Degree Of Separation
CS 425 Course Project Report

Group 7:
M. Gokhan Simsek
E. Berker Senol
Orhun Caglayan

December 21, 2016

Instructor: M. Mustafa Ozdal

1 Project Description

Computing the average degree of separation in the actor list taken from the IMDB database.(approx. 2GB data) Aim is to find average degree of separation of IMDB actors and actresses graph using some approximation algorithms.

1.1 What is six degrees of separation?

The definition was found by a Hungarian author Frigyes Karinthy in 1929. [1] The theory of six degrees of separation states that any two randomly selected people on this world can get to know each other by no more than six steps of intermediate friend chains.[2] After checking 30 billion electronic messages, Microsoft researchers say the theory stands up. [3]

1.2 Facebook Example

Facebook tried to find the average degree of separation on its huge graph consisting 1.6 billion users. Running BFS on every node and computing exact average of degree of separation is quite impossible on such data even with very powerful machine and parallelism because algorithm runs in quadratic time. Therefore they used an algorithm similar to HyperANF algorithm and found the average degree of separation as 3.57.[4]

[1] $en.wikipedia.org/wiki/Six_degrees_of_separation$
[2] $journal.webscience.org/147/2/websci09_submission_49.pdf$
[3] $theguardian.com/technology/2008/aug/03/internet.email$
[4] $research.fb.com/three-and-a-half-degrees-of-separation/$

2 IMDB Dataset

The dataset covers the actors and actresses in two different files, each of them is approximately 1 GB. The dataset has the format that is shown in the figure below.

```
Sanders, Bernie     #UNDER35POTUS (2016) (V)  [Himself]
            American Blackout (2006)  [Himself]
            Broadcast Blues (2009)  [Himself]
            Capitalism: A Love Story (2009)  [Himself]
            CNN Democratic Primary Debate (2015) (TV)  (as Sen. Bernie Sanders)  [Himself - Candidate]
            Courting Des Moines (2016)  [Himself]
            Democratic Candidates Debate (2016) (TV)  [Himself - Candidate]  <3>
            Denial (2016/IV)  [Himself]
            Ethos (2011/I)  (archive footage)  [Himself - US Congress]  <29>
            Fall of the Republic: The Presidency of Barack H. Obama (2009)  (archive footage)  [Himself - Senator]
            First in the South Democratic Candidates Forum on MSNBC (2015) (TV)  [Himself - Candidate]  <2>
            Heist: Who Stole the American Dream? (2011)  [Himself]
            Koch Brothers Exposed (2012)  [Himself - United States Senator for Vermont]  <4>
            Longshot... The Biopic of Senator Bernie Sanders Campaign 2016 for POTUS (2016)  [Himself]
```

Figure 1: Sample from IMDB dataset

2.1 Problems with the dataset

There were 3 main problems about IMDB dataset for our purpose:

a. Each episode of a TV series are indexed separately. Instead of considering each episode of a TV series a separate production, we will consider each TV series as one production. Therefore, TV series episodes should be merged.

b. Actor and actress datasets are separate that needs to be merged.

c. The TV shows where actors appear as themselves, represented with [Himself] or [Themselves] in the datasets that needs to be removed. (See Figure 1)

d. Adult movie actors have extensive amount of movies. For example there are actors that have more than 1,700 movies.

2.2 Solutions to the problems

We used AWK language to filter the data. Awk is very efficient for processing huge text files like IMDB datasets. Size pf Actor dataset was 1.17 GB before filtering After filtering the dataset looks like in Figure 2.

3 Creating the Graph

Due to huge and sparse network, we wanted to create graph in list format. We implemented 2 Map Reduce steps on our filtered dataset to create our graph. In

```
Acaibe, Jo?o Batista , Atrav?s da Janela (2000)
Acain, Al , Parent Trap: Hawaiian Honeymoon (1989),"Jake and the Fatman" (1987)
Acajou , Overboard (1978)
Acajueiro, Luis , Amor! (1994)
Acal, Miguel , "Andaluc?a, un siglo de fascinaci?n" (1996)
Acal, Robin , Kiss Kiss (2017)
Acala, Butch , Magtago ka o lumaban (1996)
Acaldo, Morgan , Magic Power Scouts (1998)
Acaley, B.J. , The Secret Game (2008)
Acama, Val , "Mix Master" (2006),"Mix Master" (2006)
Acame, Benya , Operasiones espesiales (2014)
Acampa, Mario , Accetta il consiglio (2010),Time - Ancora Sessanta Secondi (2006),"Non uccidere" (2015)
Acampado, Jun , Pandanggo (2006)
```

Figure 2: Filtered dataset

order to reduce the total memory, we indexed each actor with an integer after first map reduce step. We ran our map reduce programs on our rented Digital Ocean cloud Ubuntu server with 16 GB memory.

- The first map reduce takes the filtered data and reverses the dataset as:

  ActorId1 MovieName1 MovieName2 MovieName3
  ...
  to
  MovieName1 ActorId1 ActorId12 ActorId34
  ...

