Graph Mining Using SQL

Ye Zhou
School of Computer Science
CMU
yezhou@cs.cmu.edu

Jin Hu
School of Computer Science
CMU
jinh@cs.cmu.edu

November 6, 2014

Abstract

In this project, we investigate the possibility of using only SQL to do graph manipulations on *graphMiner*. For the first stage, we implement kcore algorithms within *graphMiner* framework and have unit test on kcore and other already implemented algorithms in the system to gain better understanding of methods like kcore, PageRank, Eigenvalue computation. We also test the algorithms with wiki-Vote and soc-Epinions1 dataset from SNAP.

1 Introduction

Specify the problem; Give the motivation; List your main contributions

The problem we are trying to solve is: Given a graph of source-destination pairs on disk, we try to use only SQL to implement basic matrix and vector operations, which make it possible to handle more complex graph manipulations. We will then be able to implement algorithms like PageRank and Degree distribution without the need of an additional language.

The motivation is that we can use only SQL and do not need to depend on other languages. With SQL we can enjoy the benefits of various features of databases and become more effcient in implementing and testing some of the algorithms without the worries like IO and file read and write, etc.

This is an important problem, because graph manipulation and mining has many real world applications and is a very important topic in research community. Thus a convenient way of implementing graph manipulation algorithms are highly desired.

The contributions of this project are the following:

- Our implemented *K-core Decomposition* is fast and uses only SQL to do the calculations. Python only works as some basic loop control and input output format handling.
- We tested various graph handling algorithm on unit tests and some larger dataset like wiki-Vote and soc-Epinion1 from SNAP.

2 Survey

Next we list the papers that each member read, along with their summary and critique.

2.1 Papers read by Ye Zhou

The first paper was the Pegasus paper by U Kang

- *Main idea*: When graph data grows larger and larger, using traditional graph mining algorithms is difficult to deal with trend. In order to solve the graph mining problems with several Petabytes data, Pegasus is the first such library, impelemted on the top of Hadoop platform. In the paper, first they try to find out the common operation, which is the matrix-vector multipilication, underlying several primitive graph mining operations. They call it GIM-V. As GIM-V is so important, they successfully proposed several optimizations, and got more than 5 times faster performance. They also took real big data graph into Pegasus to get the mining result, which revealed important patterns. This showed the succuess of Pegasus with large data graph mininig as the graph data they used had never been studied before.
- *Use for our project*: It showed that with large data, we always have new problems using traditional graph mining methods. It is really hard to say that with SQL, we can do enough with graph mining now, as SQL is based on RMDB. But with more and more new mature products/framework like hive, pig, shark which support traditional SQL operations on big data and No-SQL database, we can do more with SQL.
- *Shortcomings*: PEGASUS focus on large graph querying/mining and most of the job was focused on how to compute fast, but it ignored the storage part which can also take effect to improve the performance like indexing. In addition, PEGASUS essentially perform node/vertex-centralized computation but cannot supports edge-centralized processing like induced subgraphs. Finally, hadoop is not so efficient for iterative calculation, as everytime it needs to write output data to hard disk. Spark has better performance as it output its intermediate data in memory and can even cahche the data in memory.

2.2 Papers read by Ye Zhou

The second paper was the Spectral Analysis for Billion-Scale Graphs paper by U Kang

- *Main idea*: The paper proposed HEIGEN algorithm which is designed to be accurate, efficient to run on highly scalable hadoop environment and solve the problems that will calculate out the spectral value. The paper first showed specific observation using HEIGEN with real world data on billion-scale graphs, focusing on structural property of networks: spotting near-cliques and finding triangles. Then the author explained that the alternatives for computing the eigenvalues of symmetric matrix including Power Method, Simultaneous iteration and Lanczos-NO are not suitable for big data on mapreduce. So the author described the algorithm for computing the top K

eigenvectors and eigenvalues with four speficic fields improvement: Careful Algorithm Choice, selective parallelization, blocking and skewness exploiting. Finally it turned out that performance improved in both scalabitily and skewed matrix data, compared with HEIGEN-PLAIN.

- *Use for our project*: Largely based on mapreduce, HEIGEN is a totally new algorithm. What we can learn is that with massive data, we can change the former way of thinking for graph mining, so that we can largely improve the performance without SQL. And also, with the infrastructure of hadoop, and the way map/reduce reading data, we can do modification to use the advantages to get even better performance.
- *Shortcomings*: Highly based on map/reduce architecture also brings lots of problems that hadoop has. Such as the job schedualing and data shuffling. As the matrix operation needs to read large amount of data, and for iterative calculation, spark is a better choice as everything is in memory.

