Schema Discovery for Property Graphs

Yuzhan Wang

Laboratoire David, Université de Versailles Paris-Saclay

Abstract. In our work, we are interested in the automatic extraction of a schema describing the content of a graph database, where data is decribed as a property graph, in order to provide a synthetic representation of the data. Such schema could be used to better understand the available data and to ease the exploitation of the considered database. We introduce a schema discovery approach suitable for property graphs. It relies on an indexing technique, Locality Sensitive Hashing (LSH), analysing the nodes and the edges of the graph to generate node vectors for each node, and ultimately dividing each label type into multiple subtypes, thereby providing a simplifying representation of the property graph into acompact schema. When adding new data, our approach only processes the new nodes, in order to avoid the need to recalculate the entire schema, which ensures the scalability of the discovery process.

Keywords: Property Graph · Schema Discovery · LSH.

1 Introduction

In many sources, the data is not described considering a predefined structure, and it does not have to conform to this structure. Data in these sources is irregular and does not follow a predefined schema. This is the case of data on the semantic web, described in the RDF languate, or data stored in graph databases, which are described as property graphs. This lack of schema can make understanding a data source difficult and complicate its use. Given these challenges, discovering an appropriate schema that can describe the diversity of data is important, as schema discovery plays a crucial role in effective data integration and management.

There have been several studies on schema discovery for property graphs and RDF data. In schema discovery for property graphs, one of these studies introduces and approach based on Gaussian Mixture Models (GMM) [2], taking into account both node labels and attributes. This method divides the dataset into multiple clusters, with each cluster corresponding to a subtype of a node type. GMMSchema then applies a recursive clustering process to the data graph until no significant subtypes can be found. However, this method is not suitable for incremental schema discovery, which can be useful as property graphs tend to expand with increasing data volume.

Incremental schema discovery has been addressed by the HinT approach [3], which incrementally discovers the schema of an RDF dataset. This method considers entities in a dataset and their properties, and creates a node vector for each entity. It then applies Locality-Sensitive Hashing (LSH), using a series of hash functions to map similar patterns to the same hash bucket. This approach aims to assign patterns with similar attributes to the same group. Each group represents a type, and all patterns within a group represent entities having the same type.

The objective of our work is to implement an automated schema discovery method for property graphs and propose a solution that supports both incremental and scalable processing, taking into account both node labels and attributes during the schema discovery process. The final discovered schema will be created based on a schema model proposed in [1].

2 Our Proposal

To achieve incremental type discovery in property graph data, we propose using LSH to assign similar entities to the same hash buckets through different hash functions. Then, by grouping the patterns in these hash buckets, we can determine the subclasses of each label in the graph. Based on the label categories that we obtain, we can derive a schema for the property graph according to the formalisme proposed in [1]. Our approach is composed of three main steps: pattern discovery, locality-sensitive hashing of the nodes in the property graph, and type assignment.

In the first step, which is pattern discovery, a node vector is defined or each node of the graph. This vector can be defined in two ways: in the first method, the node vector consists of all the attributes of the node; in the second method, we also consider the relationships between nodes, the node vector includes not only all the attributes of the node but also the labels and attributes of the outgoing edges of this node. We then identify the pattern, each one is characterized by a set of properties and represents the set of nodes described by the same set of properties. The patterns are stored in a pattern index, which allows us to efficiently retrieve the patterns based on the node vectors and to check whether a specific node vector already exists among the stored patterns.

The second step is Locality sensitive hashing. First, we randomly select r hash functions from the LSH family. These hash functions are designed to ensure that similar nodes have a higher probability of being mapped to the same hash bucket. When a new pattern is generated, its signature is calculated using the r selected hash functions. Each signature is divided into b bands, and these bands correspond to b different hash tables. The LSH index stores the signatures of patterns across multiple hash tables, making it more likely for similar patterns to be placed in the same hash bucket, meaning patterns in the same bucket can be considered as the same subclass.

The third step is type assignment. We group similar patterns based on the values of their signatures generated by the sensitive-hashing functions. This grouping allows us to identify patterns that are similar, and that are likely to have the same type. For each node in the dataset, we search for the buckets to which the node is assigned and whether these buckets contain other patterns. If the node is similar to existing patterns, we assign it to the same type as those patterns. If not, we consider it a new type.

In our method, each node is processed independently, without being compared to all other nodes in the graph to determine its type. This allows us to generate a schema without making pairwise comparisons between nodes, which can become very costly on large graphs. Moreover, when a new node is inserted into the graph, it is not necessary to compare it with all other nodes or to restart the schema generation process to determine its type. Through these steps, we can achieve incremental type discovery for large property graph datasets.

After group assignment, the final schema describing the property graph is generated. To define this schema, we use the language proposed in [1]. The generated schema proposed can be intepreted by a query language which is similar to Cypher, which means that once the type of the nodes have been discovered, we can generate the schema using these types, the attributes and the edges of the nodes. During the third step, if a node of a pattern contains new node attributes or edge attributes, we add these attributes and edges to the corresponding node and edge types.

3 Conclusion

This project aims to propose an approach for automated property graph schema discovery. Compared to previous methods, our approach is incremental and scalable, and it retains only the outgoing edges of each subtype of node, rather than all outgoing edges of that type. To ensure scalability, we have employed Locality-Sensitive Hashing (LSH) indexing technique, analyzing nodes and edges, generating node vectors, and refining label types to provide a simpler schema structure for the property graph structures.We are currently in the implementation and testing stage of our work.

References

1. Angles, R., Bonifati, A., Dumbrava, S., Fletcher, G., Green, A., Hidders, J., Li, B., Libkin, L., Marsault, V., Martens, W., et al.: Pg-schema: Schemas for property graphs. Proceedings of the ACM on Management of Data **1**(2), 1–25 (2023)
2. Bonifati, A., Dumbrava, S., Mir, N.: Hierarchical clustering for property graph schema discovery. In: EDBT 2022: 25th International Conference on Extending Database Technology. pp. 449–453. OpenProceedings. org (2022)
3. Kardoulakis, N., Kellou-Menouer, K., Troullinou, G., Kedad, Z., Plexousakis, D., Kondylakis, H.: Hint: Hybrid and incremental type discovery for large rdf data sources. In: Proceedings of the 33rd International Conference on Scientific and Statistical Database Management. pp. 97–108 (2021)