Lab - Key Discovery in Graph data -

Computer Science Master 2 – Data Science – Paris Saclay University

January 17, 2025

1 Key discovery for data linking

In table 1 we give an extract of some film descriptions. These films are described by five properties {title, hasActor, rDate, director, lang}.

	title	**hasActor***	**rDate**	**director***	**lang**
i_1	Ocean's 11	J. Roberts; B. Pitt;	2001	S. Soderbergh	
i_2	Ocean's 11	J. Roberts; B. Pitt; G. Clooney	2001	P.Greengrass	en
i_3	Ocean's 13	B. Pitt; G. Clooney	2007		
i_4	The descendants	N. Krause; G. Clooney	2011	A. Payne	en
i_5	Bourne Identity	M. Damon; F. Potente	2011	P. Greengrass, J. Cameron	en
i_6	Ocean's twelve	J. Roberts; B. Pitt; G. Clooney	2004	P.Greengrass	

Table 1: Extract of film descriptions data (D1)

Given these data if we apply SAKey, a key discovery tool that allows to discover n-almost keys:

- *Question 1.* Give a minimal 0−almost keys composed of two properties that can be discovered from D1.
- *Question 2.* Give a minimal 3-almost key that can be discovered from D1.
- *Question 3.* Give the minimal S-Keys, F-Keys and SF-keys with 2 excpetions that can be discovered in the data presented in Table 1 ?
- *Question 4.* If we declare a key $K = hasKey(c(OP_1, \ldots, OP_m)(DP_1, \ldots, DP_n))$, its OWL 2 semantics is expressed as follows:

$$hasKey(c(OP_1, \ldots, OP_m)(DP_1, \ldots, DP_n)) : \forall x, z_1, \ldots, z_m, w_1, \ldots, w_n, (C(x) \wedge C(y)) \wedge$$

$$\bigwedge_{(1 \leq i \leq m)} OP_i(x, z_i) \wedge_m OP_i(y, z_i) \bigwedge_{(1 \leq j \leq n)} DP_j(x, w_j) \wedge_n DP_j(y, z_j) \rightarrow sameAs(x, y)$$

The SF-key semantics is given in the following logical expression:

$$hasKey(c(OP_1, \ldots, OP_m)(DP_1, \ldots, DP_n)) : \forall x \forall y (C(x) \wedge C(y)) \wedge$$

$$\bigwedge_{(1 \leq i \leq m)} (z_i \; OP_i(x, z_i) \rightarrow OP_i(y, z_i)) \wedge \bigwedge_{(1 \leq j \leq n)} (\forall w_j \; DP_j(x, w_j) \rightarrow DP_j(y, w_j)) \rightarrow sameAs(x, y)$$

Let consider a key $K_1 = hasKey(Film)(hasActor, director)$.

Let D_2 be a second dataset given in Table 2. What would be the `sameAs` links that can be inferred when applying K_1 to $D_1 \times D_2$:

- S-Key semantics (i.e OWL 2 semantics)
- SF-Key semantics

	title	**hasActor***	**rDate**	**director***	**lang**
i_{12}	Ocean's 11	J. Roberts; B. Pitt;	2001	S. Soderbergh	
i_{22}	Ocean's 12	J. Roberts; B. Pitt; G. Clooney	2004	S. Soderbergh; P.Greengrass	
i_{32}	Ocean's 13	B. Pitt; G. Clooney	2007	S. Soderbergh; P.Greengrass	
i_{52}	Bourne Identity		2002	P. Greengrass	en
i_{62}	Ocean's twelve	J. Roberts; B. Pitt; G. Clooney	2004		

Table 2: Extract of film descriptions data (D2)

2 Hands on key discovery

This part is dedicated to the use of some existing key discovery tools. The idea is to run them, analyse the results and compare their results.

