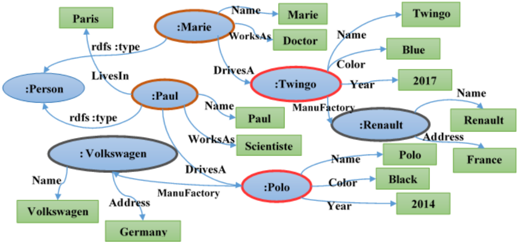
Schema Discovery in Large Web Data Sources

Redouane Bouhamoum, Zoubida Kedad, Stéphane Lopes

Journée thématique
EGC et IA
Université Paris Sud, CNRS, Université Paris Saclay, France,
May 10, 2019

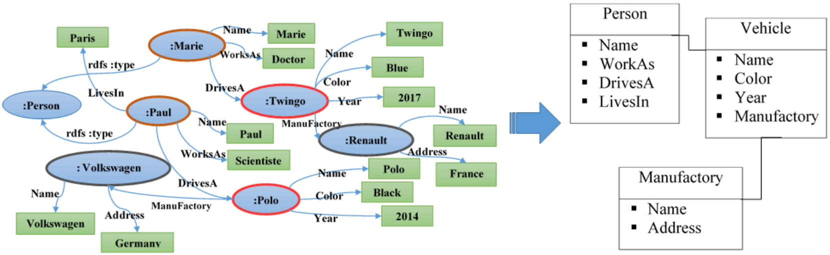
Context: Semantic Web Data

- Increasing number of datasets published in languages proposed by the W3C (RDF(s)/OWL)
    - Represented by triples $< S, P, V >$
    - Contain the data and the schema
- Difficult exploitation of these datasets
    - Incomplete or missing schema
    - Data do not always follow the schema



Our Goal: Toward a Scalable Schema Discovery Approach

- Our goal is to automatically discover the underlying schema given an RDF dataset
- Descriptive schema for the entities within a dataset
- Ensuring the scalability of our approach
  - Implement our proposal using a big data technology

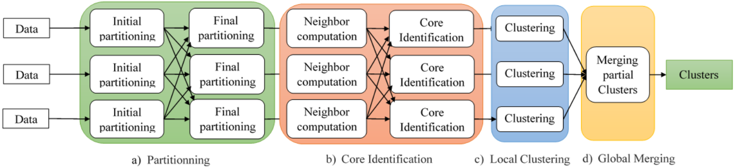


General Principle

- Grouping similar entities into clusters
- Similar entities are those having common properties
  - Evaluated using Jaccard Index: $J(e_i, e_j) = \frac{|e_i \cap e_j|}{|e_i \cup e_j|}$
  - ϵ similarity threshold
- A cluster represents a class in the descriptive schema
- SC-DBSCAN, Density-Based clustering algorithm inspired by DBSCAN
  - Scalable schema discovery approach
  - Implemented using Spark
  - Provides the same results as the sequential DBSCAN

Overview of Our Approach (SC-DBSCAN)

- Partitioning the data
- Identifying the cores
- Computing the partial clusters
- Merging the partial clusters



- The entities are distributed over the calculating nodes according to the properties
- Partitions
  - A partition $part_{p_x}$ is a subset containing entities described by the property p_x
- The question is how to assign entities to partitions ?

Data Partitioning

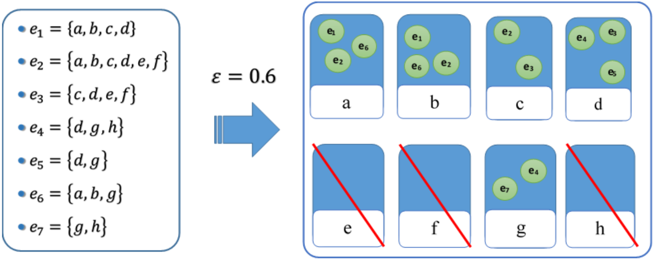
Entity assignment

- Naive assignment
  - Basically, assign an entity e to a partition $part_{p_x}$ if e is described by p_x
  - This assignment ensures that all similar entities are compared
  - Many meaningless comparisons
- Optimized assignment
  - Assign the entities to a minimum number of partitions ensuring all similar entities are compared
  - Reduce the number of partitions
  - Reduce the number of entities in each partition
  - Skip more meaningless comparisons

