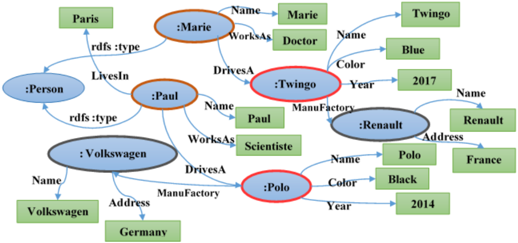
Schema Discovery in Large Web Data Sources

Redouane Bouhamoum, Zoubida Kedad, Stéphane Lopes

AWD 2019
Workshop on web data
Metz, France, January 22, 2019

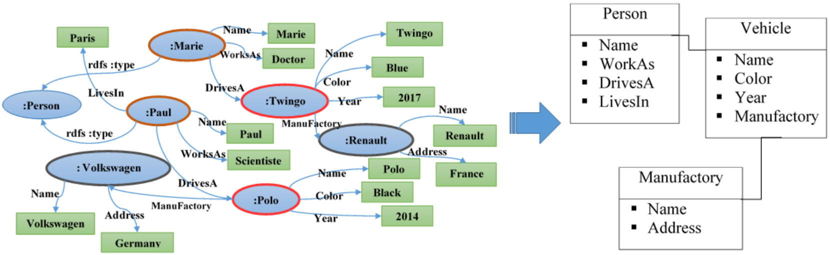
Context: Semantic Web Data

- 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

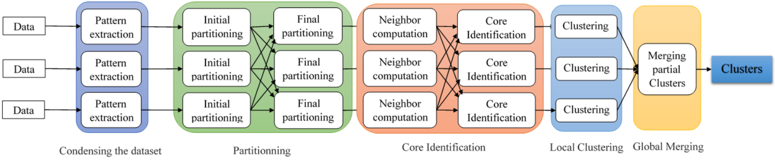


General Principle

- Grouping similar entities into clusters
- Similar entities are those having common properties
- 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

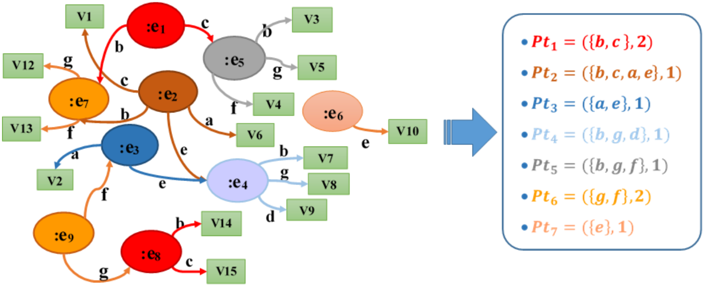
Overview of Our Approach (SC-DBSCAN)

- Building a condensed representation of the data
- Partitioning the data
- Identifying the cores
- Computing the partial clusters
- Merging the partial clusters



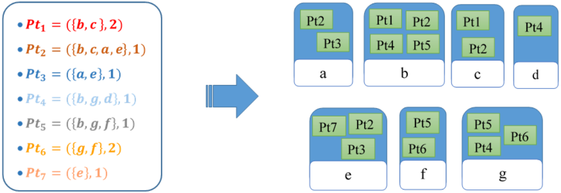
Building a Condensed Representation of an RDF Dataset

- Extracting a set of patterns representing the structure of the entities of the dataset
- Reducing the number of inputs for the clustering algorithm
- A pattern P is a set of distinct properties such that there exists at least one entity which property set is equal to P



Data Partitioning

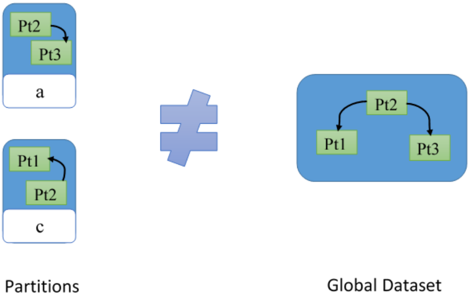
- The patterns are distributed over the calculating nodes
  - One partition is created for each property
  - Patterns are partitioned according to the properties
- A partition $part_{p_x}$ is a subset containing all the patterns described by the property p_x
- This partitioning ensures that all similar patterns are compared
  - Patterns that are never together in one partition are not similar



Core Identification

- A pattern is a *core pattern* if the sum of its number of entities and the number of entities of its neighbors in the ϵ-neighborhood is greater than *minPts*.
    - *minPts* density threshold
    - ϵ similarity threshold