You can find at the following link the needed material for this handson session: `https://ecampus.paris-saclay.fr/course/view.php?id=135920`

You are asked to deliver your work on sub-sections 2.2, 2.3 and section 3 (as a bonus). The submission link is : `https://ecampus.paris-saclay.fr/mod/assign/view.php?id=1921356`

We will use SAKey that discovers n-almost keys with $n-1$ is the number of allowed exceptions for a set of properties to be a key. Bellow, is the readme file.

```
------------------------------------------------------------
SAKey takes as input two parameters:
a) the file where you want the key discovery to take place
b) the number of exceptions n.
Note that the n will lead to the discovery of
n-non keys and then the extraction of (n-1)-almost keys

To run SAKey use the following command:
java -jar "jar_fullpath"/SAKey.jar "file_fullpath" n

For example, in the following command:
java -jar /Users/danai/SAKey.jar  /Users/danai/datasets/DB_Lake.nt 1

In this example, SAKey will discover first 1-non keys and then derive 0-almost keys
for the file /Users/danai/datasets/DB_Lake.nt.
```

```
The file has to be in a n-triple format.
For example:
<http://www.okkam.org/oaie/restaurant1-Restaurant56>
<http://www.okkam.org/ontology_restaurant1.owl#phone_number> "718/522-5200" .
Or
http://www.okkam.org/oaie/restaurant1-Restaurant56
http://www.okkam.org/ontology_restaurant1.owl#phone_number
718/522-5200

-------------------------------------------------------------
```

2.1 SAKey on a small dataset

1. run SAKey on the dataset "`OAEI_2011_Restaurant_1.nt`" with $n = 1$, then copy the results that are displayed in the terminal and saved them in a text file `sakey-keys-restaurant-n1.txt`.
2. run SAKey on the dataset "*OAEI_2011_Restaurant_1.nt*" with $n = 7$, then copy the results that are displayed in the terminal and saved them in a text file `sakey-keys-restaurant-n7.txt`.

Compare the obtained keys, in terms of number of keys and minimality (i.e. the number of properties involved in the keys).

2.2 SAKey on a bigger dataset from DBpedia

1. run SAKey on the dataset "`library-6k.nt`" with $n = 1$, and redirect the results to a text file `sakey-keys-library-6k-n1.txt`.
2. run SAKey on the dataset "`library-196k.nt`" with $n = 1$, and redirect the results to a text file `sakey-keys-library-196k-n1.txt`.

Compare the results in terms of number of keys and non-keys. If no keys found, try with a higher value of n.

2.3 Use the keys for data linking

Using the keys that you obtained on dbpedia `library-6k.nt` and by following the SPARQL query given as example bellow, write SPARQL queries for keys that you choose and run them on `http://yasgui.org/`.

Bellow an example of a SPARQL Query for a key `hasKey(Book(rdfs:label)())`. NB: the limit 1 is important to have reasonable time execution of the query.

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>

# everything that starts with the # character is a comment
# INPUT-1: one key detected in DBpedia for the class (e.g Book)
# In this example query, the ranked key contains only the property rdfs:label

SELECT DISTINCT ?dbpediaID1 ?dbpediaID2
WHERE {
    # first book entity
      ?dbpediaID1 rdf:type dbo:Book  .
      ?dbpediaID1 rdfs:label ?labelDBP1 .
      ?dbpediaID2 rdf:type dbo:Book  .
      ?dbpediaID2 rdfs:label ?labelDBP2 .
        FILTER(STR(?labelDBP1) = STR(?labelDBP2))
      }
} limit 1
```

3 Going further

4 Going further

Redo the exercise 2.1 and 2.2 using the three tools sakey, Vickey and Rocker and compare the results in terms of :

- number of keys,
- run-time,
- if possible, the average size of keys

For more details on how to run Vickey, please see here: `https://github.com/lgalarra/vickey/`
This is an example of how to use vickey:

```
java -jar ./Lab/vickey.jar -mins 5 -p ./Sakey-handson-materials/datasets/DB_Lake.nt.
```

```
-------------- readme -------------
To use vickey.jar, run in a command line:

$ java -jar vickey.jar -mins <support-threshold> [-p] <input-file>

<support-threshold>: minimal number of entities that must be satisfied by the conditional
key in order to be reported.

-p : optional argument. If present, it makes VICKEY interpret the support threshold
as a percentage coverage threshold instead of an absolute support number. The formula
is (support / #subjects)*100, that is, support 60 with the -p option enabled means that
a key will be reported if it covers AT LEAST 60% of all subjects in the dataset.

The list of unique conditional keys corresponds to the text output after the indicator
"==== Unique C-keys =====". VICKEY uses the non-keys reported by SAKey, which are stored
```

```
in a file reported by VICKEY.
--------------
```

For more details on how to run Rocker, please see here: `https://github.com/AKSW/rocker`