2.3 Papers read by Ye Zhou

The third paper was the Unifying Guil-by-Association Approaches paper by Koutra

- *Main idea*: The paper mainly proposed FaBP, which is a Fast Belief Propagation algorithm on Hadoop. It first compare and contrast several very successful, guilt-by-association methods: Random Walk with Restarts, Semi-Supervised Learning and Belief Propagation. The author showed that these three methods are closedly related but not identical. Then he proposed the algorithm FaBP, showed the experiments result. It turned out that the accuracy keeps the same or even better with the traditional BP, but the performance is twice better. It also has convergence guarantee. It is even sensitive to the "about-half" homophily factor, as long as the latter is within the convergence bounds. It also scales linearly on the number of edges.
- *Use for our project*: Learn the way using hadoop to impelment the algorithm for machine learning.
- *Shortcomings*: Again it is based on Hadoop, the performance for iterative calculation is not so good compared with spark. And BP algorithm is not so efficient when dealing with graph which has circle. And the convergence is limited due to specific requirement.

2.4 Papers read by Jin Hu

The first paper was the GBASE paper by U Kang

- *Main idea*: The paper introduces a general graph management system GBase for large scale graph storage and computation.
- *The main contribution of the paper:*: GBase uses "compressed block encoding" method to make graph storage more efficiently. For graph indexing, the paper succeeds in handling multiple type of queries on a large graph instead of a specific type and is suitable for distributed environment. By supporting homogeneous block level indexing and being flexible in both edge and node centralized computing, GBase has better

properties than similar distributed systems. The framework the paper proposes also supported both graph-level and node-level queries, making it applicable to various applications. GBase partitions data in two dimensions to better use the block and community-like properties of real-world graphs, which gives it advantage over either row-oriented or column-oriented storages.

- *Limitations*: The paper's indexing method handles large graphs successfully, but its property compared to frequent subgraph or significant graph pattern methods are not shown in the experiment. Optional indexing methods may be added to the system.

2.5 Papers read by Jin Hu

The second paper was by Danai Koutra

- *Main idea*: The paper does the comparison among some of the most popular guilt-by-association method.
- *The main contribution of the paper:*: The paper manages to prove that all methods result in a similar matrix inversion problem. In addition, the paper proposes a fast and accurate BP algorithm. In theory, the paper finds that RWR(Personalized Random Walk with Restats), SSL(Semi-Supervised Learning) and BP(Belief Propagation) are closely related, but not the same. RWR and SSL are not heterophily, but BP is heterophily. All three methods are scalable. RWR and SSL have convergence while BP is unknown. The proposed FABP method has nice property with all these perspectives. FABP is an approximation of standard BP, but FABP is significantly faster based on the experiment and guarantees convergence, which makes it better than BP. The experiments also verify the paper's ideas. The author tested the theory and the properties of the proposed FABP method in terms of accuracy, convergence, sensitivity to parameters and scalability.

2.6 Papers read by Jin Hu

The third paper was by Ignacio Alvarez-Hamelin

- *Main idea*: The paper introduces K-core decomposition and its application in the visulization of large scale networks.
- *The main contribution of the paper:*: K-core decomposition can find subgraphs which all of the nodes in the subgraph have degrees higher than k after removing nodes with lower coreness. This method can find the subgraphs which are more closely connected and achieves "clustering" in large graphs. As K-core decomposition can produce two-dimensional layout of large scale networks with their important topological and hierarchical properties, the paper takes advantage of the K-core algorithm to allow visulization of network and offer features like fingerprint identification and general analysis assistance. The visualization algorithm has linear running time proportional to the size of the network, making it well scalable for large networks. In addition, the algorithm offers 2D representation of networks which makes information visualization

more accessible than other representations and the parameters of the algorithm are universally defined, which makes it suitable for all types of networks.

- *Limitations*: The proposed visualization algorithm still utilizes certain parameters to identify the properties of the network, which involves considerable human interactions and prior expemental knowledge. Self adjusting parameters might be a huge improvement and can be an interesting topic to follow.

...

3 Proposed Method

We implemented K-core using SQL following the Batagelj and Zaversnik k-core variation. We recursively prune the nodes and edges in the graph with degree less than k to finally arrive at points of degree greater than or equal to k.