- The neighborhood of a pattern may span across several partitions



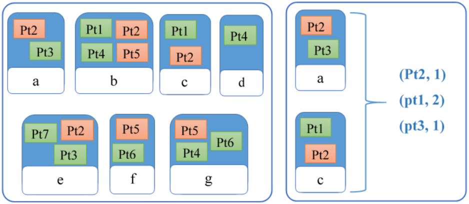
Core Identification

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

- $Pt_1 = (\{b, c\}, 2)$
- $Pt_2 = (\{b, c, a, e\}, 1)$
- $Pt_3 = (\{a, e\}, 1)$
- $Pt_4 = (\{b, g, d\}, 1)$
- $Pt_5 = (\{b, g, f\}, 1)$
- $Pt_6 = (\{g, f\}, 2)$
- $Pt_7 = (\{e\}, 1)$

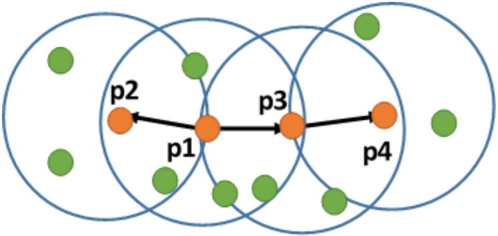
$minPts = 4$

$\varepsilon = 0.5$



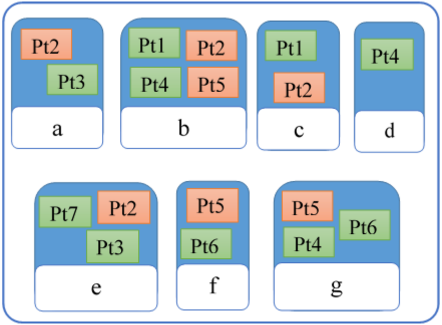
Partial Clustering

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



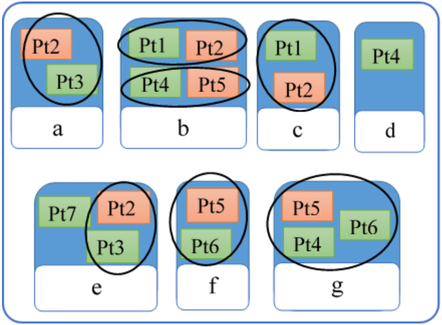
Partial Clustering

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



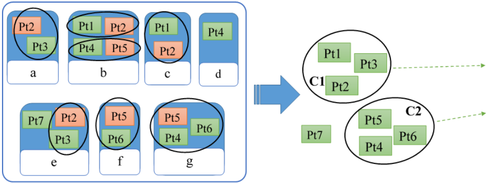
Partial Clustering

- For each core pattern *pt*
  - A cluster *C* that contains *pt* 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 pattern in their intersection are merged
- Each resulting cluster represents a class of the descriptive schema



- $Class1 = \{a, b, c, e\}$
- $Class2 = \{b, f, g, d\}$

Evaluating our Approach

- Pattern extraction
  - Size of the condensed representation
  - Execution time
  - Using DBpedia, DBLP, Katrina and Charley
- 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

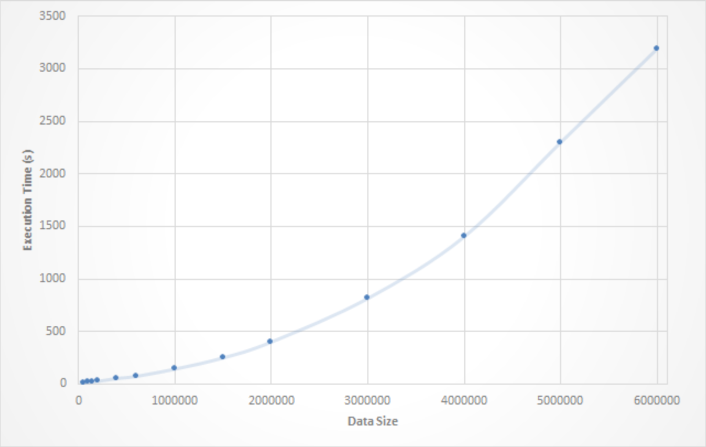
Size of the Condensed Representation

Inputs			Results	
Dataset	**Triples**	**Entities**	**Patterns**	**time (s)**
DBpedia	9 500 000 000	66 195 296	1 918 480	744
DBLP	222 375 855	16 086 516	351	120
Katrina	203 386 049	3 409	37	100
Charley	101 956 760	3 353	52	61

- DBLP, Katrina and Charley: the number of patterns is small
- DBpedia: the number of patterns is still high
  - Large set of properties (in average 150 for an entity)
  - Very heterogeneous property sets

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
  - Building a condensed representation of the dataset
  - Partitioning according to properties

Future Works

- Perform more experiments on SC-DBSCAN
  - Number of properties describing the data
  - The size of the patterns
  - 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
suksema
danke
thank
you
baie dankie
धन्यवाद
molte grazie
gracias
obrigada
takk
dank u
teşekkür edire
mahalo
obrigado
gràcies
teşekkür ederim
شكرا
tänan
tack så mycket