Social Network Analysis at Scale in Cloud

Project Proposal

Xiaolan Cai

1.Problem statement

Social network analysis has drawn a lot of attention in the past decades. Initially sociologist have started to study how social networks is formed and how people are connected. Nowadays social networks like Facebook, Twitter is used to spread information like news, ideas. And also for marketing purpose, for example we may also want to study how to maximize the influence of advertisement by choosing proper targets. Social networks normally can be modeled as graph structured models, with vertices/node representing an individual and edge representing the relationship between individuals.

Meantime big scale data mining tools is also well developed, and distributed computing is significantly improving the speed of data mining by parallely executing subtasks divided from one single big problem with distributed systems in scalable ways. And social network analysis normally includes massive amount of graph-structured data to processing, and how to leverage scalable distributed computing work to handle graph-structured data is indeed an interesting topic.

In this work we are interested in a specific problem of SNA, called influence maximization [1], and we want use currently state-of-art distributed

computing tools to examine how well we can process the graph-structured data in terms of quality and speed.

Maximum influence in Social network is an interesting topic of SNA (social network analysis). In [1] this problem is formally defined. Given a Graph N (V, E), and choose an ignition k number of vertices set S, what is the maximum number of influence we can achieve? In [1], two Basic Diffusion Models are considered specifically Linear Threshold and Independent Cascade models. In this work, we consider only Independent cascade models for simplicity. In this model, initially start with a set of active nodes A_0, in each step an active nodes v will have a chance to flip his any inactive neighbor w into active one with probability p_{vw}. In the whole process, node v only have a single chance to activate any of his inactive node w, meaning he cannot make any further attempts no matter he is successful in flip w or not. And the process stops when there is no mare activation are possible [1].

Finding the optimal solution is NP-hard problem, and in [1], an approximation solution is proposed, it uses greedy algorithm to achieve slightly better than 63% of the optimal solution. However, the computation is still pretty heavy, because they need to run Monte Carlo simulation in each update step.

It is a challenging task to process massive data not even to say graph structured data. There are two kinds of graph processing frameworks. **GraphLab[7]**: which shares state in parallel vertex computation. And **Pregel[8]**: which is used for passage passing.

We need to modify existing algorithm to make it suitable for vertex programing, which is requested by most graph based commutating framework. We provide a vertex program that is run by each vertex in parallel.

So our question is how the influence maximization problem can be processed in cloud computing environment with scalable setups? What performance need to be sacrificed in terms of accuracy to archive speed? (what's the trade off between quality and speed?)

2.Solutions for processing

One option is to use Spark graphx which uses message passing between vertices (e.g. Pregel). We first store our data in HDFS and use graphx to build graphs by creating vertices and edges from it as normal spark programing do, and map to process the source data and then do targeted programing computing on individual graph objects.

Another option is to combine spark with graph based database such as Neo4j. We can import the dataset into Neo4j first. Then we need distribute the data in Neo4j to HDFS, after that Spark will do parallel computing and put results in HDFS followed by exporting back to Neo4j.

Besides message passing based graph computing framework, we can also use state shared tool, such as graphlab, we can also explore this tool for comparing.

Another orthogonal method of improving the speed of computing is edge sampling, for example we only delete some of the edges of a vertex that have high degrees to reduce the computation with approximation [3].

3. Setups and Datasets:

We will setup our experiment in Cloudlab as it is free of charge and AWS is charging for EMR resources.

As to dataset, we will use undirected social network in snap datasets, such as “ego-facebook”, or “gemsec-facebook” [2]. And the data comprises lists of pairs of node-ids, which represents edges of a graph. The data file is in txt or csv format. We will store the data in HDFS in our Cloudlab setups. For preprocessing we will need to assign a random probability on each edge as the probability that a vertex can flip the corresponding neighbor vertex.

4. Timelines of the project:

We can divided our project in following phases

Phase 1 (1 week)
We will setup the experiment in Cloudlab, including Spark graphx alone setup and Neo4j+spark setup.

Phase 2 (1.5 weeks)
We will implement the codes in Scala and measure Spark alone setup.

Phase 3 (1.5 weeks)
We will implement the codes and measure Neo4j + Spark setup, graph data analyze and visualization.

Phase 4 (1 week)
Report write up.

5. Challenges:

To adapt the influence maximization problem for Spark-liked framework is difficult, as we need to defined vertex based programing for parallel with approximation and not to sacrifice the accuracy too much.