Data Partitioning

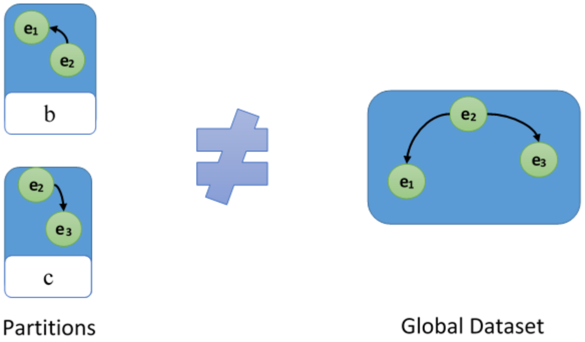
Optimized assignment

- An entity e_j is similar to e_i if they share at least $|e_i| * \epsilon$ properties
  - $\frac{|e_i \cap e_j|}{|e_i \cup e_j|} \geqslant \epsilon \iff |e_i \cap e_j| \geqslant |e_i \cup e_j| * \epsilon$
  - $|e_i \cup e_j| \times \epsilon \geqslant |e_i| * \epsilon \implies |e_i \cap e_j| \geqslant |e_i| * \epsilon$
- Dissimilarity threshold $k_{e_i} = |e_i| - (\lceil |e_i| * \epsilon \rceil) + 1$
- Assigning an entity e_i to k_{e_i} chosen partitions
  - Reduces the duplication
  - Ensures comparing e_i with all its neighbors

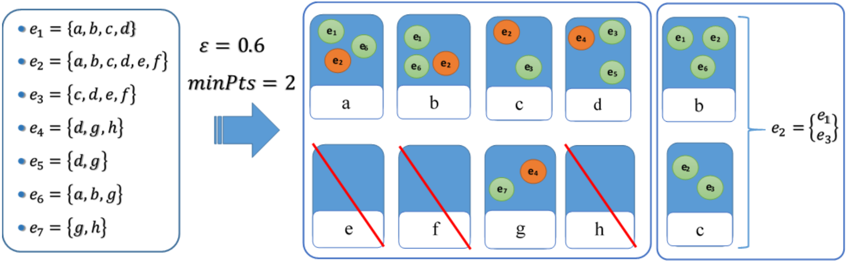


Core Identification

- An entity is a *core entity* if the number of entities in its ϵ-neighborhood is greater than *minPts*.
  - *minPts* density threshold
  - ϵ similarity threshold
- The neighborhood of an entity may span across several partitions

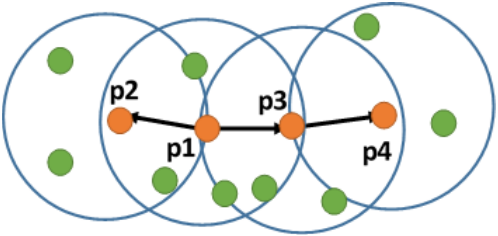


- Neighborhood computation
  - The neighbors of each entity in each partition are computed in parallel
  - Merge for each entity the lists of its neighbors
- The entities having a number of neighbors greater than *minPts* are cores



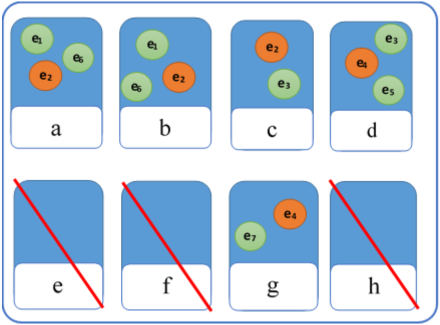
Partial Clustering

- For each core entity e
  - A cluster C that contains e and its neighbors is created
  - Recursively the neighbors of the cores in C are added to C



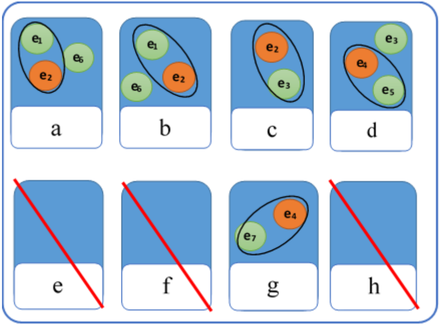
Partial Clustering

- For each core entity e
  - A cluster C that contains e and its neighbors is created
  - Recursively the neighbors of the cores in C are added to C



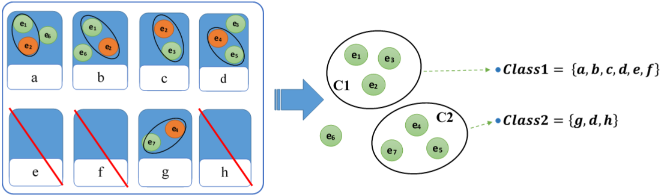
Partial Clustering

- For each core entity e
  - A cluster C that contains e and its neighbors is created
  - Recursively the neighbors of the cores in C are added to C



