MOBILE NETWORK

MULTILEVEL KNOWLEDGE GRAPH PARTITION

IMPLEMENTATION AND GENERALIZATION

Mingcheng Chen
515030910568
mcchen1997@outlook.com

Zhikun Wei
515030910574
nin9K@sjtu.edu.cn

Supervisor: Prof. **Xinbing Wang**, Ph.D **Yuting Jia**
Department of Computer Science and Engineering
Shanghai Jiao Tong University

Final Report for EE447: Mobile Network

Contents

Abstract

Knowledge Graph(KG) is a branch of knowledge engineering, which has evolved from the semantic network. Due to its great application prospects in search and recommendation systems, it has been rapidly developed in recent years, driven by the latest technologies such as machine learning and natural language processing. Knowledge graph has drawn extensive attention of the industry and academia. However, as the scale of knowledge graph becomes larger and larger, it is difficult to put the entire KG into use directly. Our work focus on dealing with large-scale knowledge graph, applying efficient and high-quality partition on the original KG so that derive the sub-graphs to meet user customization requirements.

Keyword: Knowledge Graph, Multilevel Partition, Generalization, Approach/Method

1 Introduction

Today, knowledge graph is widely applied in search and recommendation systems. The Acemap Team led by Prof. Xinbing Wang and Prof. Weinan Zhang of Shanghai Jiao Tong University recently published a Academic Knowledge Graph — AceKG, which provides a data set of nearly 100G in size, providing rich attribute information for each entity and covering authoritative academic knowledge. The major of our work is based on AceKG. We proposed an approach to divide huge raw KG into small sub-graphs with high quality efficiently, and further encapsulate this method to design a general tool providing user customization KG partition service. Our work dedicate to meet the designing intention of AceKG, supporting varieties of academic big data mining projects.

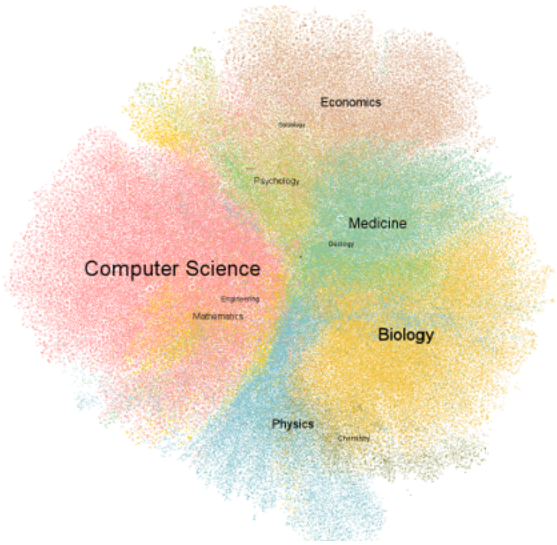

Figure 1: Visualization of AceKG Knowlege Graph

1.1 The Composition of AceKG

The AceKG is a large-scale academic knowledge graph, it covers over 100 million academic entities and 2.2 billion triplets over comprehensive academic information. The types of entities and triplet relationship is shown as follows: There are 5 entities in AceKG, which

Number	Entity Name	Description
1	Paper	The most important skeleton entity
2	Author	The author of a paper
3	Field	Research field an author or a paper is in
4	Affiliation	Affiliation an author belongs to
5	Venue	Journal or Conference a paper is published on

Table 1: The Composition of Entities in AceKG

interconnected with one another according to following 7 triplet relationship, and build up the topology framework of AceKG.

Number	Triplet Name	Description
1	*work_in*	Author A is work in Affiliation B
2	*author_is_in_field*	Author A is in Field B
3	*paper_is_written_by*	Paper A is written by Author B
4	*paper_is_in_field*	Paper A belongs to Field B
5	*paper_publish_on*	Paper A is published on Journal or Conference B
6	*paper_cit_paper*	Paper A's citation papers contains Paper B
7	*field_is_part_of*	Field A is a sub-field of Field B

Table 2: The Composition of relationships in AceKG

We conducted summary statistics in database of AceKG, the overall results are listed as follows,

Item	Statistics Result
Academic Entity	100 million
Triplet Information	2.2 billion
Paper	61,704,089
Scholar	52,498,428
Research Field	50,233
Academic Research Institutions	19,843
Academic Journal	22,744
Academic Conference	1,278
Academic Alliance	3

Table 3: The Composition of relationships in AceKG

1.2 Overview of Topology

AceKG provides rich attribute information for each entity, and adds semantic information based on the network topology. The entities and triplets together build up the complex topology framework.

