**Presentation**

—————————————————————————————

#### Page 1

Knowledge graphs are popular in managing large scale and real-world facts. It models the entities with attributes and the relations between entities as a big graph. This is a knowledge graph snapshot from DBpedia, involving the knowledge related to automobile manufacturing.

#### Page 2

Querying knowledge graphs is critical for a wide range of applications, such as question answering and semantic search. There are two important query forms on knowledge graphs, that are: factoid query and aggregate query.

The answers to a factoid query are defined as an enumeration of noun phrases.

While for aggregate query, it’s usually used to explore the statistical result of a set of entities given a specific entity and a semantic relation.

#### Page 3

One frequently used technology to answer factoid queries is graph query. Specifically, a user constructs a query graph to describe her query intention, and identifies the exact or approximate matches of query graph in a knowledge graph through graph matching.

#### Page 4

In contrast, answering aggregate queries on knowledge graphs has been mostly ignored in the literature. Aggregate queries can be extended from factoid queries, by applying an additional aggregation on factoid queries’ answers to obtain the statistical result of interest. As we illustrated here, given an aggregate query that includes a query graph and an aggregate function, we can first apply existing graph query algorithms to obtain graph matches to the given query graph, then we do the aggregation on these answer entities to get the aggregate result.

However, this straightforward solution is problematic in both effectiveness and efficiency aspects. First, this solution depends on the quality of factoid queries answers. Since none graph query methods can return 100% accurate answers, and none of them have an effective way to quantify the result’s quality, seriously affecting the aggregate queries’ accuracy. Second, aggregate queries’ runtime depends on the computationally expensive graph query methods, which is also inefficient.

#### Page 5

Different to factoid query-based solution, we propose a “sampling-estimation”-based approximate solution to efficiently answer aggregate queries over knowledge graphs, having an accuracy guarantee, but without requiring factoid query evaluations. This is the pipeline of our solution: We first present a semantic-aware random walk sampling to collect answers that are semantically similar to a query graph as a random sample. Next, we estimate an unbiased (or consistent) approximate aggregate result based on the random sample, and provide an accuracy guarantee by iteratively computing a tight enough confidence interval at a confidence level. We terminate the query when a tight confidence interval is obtained and we prove that the relative error can be bounded by a predefined error bound if the confidence interval is tight enough.

#### Page 6

Comparing with recent works on knowledge graph search, our solution has two orders of magnitude less relative error on average than other methods. While for efficiency evaluation, our method requires up to an order of magnitude less response time than others.

#### Page 7

Now, I will show the demonstration of our “sampling-estimation” solution to aggregate queries based on DBpedia dataset. We list several aggregate queries in natural language form on the left, including some aggregation variants of the factual query in QALD, as well as some synthetic queries. When we select one query, its query graph is formalized on the top with a specific entity (the red node with name *California*), a target entity (the orange node with type *software*), a predicate *foundationPlace*, and an aggregation function COUNT. We also provide a knowledge graph snapshot of DBpedia that contains the specific entity *California*. Since the graph queries exhibit strong access locality, most correct answers could be found in the area close to California. When we submitted this query, we will conduct a semantic-aware random walk starting from *California* until it converges to a stationary distribution. Each entity would have a stationary visiting probability. Then we can collect a random sample of software according their visiting probabilities. The larger the visiting probability, the greater the semantic similarity of an answer to the given query graph. At the beginning, we obtain a sample with 8 software. We show each software’s visiting probability and its semantic similarity to query graph, as well as the path between *California* and this software. We can estimate the approximate aggregate result based on the collected random sample, through unbiased (or consistent) estimators. Moreover, we provide an accuracy guarantee in the form of confidence interval with the half width . The smaller the half width, the more accuracy of the approximate aggregate result. We prove that when the half width is small enough to satisfy a certain condition, then the relative error would be bounded by a user predefined error bound. Here, we predefine the confidence level as 95% and error bound as 1%. Since the approximate aggregate result after 1st round estimation is covered by a large confidence interval, it indicates the relative error is quite large now, **so we need to** continue sampling and estimation. With more sample is collected, we can refine the approximate aggregate result with a tight confidence interval. In this case, we can terminate this query after 4 rounds. Finally, we get an approximate aggregate result with a small range of confidence interval. And we found that the relative error is bounded by the user predefined error bound 1%.

#### Page 8

Thanks for watching this vedio!