Issues:

- Time constraint, since I am doing this project alone, I have concern that I may not able to finish all the measurement I have planned to, but at least two feasible scalable setup should be build and measured.
- Spark graphx use Scala for programing and I need to learn Scala for this.

Reference:

[1]. Maximizing the Spread of Influence through a Social Network
[2]. https://snap.stanford.edu/data/
[3]. Mark S. Handcock, David R. Hunter, Carter T. Butts, Steven M. Goodreau, and Martina Morris (2003). statnet: Software tools for the Statistical Modeling of Network Data. URL http://statnetproject.org
[4]. Bridging the GAP: Towards Approximate Graph Analytics
[5].https://medium.com/mobileforgood/social-network-analysis-using-apache-spark-and-neo4j-1ccba3c8af9a
[6].https://www.slideshare.net/GhulamImaduddin1/social-network-analysis-with-spark
[7].https://en.wikipedia.org/wiki/GraphLab
[8].https://kowshik.github.io/JPregel/pregel_paper.pdf

Check Point 1

1.Change of Proposal:

There is no big change from the original proposal. I will add graph data visualization for the future work after I have finished the analysis on the data. Because good visualization can make the work more easy to understand for readers. The question raised in the proposal is answered in highlight: what is the problem stated in the paper.

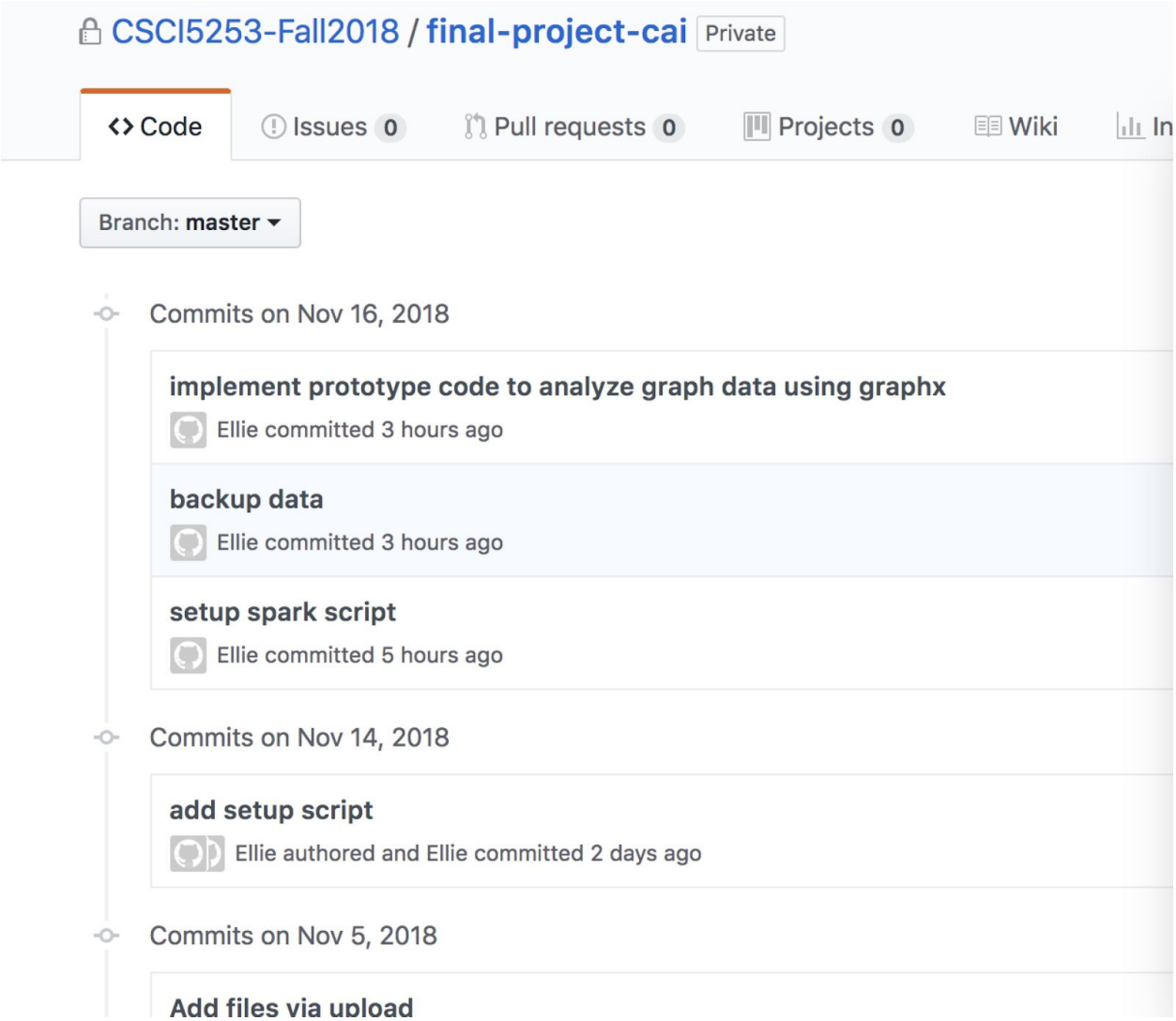
2. Work has been done:

- I wrote a script to setup working environment in Cloudlab automatically. I met several problems when setting up Hadoop 2.7 and Spark 2.4 on Cloudlab. Because it is bare-metal Linux machine, there is not firewall or network security setup. I found the environment is not stable and being attacked frequently. Then I figure out the default Hadoop cluster doesn't require mandatory authorization in the configuration:

```
<name>hadoop.security.authentication</name>
<value>simple</value>
<source>core-default.xml</source>
```

  That's probably why if I don't configure the security mode, it can be easily accessed by anonymous user. There is certificating system called Kerberos that is used to identify if the user authorization is legal. And I can also change port number of resource manager which is originally 8080 to prevent being hacked.

- The GraphX code is successfully running on Cloudlab and is tested with sample code. I have learned graphX programing in Scala and started with graphx example to analyze sample facebook graph data from SNAP with socialgraph.pregel model (msg based communication).



3. Work to be done:

- Continue to learn about Spark GraphX and its toolset.

- For the next step I will use the processed large dataset for my experiment. I will implement the code to use GraphX with hdfs and GraphX with Neo4j on the same data for comparison. I will visualize the result with charts or D3.js for better knowledge.

- Use GraphX to realize the complex greedy algorithm in[1] to compute node influences and analyze how to maximize the influence.

4. Cost:

Since I don't have much AWS free quota left, I setup Spark environment on Cloudlab and it is free to use.

5. Dataset Management:

I use the ego-Facebook undirected graph data in SNAP for this project. I have studied the data files first since the data is not straightforward to read on the first sight. The data files consisted of 5 kinds of files.

nodeId.edges : The edges in the ego network for the node 'nodeId'. Edges are undirected for facebook, and directed (a follows b) for twitter and gplus. The 'ego' node does not appear, but it is assumed that they follow every node id that appears in this file.
nodeId.circles : The set of circles for the ego node. Each line contains one circle, consisting of a series of node ids. The first entry in each line is the name of the circle.
nodeId.feat : The features for each of the nodes that appears in the edge file.

nodeId.egofeat : The features for the ego user.
nodeId.featnames : The names of each of the feature dimensions. Features are '1' if the user has this property in their profile, and '0' otherwise. This file has been anonymized for facebook users, since the names of the features would reveal private data.

The dataset files of my interest is nodeId.edges. The graph data is represented in table view and I found that undirected graph structure may include duplicate data such as edge 1234 5678 is same as edge 5678 1234. So I have to do data cleaning and remove the duplicated data.

Check Point 2

1.Group Meeting

There is no group meetings since I am working solo.