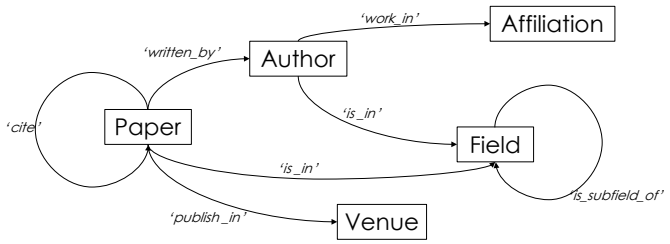

Figure 2: Topology of AceKG

2 Motivation

The enormous size and complexity of the components triplets in AceKG, result in the difficulty of its flexible usage. When it comes to application of knowledge graph with nearly 100G data volume, how to divide and make good use of such a huge KG becomes a key problem. This release of AceKG is an overview of the entire academic community from a higher perspective, so the data set released this time not only in the field of computers, but also in medicine, communications and other fields. Therefore, to further efficiently make use of huge data set, it's a compelling need to make partition of raw knowledge graph.

Take a vivid metaphor, the total raw data of AceKG just like a whole pork. For a normal person, this whole pork cannot be eaten at once. Scholars in the medical field may like to have some 'ham', while scholars in the computer field may like to eat 'trotters'. Even for the same portion of meat, one can make different kinds of dishes such as fish fragrant pork or sweet and sour pork ribs.

Figure 3: Knowledge Graph Partition Metaphor Figure

Before recommend the specific and detailed data set to man we interested through social networks, there are more important things to do. That is making partition on the raw KG data set to gain multilevel academic heterogeneous network data set and multi size of sub-graphs.

3 Tasks Division

For the purpose stated, we formulated following step-by-step tasks,

1) Proposed approaches to get different size of small-scale subgraphs, while ensuring the quality(density and balance) of generated subgraph.
2) Optimize approaches in first step, further improve the efficiency of algorithm. Implement the methods for customizing personalized data sets, which mainly contains two requirements:
   a. Divide the raw data set by *Field*
   b. Divide the raw data set by *Venue*
3) Finally encapsulate the work above into a generalized tool, to meet user customization requirements and help them use the AceKG flexibility and conveniently.

The detail tasks division intra our two-person team is given as follows,

3.1 Basic Approach

This part of task is done by both Chen and Wei together.

3.2 Advanced Partition customization

Zhikun Wei is responsible for divide the raw data set by *Field*, while Mingcheng Chen take responsibility of dividing the raw data set by *Venue*. Wei optimized the original algorithm and Chen make sure the density of sub-graphs derived.

3.3 Further Generalized Tool

Mingcheng Chen encapsulate the two part of approaches, Zhikun Wei combine them together to get the final generalization tool. Both of two people take part in the testing and debugging.

4 Partition by Field

As is introduced before, there is an urgent need for some suitable data set. And the First work we do is to find some dense, small size KG of some specific areas from AceKG. We have to ensure the smaller size and higher quality at the same time. Detailedly, we will keep abundant edges in the graph, more denser than before with edge distributing uniformly. Finally, we also need to notice the efficiency for time consumption will be considerable when faced a 100G data base.

Now we need to cut a KG out which is under some fields. We are given 5 L0 level fields and 5 L1 level fields for each L0 fields. The objective is to form 5 datasets for L0 level fields. In each dataset, 5 L1 level fields must have obvious difference while they must have relationship.

4.1 Methods

The methods we use is objective-driven, and the whole process can be concluded as following:

1) Initially choose some seed nodes.
2) Extent to other nodes according to specific rules.
3) Choose proper edges as new data set.

We search the database to find all the paper ID related to L1 fields as th seed nodes then extend to author, venue, field and affiliation. The extension rules is designed to balance the edges and ensure quality. Rules include author only extend to L0 and L1 fields, affiliation is extended only when paper and author are matched, and so on. When we have gone through all the paper ID, we consider adding edges. We balance types of edge and keep edges are divided into train/test set with proper ratio.

4.2 Challenges and Solutions

Challenges we faced are quit valuable ones. Through the process of solving them, we developed many useful methods and we use them as tools to solve all the similar problems.

