the job, conceivably of different types.[1] ,[2]

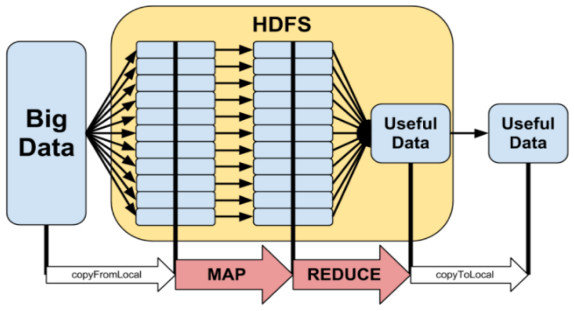


Fig 5. MapReduce Flow

7. APPLICATIONS

There are various applications of finding the centers of influence in a given network:-

7.1 Digital marketing

Major application of this is to find how influential a person is in a social network. Finding an individual who can influence the most number of people is an integral part of directed advertising and brand building. This data analysis can help organizations plan their digital marketing strategies accordingly. Planned strategies can help avoid financial loss and increase sales and revenue of an organization.

7.2 Traffic management

The concept can even be extended to real networks such as a road transport network. Finding roads or highways with maximum traffic can be found using this system. Government can plan traffic management system according to the output obtained. This can help reduce traffic congestion.

7.3 Centers of relevance

Search engines find the most relevant pages for a given search query. By defining the relative relevance between pages based on the relevance of pages that link to it, the problem can be translated to finding the centers of "relevance" over the internet. In fact, this is exactly how Google's PageRank algorithm works.

7.4 Organization strategy plan

Various multinational organizations have their branches and offices worldwide. With Ubiquitous Search Engine, branch with maximum sale and influence can be calculated. Also, product with maximum sale in a branch can be calculated and strategies can be planned accordingly for increasing revenues.

8. RESULT:

Ubiquitous Search Engine is developed right now using hadoop single cluster. The analysis is currently being performed on few inputs to get output and verify the system.

The expected output is analysis of complete dataset.

```
12      13
12      14
12      15
12      16
13      20
13      25
17      12
17      13
21      11
18      17
```

Fig 6.Input file

```
0.3723077       13
0.95438343      18
1.2795084       12
2.5565271       17
```

Fig 7.Output with 2 runs

```
0.3723077       13
0.69057816      12
1.1641089       17
1.9970908       18
```

Fig 8.Output with 3 runs

Similarly, Depending on number of relationships and number of runs, output can be achieved as needed.

The table below shows analysis performed on the system.

Table 1.Analysis of relationships

No. of relations	No. of runs	Time taken(in seconds)
10	1	18
10	2	30
1113854	1	85
1113854	2	107

Here it can be noticed that even if there is exponential increase in the size of relation, the time taken hasn't increased in an exponential way. Thus, providing the proof of system's efficiency.

9. CONCLUSION

In conclusion, based on our experiments, our implementation not only works correctly but even scales very close to our expectations with increase in data, number of nodes and iterations. Exploring avenues for performance optimization, especially for higher number of compute nodes is part of our future research.

10. ACKNOWLEDGMENTS

The author wishes to thank contributors of the previous research work and also thank family members for their support throughout. Humble gratitude towards teachers and project guide for their inputs and support.

11. REFERENCES

[1] MapReduce Tutorial https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html

[2] MapReduce Wikipedia Page, http://en.wikipedia.org/wiki/MapReduce

[3] Centrality and network flow, by Stephen P. Borgatti (Department of Organization Studies, Boston College, Carroll School of Management, Chestnut Hill, MA 02467, USA).

[4] Inverted Indexing In Big Data Using Hadoop Multiple Node Cluster, by Kaushik Velusamy, Nivetha Vijayaraju, Greeshma Suresh, Deepthi Venkitaramanan, Divya Madhu (published in (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 4, No. 11, 2013).

[5] Hadoop Official website,http://hadoop.apache.org/.

[6] The PageRank Citation Ranking: Bringing Order to the Web http://ilpubs.stanford.edu:8090/422/1/1999-66.pdf

[7] Twitter official website.

[8] Challenges and opportunities with Big data, by Alexandros Labrinidis , H. V. Jagadish.

[9] Hanneman, R. A., & Riddle, M. (2005). Introduction to Social Network Methods. Retrieved from http://faculty.ucr.edu/~hanneman/nettext/

[10] Koschützki, Dirk; Katharina A. Lehmann; Leon Peeters; Stefan Richter; Dagmar Tenfelde-Podehl; Oliver Zlotowski (2005). "Centrality Indices". In Ulrik Brandes, Thomas Erlebach. Network Analysis – Methodological Foundations. LNCS 3418. Springer Verlag, Heidelberg, Germany. pp. 16–60. ISBN 978-3-540-24979-5.4

[11] Aggarwal, C. C. (2011). Social Network Data Analysis. New York, NY: Springer