Merging Partial Clusters

- The partial clusters having a core entity in their intersection are merged
- Each resulting cluster represents a class of the descriptive schema

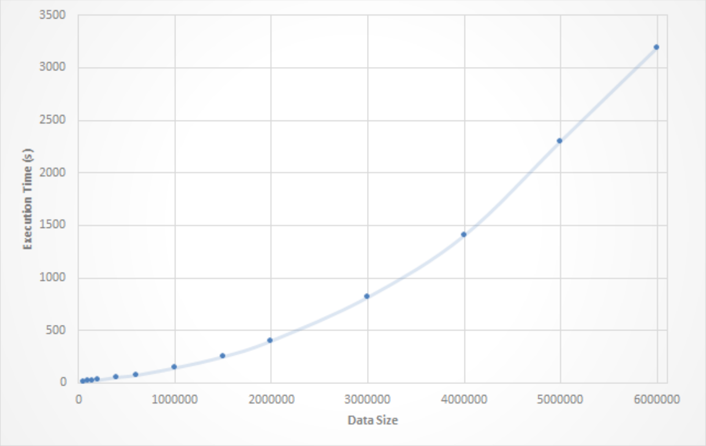


Evaluating our Approach

- Scalability of the clustering
  - Execution time
  - Using synthetic datasets [IBM Quest Synthetic Data Generator]
- Environment
  - Ubuntu Linux, Apache Spark 2.0
  - Scala
  - 5 nodes (1 master and 4 slaves), 30 GB of RAM and 12 Core CPU

Scalability of the Clustering

- Evaluating the similarity using Jaccard Index
- Parameters: $\epsilon = 0.8$, $minPts = 3$



Existing Approaches for Discovering the Structure of a Dataset

- Schema discovery using clustering algorithms
  - Cluster similar entities into classes that form the schema
  - Do not scale-up [K. K-Menouer, Z.Kedad, TLKDS 2016, K.Christodoulou et al., TLKDS 2013]
- Schema discovery for big data
  - Grouping entities having the same type declaration and propose a descriptive schema [M.Baazizi et al., EDBT 2017, D.Ruiz et al., ER 2015]
  - Not suitable when the schema is incomplete or missing
- Scalable versions of DBSCAN
  - Duplicating the whole datasets in all the calculating nodes is too costly [M.Patwary et al., SC 2012]
  - Some approaches are probabilistic and do not provide the same result as DBSCAN [G. Luo et al., BDCloud 2016, I. Savvas et al., WETICE 2016, A. Lulli et al., VLDB 2016]
  - Because of the high dimensionality of web data, the algorithms that require to order the data or partitioning the data using methods such as BSP are not efficient [D. Han et al., IPDPS 2016, Y. HE et al., IPDPS 2013]

Conclusion

- Contribution towards the scalability of schema discovery
  - Extracting a descriptive schema in large RDF datasets
  - Facilitating RDF datasets exploitation
- SC-DBSCAN: a novel distributed clustering algorithm
  - Implemented using big data technology
  - Providing the same clustering result as DBSCAN
- Key ideas of SC-DBSCAN
  - Partitioning according to properties
  - Parallelize the clustering

Future Works

- Perform more experiments on SC-DBSCAN
  - Number of properties describing the data
  - The size of the entities
  - Use Spark clusters of different configurations
- Study the evolution issues
  - Update the schema


tusind tak
谢谢
dakujem vám
ありがとう
ngiyabonga
dziękuję
merci
thank
you
baie dankie
धन्यवाद
molte grazie
suksema
danke
gracias
obrigada
takk
obrigado
teşekkür ederim
شكرا
gràcies
tänan
tack så mycket
dank u
teşekkür edirə
mahalo

Quality Evaluation

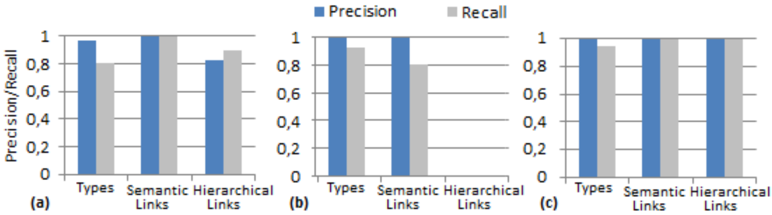


Figure: Evaluation of Schema Discovery in Conference (a) BNF (b) and DBpedia (c).