There are three typical and critical challenges:

- Nonuniform density graph
- Algorithm complexity
- Quality and usability

4.2.1 Nonuniform density graph

In the real data base AceKG, we found that some part is sparse while some part is dense. The concept sparse and dense roughly means the low node degrees and high node degrees respectively. The data is used to train some KG model, so a uniform density graph is of much significance. We need try to select dense parts and avoid sparse ones.

The challenges lie in the fact that we don't exactly know which part is more dense until we visit through the whole graph. Actually, it is impossible to load the whole into memory and operate on it for the hardware limit, and this direct way is not efficient and economy either.

To address the nonuniform density problem, I use an algorithm: "batch BFS". Just as its name suggests, this algorithm is a adjusted version of BFS. Algorithm is described as below:

1) Firstly we apply BFS to the graph starting from some point.
2) But we don' t add visited points to our set until it is connected to its brothers through at most two edges(other than edges to its parent).
3) It works as a latency version of BFS. The "batch" means decision is made two steps after visit, then the batch is two.

This algorithm can solve the problem of density requirement. And in the meantime, for the degree of each point is smaller than a constant c, the complexity will be still O(m+n).

By using this algorithm, we can avoid get into sparse area of the graph and stay in the dense area. We also don't know which part is dense or not, but we will just choose the more likely dense node and ignore the likely sparse node.

4.2.2 Algorithm complexity

This challenge shows up when we develop our algorithm to visit the nodes. There are millions of nodes the AceKG, and the edges are approximately at the polynomial level of nodes. Not only nodes matters, edges matters too. For reducing the complexity, we have to prune on the edges we will visit.

Besides prune the nodes as in previous issue, we have to limits the types and number of edges. Some nodes, say, author may have hundreds of edges connected to all level fields, But what we need is just several L0 or L1 level field. To address this problem, we develop some pre-process trick in fetching data from the database.

When we send query to the database from one node to other nodes, We always ask one more information: the level(or count, or degree) of the other nodes. And we will order the fetched nodes before adding them into new graph and just keep a few nodes which have higher level. We do this to break the complexity of a node from m to a constant c.

Some other problems are related to the actual code realization. A normal query to database would take at least "ms" level time. So it' s impossible to query for each node. We have tried:Batch query, integrate queries into one;Asynchronous query, realized using threads. But they have no use in reducing the order of magnitude of time consumption. Weeks are needed if we use this way.

Thanks to senior(Jia/Wang)' s help, we solve this in a completely opposed way: we don' t probe using queries, instead we operate in memory and probe in dictionary (hash probe takes just 'ns' level time)

The specific method is that

- Download the whole table into disk as files.
- Read data from the file and run our algorithm simultaneously. This could be realized due to that our algorithm don't have to see the whole graph to determine what to do.
- After reading all the nodes we need, we begin fix edges between some self circle nodes(such as paper citing paper). Due to the previous work, the complexity of adding edges are not $O(n^2)$ but $O(cn)$ where n is the number of paper nodes and c is a constant for a paper's max citation.
- Finally we get a proper data set we want.

4.2.3 Quality and usability

Our objective is not just get a smaller and dense data set, but also a usable KG data set with high quality and suitable for KG model. The challenge is that when the original graph is unstable and its edge type don't distribute uniformly, the data set we cut out face the same risk. Anther problem is when we divide train set and test set, we have to make sure all entities show up in the training set and keep the train/test ratio constant. The challenge is that we divide the data set by edges but this require is about nodes.To solve them, we also develop some methods:

- The degree of each node is conditionally controlled.
- A dynamic allocation method to divide train and test set.

The degree of each node is conditionally controlled based on the edge types. In the data set, edges distribute non-uniformly mainly on some edge types, such as author-affiliation, author-field. Some node have hundreds of this kind of edges while some nodes have just a few. And edges differ between types too, some type may average over hundreds of edge but some may stay less than ten. So the control is also mainly on types. To some sick distributed type, we use the prune trick developed previously.

The dynamic allocation method is easy to realize. It can be derived from our algorithm previous. We just need to keep track on the nodes in the train set. When we add a new edge, we firstly check if the two ends are in the train set nodes or not. If not, add it in train set immediately. Otherwise, compute the ratio of train and test set, and add it into the lower one. At the beginning of the process, the train set may be over balanced, but with the process going, new nodes will not appear, the balance will come back.

Quality and usability can be ensured by the two tricks. And in fact, the data set we generated has being used successfully.

5 Partition by Venue

This part focuses on subdivide the data set in another angle. Similar to the approaches before, we proposed an approaches to divide the knowledge graph by Venue.