...

4 Experiments

We implemented kcores method and tested other algorithms that are already implemented in the system with unit tests and other datasets from SNAP.

Figures below give the degree distribution and pagerank result of two dataset from SNAP. We can find that the degree distribution and pagerank results are consistent with the power law as nodes or pages with higher degree or rank have a small number while nodes or pages with lower degree or rank have a large number. You can also find the detailed results in the output folder, which contains csv files for the results of belief propagation, connected components, node degrees, degree distribution, eigen values, k-core connected components, pagerank results, radius, etc.

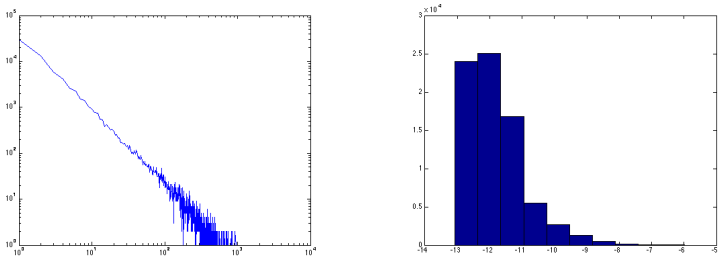
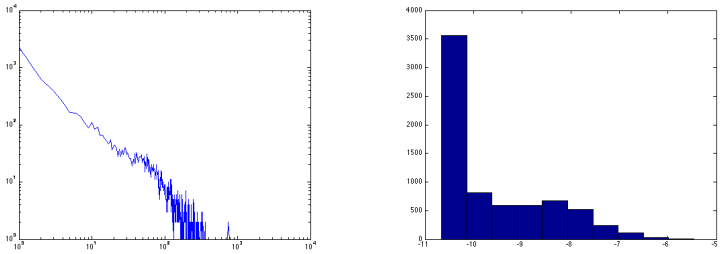


Figure 1: Degree Distribution(a) and PageRank(b) for Dataset SOC-Epinions1



(a) (b)

Figure 2: Degree Distribution(a) and PageRank(b) for Dataset wiki-Vote

The figures below also include the degree distribution, connected components, k=5 cores algorithm results on the 5 unit tests. As the unit tests are small, there is no nodes in the tests that satisfy k=5 cores, so the output result for these 5 unit tests are empty. However, you can find the node id, component id pairs in the stdout output from console or in the kcorecomponent.csv file, which shows that the k-core algorithm works as it claims to find correct coreness subgraphs.

Degree	Count
2	11

Node_id	Component_id
8	0
4	0
1	0
5	0
3	0
0	0
10	0
9	0
6	0
2	0
7	0

Figure 3: Degree Distribution(a), connected components(b) for Dataset1

Degree	Count
4	5

Node_id	Component_id
4	0
1	0
3	0
0	0
2	0

Figure 4: Degree Distribution(a), connected components(b) for Dataset2

Degree	Count
4	7
3	3
9	1

Node_id	Component_id
8	0
4	0
1	0
5	0
3	0
0	0
10	0
9	0
6	0
2	0
7	0

Figure 5: Degree Distribution(a), connected components(b) for Dataset3

Degree	Count
1	2
2	3

Node_id	Component_id
4	2
1	0
3	2
0	0
2	2

Figure 6: Degree Distribution(a), connected components(b) for Dataset4

Degree	Count
3	16
2	5

Node_id	Component_id
8	0
16	0
15	0
4	0
20	0
1	0
13	0
5	0
11	0
3	0
14	0
17	0
0	0
19	0
12	0
10	0
18	0
9	0
6	0
2	0
7	0

Figure 7: Degree Distribution(a) connected components(b) for Dataset5

5 Conclusions

Based on the experiments and the implementation of Kcore, we find that the Kcore algorithm can find subgraph structures effectively and we also find that the implemented algorithms in graphMiner, i.e. the pagerank, degree distribution, connected component etc works properly. Above this, we find that using SQL to do basic and advanced graph mining queries are viable and actually effective. We will have future exploration into this to make better use of SQL and advantage of database.

A Appendix

A.1 Labor Division

The team performed the following tasks

- Implementation of Kcore [Jin Hu]
- Unit Tests and visualization of results [Ye Zhou]
- General debugging and testing[Ye Zhou and Jin Hu]

A.2 Code for K-core

asd

Contents