Local graph clustering on real-life dataset

Lichen Zhang

October 31, 2019

Nowadays, recommender system has become more and more important to industrial applications. Classic approaches including collaborative filtering and matrix factorization cannot embed more structured information of data, i.e., the *relationship* between different contents. One novel way to capture this idea is to construct a relational graph to model this kind of problem. As the size of the data gets larger and larger, traditional way of constructing graphs and classic graph algorithm which has run time with respect to the global parameter of graph becomes especially impractical and inefficient. However, for real-life applications especially for recommender system, it is actually not necessary to process the whole graph, in the sense that queries are usually corresponding to a single vertex or a small subset of vertices, and our goal is to find a good local cluster of it, such as vertices inside the cluster is closely connected, compared to edges go outside of the cluster. This is the idea of *local graph clustering*. My research project builds on this idea.

In my project, I will work with Professor Gary Miller, and we plan to do things in several stages: we start by reading papers related to local graph clustering, from the old ones that use random walk with restarts [1], to more advanced and nearer ones that use evolving sets [2] and paint spilling, particularly a recent paper that used flow-based algorithm [3]. It modified the push-relabeling algorithm so that it gives a better guarantee on finding a good local cluster. Then we will look at some real-life graphs, for example, social network graphs of Facebook and Pinterest graphs. The goal is to implement and run these theoretical algorithms on these datasets and examine their performance. Specifically, our work might be closely related to the state-of-the-art recommender system that has been developed by Pinterest, which is called PinSage [5]. In their applications, their graph has 3 billion nodes and 17 billion edges, and user will make a query by clicking on a *pin*, which corresponds to a node in the graph, and their task is to find other pins that are "close" to the query in the graph. Basically what they do is they first perform a very simple random walk to find a cluster around the query, then they apply a very new and popular machine learning algorithm called *graph convolutional neural network* to this cluster and learn an embedding for this node, i.e., a vector representation for it. All such process is offline training, when the learning finished, they have all embedding for nodes, and when user makes a query, they do some nearest neighbor search using that embedding. In fact, their local clustering technique is very simple and does not have any theoretical guarantees, and this might be something we can work on. The question is, if we apply some more sophisticated clustering algorithm, will the performance of GCN being improved? We want to answer this question by implementing and experimenting novel algorithms on their dataset.

The main challenge comes from three aspects: first, it is non-trivial to turn a theoretical result into a large-scale application. It is not rare that algorithm works well in theory does not scale well in practice. Second, how can we get such huge graph? It might not be realistic to get the whole graph from Pinterest. Third, assume we manage to get or generate the graph, then how can we in fact implement algorithm that runs on such large graph? Although it is possible to store the graph in a single computer(around 120G), if we want to do some empirical evaluation, then it is imperative to use some distributed or parallel architecture, such as Sparks. If we can manage to tackle this problem, then this project can really have some impact: since PinSage is the state-of-the-art, this is a chance to actually improve on the state-of-the-art.

Next, we discuss the 75%, 100% and 125% goals for this project. In the worst case scenario, i.e., we cannot find the dataset, and time does not permit us to implement all kinds of algorithms, we will generate data on our own, then implementing the capacity releasing diffusion algorithm, which is an analogy to push-relabeling algorithm, therefore we believe it is easier to implement. If everything goes as expected, then we will implement at least three algorithms, including capacity releasing diffusion, random walk with restarts and evolving sets, and do empirical evaluations on them. In the best case, we do have some additional time, then we will try to actually develop some new algorithms based on combinatorial or spectral methods, and give theoretical guarantees.

Talk about milestones for this project, for the 1st milestone at the end of this semester, we will 1) either get the dataset or generate our own dataset; 2) fully understand the algorithms we want to implement, both in theoretical(why it works? proofs?) and practical(how to achieve it in practice) aspects. We will also have our first version of capacity releasing diffusion algorithm, which is a prototype. On Jan 27th, we will finish experimenting CRD algorithm on some comparably small dataset. On Feb 10th, we will complete the implementation of large-scale CRD algorithm. On Feb 24th, we will experiment CRD algorithm on target dataset, i.e., huge graphs. On Mar 16th, we will finish the scratch implementation for random walk with restarts and evolving sets. On Mar 30th, we will hopefully complete the large-scale version of these two algorithms. On Apr 13th, we will evaluate these two algorithms on target dataset. On Apr 27th, we can finish and conclude all experiments we've done so far.

Papers we have read and studied including the general background for random walk, random walk with restarts and the PageRank Vector [1], random walk with evolving sets [2], capacity releasing diffusion [3]. For application part, we have read two recommender systems built by Pinterest, the first is called Pixie [4], which is an online recommender system that deals with queries completely using random walk. The second is PinSage [5], which is a novel architecture that uses random walk in Pixie, GCN and a sophisticated pipeline to achieve efficient inference. It will take quite a long way for me to fully understand these algorithms in full details, since they require quite a lot prerequisite knowledge involving random walks, spectral graph theory and some studies about local clustering. For resources, I haven't considered it very deep, but intuitively we need to exploit some large distributed and parallel computing platforms for empirical evaluations. I am not familiar with this part, so it will take some time to figure it out.

Finally the link: Click me.

References

[1] Reid Andersen, Fan Chung, and Kevin Lang. Local graph partitioning using pagerank vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 475–486, Washington, DC, USA, 2006. IEEE Computer Society.

[2] Reid Andersen and Yuval Peres. Finding sparse cuts locally using evolving sets. *CoRR*, abs/0811.3779, 2008.

[3] Di Wang, Kimon Fountoulakis, Monika Henzinger, Michael W. Mahoney, and Satish Rao. Capacity releasing diffusion for speed and locality. *CoRR*, abs/1706.05826, 2017.

[4] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. *CoRR*, abs/1711.07601, 2017.

[5] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. *CoRR*, abs/1806.01973, 2018.