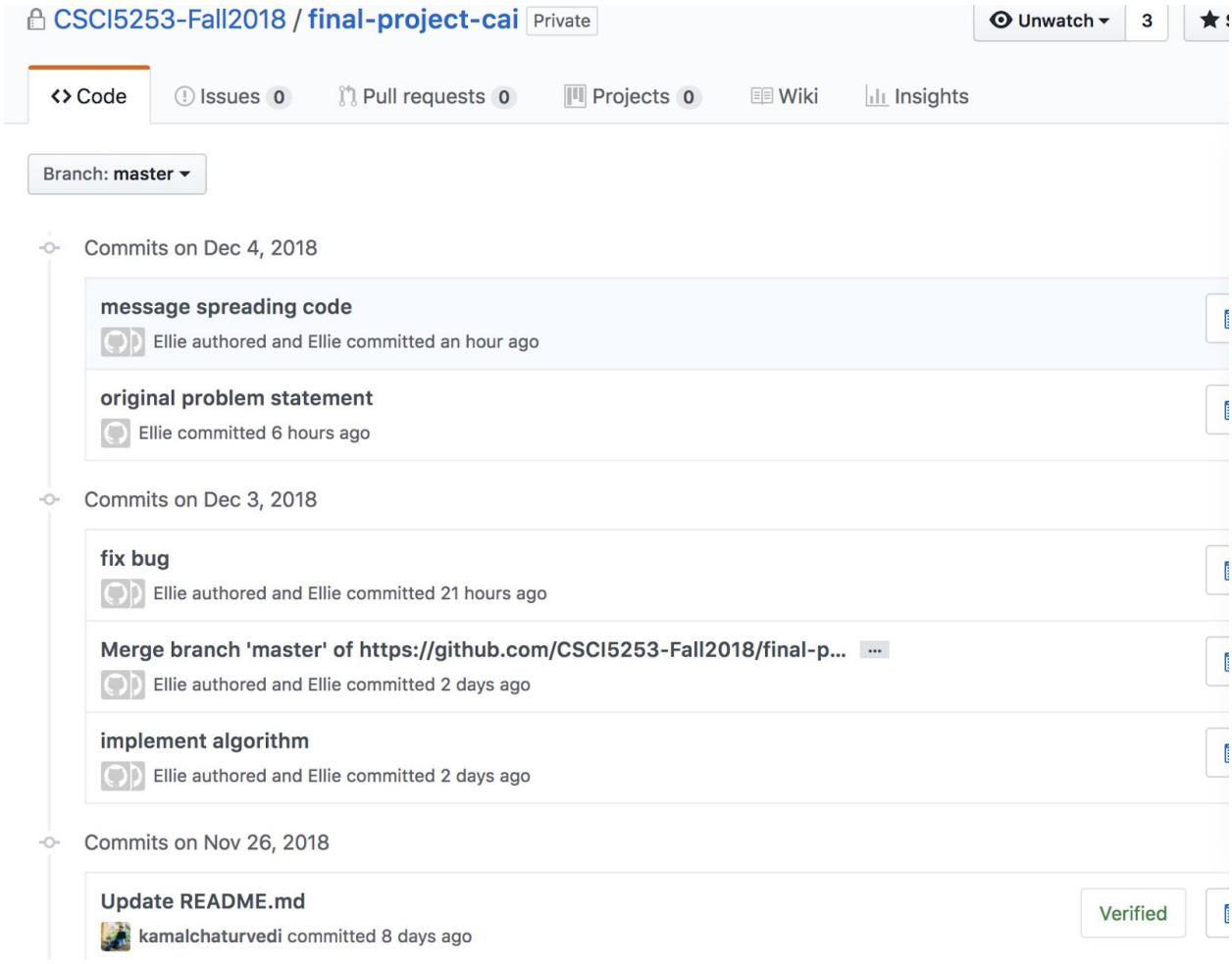
2.Change of Proposal

There is no major change on the proposal. But as I go deep into the project, I learned more and have some new thoughts. While I do research about the algorithm, I found there are several other influence maximization algorithm that is popularly used such as Page Rank, an algorithm developed by Google and Connect Component, etc. These can also be implemented using Spark's Graphx.

If time allowed I would implement one of these algorithm to compare its performance on large scale graph data with the greedy algorithm in paper "Maximizing the spread of influence through a social network", as well as the running time.

3.Work has been done

- Setting up cluster nodes: 4 physical nodes. 1 Master node, 1 resource manager and 2 slave nodes. Each node has 4 CPU cores, 15 GB of memory and 160 GB of SSD storage.

- Finished implementing the IC (independent cascade) model with Pregel API of GraphX (vertexProgram, sendMessage, messageCombiner): IC model is to simulate the influence spread across the networks. In the model, a propagation probability is picked, and a random number will be picked for each node if it is in active list. In the vertexProgram, when a message received in a vertex, and the random number is greater than the propagation probability it will be flipped to active status for each iteration of algorithm. After many iterations, all the vertices that can be active will be added to the active list.
- 

4.Work to be done

- Finish implementing greedy algorithm in Scala.
- Setup Neo4j database to compare with HDFS.
- Will implement Page Rank or Connected Component algorithm with the same IC model.
- Get performance results.
- Write up the final report.

5.Cost

There is no extra cost for this project since I am setting up experiments with CloudLab.us.

6.Dataset Management

The dataset files of my interest is nodeId.edges. The graph data is represented in table view and I found that undirected graph structure may include duplicate data such as edge 1234 5678 is same as edge 5678 1234. So I have to do data cleaning and remove the duplicated data. I will implement a package in Scala to do the transfer of directed graph to undirected graph. I will use Neo4j to store the data.