- The second map reduce input is the output of first map reduce and it creates a (key, value) pair for each actor pair in the same line (movie). In practice this approach resulted into data loss because of huge number of (key, value) pairs. Our server's memory was not able to handle this approach. We changed our approach into adding each friend of actor into a string and emitting just one (key,value) pair for each actor in each line. This approach was much better in terms of memory and our server was able to handle the program. But if two actors played in two or more different movies, our approach created duplicate friends, in order to avoid that we programmed a Python script and removed the duplicates. At the end the desired graph was created as shown in Figure 3. This graph is taking more than 8GB of memory.

```
1000000 488371,835578,51851,1748560,1754456,2011003,1493888,1120819,1486425,835578,
1000001 1387150,1973582,1509555,1666346,1613417,852907,917111,1018662,14271,142223,298999,
1000002 2444107,2417696,2396303,2319424,2384203,2199384,2403085,1180744,1346292,1187308,1423937,
1000003 2287584,2196722,2454488,2337579,2488721,2476110,2323400,2378481,2395302,2206907,572406,6
1000004 1800194,1577484,1880830,1822304,1588853,1607692,2178881,1978790,2132403,2079210,1986447,
1000005 2437157,2362905,2193040,2463980,2244815,2255081,2285277,2313064,2365117,2200886,2316223,
1000006 384137,194887,68123,87669,406370,72179,397656,576830,694882,527881,682132,1061360,857293
1000007 396623,943219,2439317,2398375,2279005,
1000008 2406082,2266087,2432237,572430,722291,718176,615913,460245,1081002,872328,192958,90027,4
1000009 1000007,
```

Figure 3: Actors' graph

4 Algorithms

First we tried to run BFS on the graph to find the exact average degree of separation. Running BFS on the graph is very costly even for just one node because the graph has more than 2.2M nodes. Therefore, we used an algorithm to approximate the average degree of seperation

4.1 Breadth First Search

Breadth First search is quadratic operation and during our implementation we observed that BFS for even one node is not scalable for huge graphs like ours. Therefore we could not run BFS algorithm on our server or our computers.

4.2 Watts and Strogatz Model

The WattsStrogatz model is a random graph generation model that produces graphs with small-world properties and the model supports the IMDB data in this sense.[5]

Watts and Strogatz showed that the average path length between two nodes in a random network is equal to ln N/ln K where N = total nodes and K = acquaintances per node.

The number of actors in April 1997 IMDB dataset was 225,226 and the average acquaintances per node was 61 and they found the average degree of seperation as 2.99.[6]

We found number of nodes as 2,210,838 and average acquaintances as 507, therefore we found the average degree of seperation as 2.34.

4.3 ANF

ANF is said to quickly answer a number of interesting questions on graph-represented data and we implemented ANF algorithm to approximate average degree of separation in IMDB graph. The simple pseudo code of algorithm is:

M(x, 0) = x for all x V
FOR each distance h DO
    M(x, h) = M(x, h − 1) for all x V
    FOR each edge (x, y) DO
        M(x, h) = M(x, h) M(y, h 1)

Unfortunately, this algorithm failed on large scale (with 2.2M nodes) because of its memory requirement so we ran the algorithm on samples of our graph.

[5] $en.wikipedia.org/wiki/Watts_and_Strogatz_model$
[6] $www.nature.com/nature/journal/v393/n6684/full/393440a0.html$

Table 1: First Sample: Actors with more than 50,000 friends for h=2

	Time	Result
ANF	17 min 30 sec	1.54947
Hyper ANF	2 min 28 sec	1.549160

Table 2: Second Sample: Actors with more than 60,000 friends for h=2

	Time	Result
ANF	10 sec	1.04286
Hyper ANF	2 min 10 sec	1.00240

4.4 Hyper ANF

Hyper ANF is an efficient algorithm which uses HyperLogLog counters in order to estimate number of friends of a vertex. Basically, instead of unioning all friends' friend sets, hyperanf unions hyperloglog counters of each friend. For this algorithm, we implemented the HyperANF function

5 Conclusion and results

- We found Hyper ANF much more faster than ANF.
- To process such a graph in a regular machine, there needs to be an algorithm with run time less than O(N) because even reading each element in the graph takes more than 30 minutes in our machines. For any other algorithm, we need machines with extensive RAM or parallelism between regular machines. We could not offer parallelism because the number of server that we could rent was at most 10.
- Creating a sample from the IMDB data was difficult because the more average friend number for a graph is, the more connected it is. Therefore we found the average degree of separation of our samples very low.
- ANF is more efficient in smaller graphs since declaring numerous HyperLogLog counters in HyperAnf is not time efficient although it is linear time operation.
- Increasing numbere of average number of friends decreases degree of seperation, which is also intuitive since fully connected graphs have degree of separation of 1.
- A Graph derived from a sample actor list has 25534 edges. HyperLogLog counter estimates number of edges as 24535.9 which is %3 near estimation. Therefore it can be concluded that HyperLogLog counters are reliable for graph estimations.