In view of the analysis of algorithmic time complexity, the major problems encountered have been described in the previous section together, here we focus on features of 'Divide by Venue' and our trial on ensure density of subgraph as well as its limitation.

5.1 Mainstream Journal

According to Dong et al. [2017], there are 8 classes of mainstream venue classes in google schoolar, and each class contains 20 top venues. There are the list of total 8 classes,

- Computational Linguistics
- Computer Graphics

- Computer Network & Wireless Communication
- Computer Vision & Pattern Recognition
- Computing Systems
- Databases & Information Systems
- Human Computer Interaction
- Theoretical Computer Science

Our partition work by venue is based on such a consist of classes.

5.2 Local Offline Algorithm

To ensure the subgraph's density, improving the quality of partition, we made a lot of trials. For some small scale knowledge graph, which can be storage into memory, we proposed an offline algorithm to get a dense subgraph.

Firstly, we add weight for the edges in the graph by following formula.

$$w_{ij}^{m} = \exp^{-\frac{2}{d_i + d_j}} \tag{1}$$

If an edge which connects two nodes with higher degree, we assign it with higher weight. To ensure the compactness[Wei et al. [2016]], a random walk, starting at some well vertex (with high degree and is the articles we interested in) are determined man-made, and then randomly travelling to a connected vertex, is more likely to stay with in a cluster than travelling between. Therefore conduction random walks on the graph can discover clusters where the flow tends to gather. The transition probability from a vertex v_i to a vertex v_j is defined as a set function $P_ij(A) : 2^E \to R$ for the graph G(V, A):

$$P_{ij}(A) = \begin{cases} 1 - \frac{\sum_{j:e_{ij} \in A} w_{ij}}{w_i} & \text{if } i = j, \\ \frac{w_{ij}}{w_i} & \text{if } i \neq j, e_{ij} \in A, \\ 0 & \text{if } i \neq j, e_{ij} \notin A, \end{cases} \tag{2}$$

which encourages random walks within dense subgraph(clusters $e_{ij} \in A$) and eliminates those between clusters ($e_{ij} \notin A$). $\sum_{j:e_{ij} \in A} w_{ij}$is the total weights incident to v_i. We add a self loop transition ($i = j$) to maintain the total transition probability out of v_i to be 1.

5.3 Limitation

This algorithm is an offline algorithm based on local knowledge graph, as a result, when it comes to ultimate large-scale data set(such as the whole AceKG), it will be time consuming and the usability declining severe.

6 Generalized Tool

In addition to our specific object that divide the data set, we also want to generalize the methods we use into a useful tool.

With the tool, we can divide a data set automatically when we input some parameters and set initial conditions. Moreover, we can ensure the quality of the result by our methods in balancing the types of edges. And we solve another tricky issue that all entities must show up in the training set while we keep steady ratio between training, valid and test set.

To encapsulate the methods we use into a tool, we do some generalization work on it.

- Separate parameters out
- Arrange the process to run automatically
- Automatically recognize imbalanced edges

The parameters include the initial seed nodes, the dataset scale, train/test ratio. Initial seed nodes can be provided as a list of paper ID. Starting from paper is the most efficient way to from a new dataset. Other initial seed nodes needs more pre-processing and have to transfer to paper ID any how. The dataset scale can be limited through stopping adding new nodes when the nodes number reach a given value. train/test ratio also can be controlled, the dynamic allocation method contains this parameter.

Run automatically is almost achieved in our original method, after feeding the seed nodes, remaining process can run automatically.

The most challenge one would be automatically recognize imbalanced edges. It is easy to detect the ratio of different types in given area, and we just set our extension rules accordingly. In the extension rules, the number of edges to extend will be determined by the relative ratios of different types.

Now we have a useful tool designed for data partition in AceKG. Any one can use it to cut proper dataset from AceKG, and he can get the size and content he wants with high quality and usability.

7 Conclusion

In this project, we started from a simple task: cut out a smaller size dataset. Then we finished two tasks with higher requirement: ensuring the quality and usability. Finally we generalize our methods into a tool: anyone who want cut a dataset from AceKG can use it.

Sincerely thanks for our supervisors for their valuable guidance and being patient with us. We also benefit a lot from this experience.

References

Y. Dong, N. V. Chawla, and A. Swami. metapath2vec: Scalable representation learning for heterogeneous networks. In *KDD*, 2017.

Y. Wei, Y. Zheng, and Q. Yang. Transfer knowledge between cities. In *KDD*, 2016.