

Laboratory: Key Discovery

Pablo Mollá Chárlez

February 8, 2025

Contents

1	Key discovery for data linking	2
2	Solution	3
2.1	Question 1	3
2.2	Question 2	4
2.3	Question 3	5
2.4	Question 4	7
3	Hands-on Key Discovery	9
3.1	SAKey on a small dataset	9
3.2	SAKey on a bigger dataset from DBpedia	10
3.3	Use the keys for data linking	10
3.4	Going further	10
4	Solution	11
4.1	SAKey on a small dataset	11
4.2	SAKey on a bigger dataset from DBpedia	12
4.3	Use the keys for data linking	16
4.4	Going Further: Vickey & Rocker	20
4.4.1	Vickey	20
4.4.1.1	OAEL_2011_Restaurant_1.nt	20
4.4.1.2	Library-6k.nt	20
4.4.1.3	Library-196k.nt	20
4.4.2	Rocker	21
4.4.2.1	OAEL_2011_Restaurant_1.nt	21
4.4.2.2	Library-6k.nt	21
4.4.2.3	Library-196k.nt	22

1 Key discovery for data linking

In figure 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	

Figure 1: Extract of film descriptions data (D_1)

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 D_1 .
- **Question 2.** Give a minimal 3 – *almost* key that can be discovered from D_1 .
- **Question 3.** Give the minimal **S-Keys**, **F-Keys** and **SF-keys** with 2 exceptions that can be discovered in the data presented in figure 1?

If we declare a key $K = hasKey(c(OP_1, \dots, OP_m)(DP_1, \dots, DP_n))$, its OWL 2 semantics is expressed as follows:

$$\text{S-keys Rule: } hasKey(c(OP_1, \dots, OP_m)(DP_1, \dots, DP_n)) : \forall x, z_1, \dots, z_m, w_1, \dots, w_n, (C(x) \wedge C(y)) \wedge \bigwedge_{1 \leq i \leq m} OP_i(x, z_i) \wedge_m OP_i(y, z_i) \wedge \bigwedge_{1 \leq j \leq n} DP_j(x, w_j) \wedge_n DP_j(y, w_j) \rightarrow sameAs(x, y)$$

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

$$\text{SF-keys Rule: } hasKey(c(OP_1, \dots, OP_m)(DP_1, \dots, DP_n)) : \forall x \forall y (C(x) \wedge C(y)) \wedge \bigwedge_{1 \leq i \leq m} (\forall z_i (OP_i(x, z_i) \rightarrow OP_i(y, z_i))) \wedge_m OP_i(y, z_i) \wedge \bigwedge_{1 \leq j \leq n} (\forall w_j (DP_j(x, w_j) \rightarrow DP_j(y, w_j))) \rightarrow sameAs(x, y)$$

Let's consider a key $K_1 = hasKey(Film)(hasActor, director)$ and D_2 a second dataset given in table 2.

- **Question 4.** 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		

Figure 2: Extract of film descriptions data (D_2)

2 Solution

2.1 Question 1

Let's first start with the formal definition:

An n -almost key for a dataset means a set of attributes K that uniquely identifies the rows up to n "exceptions." Formally, one way to say this is:

- Look at all pairs of distinct rows (r_i, r_j) .
- A "collision" on K means that r_i and r_j have the **same values** for all attributes in K . Therefore, there are 2 exceptions considering both rows r_i and r_j .
- If the number of such exceptions is **at most n**, we say K is an n -almost key.

Basically, 0-almost key means no exceptions in the set of K properties.

For the provided dataset D_1 , here are some pairs of properties that turn out to be perfect (**no collisions**) on the 6 rows considering each one of the 3 possible semantics:

1. 0-almost S-Keys

- **(title, director)**: None of the 6 rows from those 2 properties are exactly the same. For instance, i_1 has **(Ocean's 11, S. Soderbergh)** and i_2 has **(Ocean's 11, P. Greengrass)** have different sets for the property **director**, so no exceptions at all. Besides, empty cells are considered different under optimistic approach. ✓
- **(title, lang)**: For instance, i_1 has **(Ocean's 11, Empty)** and i_2 has **(Ocean's 11, en)**. Even though they share one value in the property **title**, they differ in the property **lang**, therefore no exceptions. ✓
- **(rDate, director)**: For example, i_2 and i_6 both have the same (**director**) value but differ in **rDate** (2001 \neq 2004). ✓

Bear in mind as well that, the 3 pairs of 0-almost S-Keys provided are minimal as if you remove one of the 2 possible properties, the remaining property is not a 0-almost S-Key.

2. 0-almost F-Keys

- **(title, director)**: As previously, none of the 6 rows from those 2 properties are exactly the same even when empty cells are considered equal under pessimistic approach. For instance, i_2 and i_3 in the **director** property could be equal (**P. Greengrass**) however, even in such case we would have that **Ocean's 11 \neq Ocean's 13**. ✓
- **(title, hasActor)**: None of the 6 rows from those 2 properties are exactly the same sets. For instance, i_1 has **(Ocean's 11, {J. Roberts, P. Pitt})** and i_2 has **(Ocean's 11, {J. Roberts, P. Pitt, G. Clooney})**, (they are considered equal under S semantics but with F semantics they are different because the sets are different), they have different sets for the property **hasActor**, so no exceptions at all. ✓
- **(hasActor, rDate)** has 0 exceptions under F semantics, therefore the 2-set property is a 0-almost F-key.

Once again, the chosen pairs of properties are minimal, as removing one property leaves the remaining property not being a 0-almost F-Key.

3. 0-almost SF-Keys

- **(title, hasActor)**: None of the 6 rows from those 2 properties are exactly the same sets. For instance, i_1 has (**Ocean's 11**, {**J. Roberts, P. Pitt**}) and i_2 has (**Ocean's 11**, {**J. Roberts, P. Pitt, G. Clooney**}), they share a non-empty intersection (they are considered equal under S semantics but with F and SF semantics they are different because the sets are different), they have different sets for the property **hasActor**, so no exceptions at all. ✓
- **(title, director)**: As previously, none of the 6 rows from those 2 properties are exactly the same under the SF optimistic approach. For instance, i_2 and i_6 in the **director** property are equal (**P. Greengrass**) however, even in such case we would have that **Ocean's 11** \neq **Ocean's twelve**. ✓

Once again, the chosen pairs of properties are minimal, as removing one property leaves the remaining property not being a 0-almost SF-Key.

2.2 Question 2

Basically, a 3-almost key means we allow up to 3 exceptions (0-almost, 1-almost and 2-almost \subseteq 3-almost). Notice that if a set of columns has 0 or 1 or 2 collisions, it automatically qualifies as a 3 – *almost* key, because “at most 2 collisions” is certainly “at most 3.” For instance, the minimal 3 – *almost* key that can be considered are, depending on the semantic, the following:

• 3-almost S-Keys

- **title** alone has 2 exceptions, therefore it is a 2-almost S-Key, thus a 3-almost S-Key. The proposed key is minimal because it's a single set key. ✓
- **rDate** alone has exactly 4 exceptions (the rows i_1, i_2, i_4 , and i_5), so that is a 4-almost S-Key and doesn't qualify as 3-almost S-Key, and the proposed key is minimal because it's a single set key. ✗
- **director** alone also has 3 exceptions (i_2, i_5, i_6), therefore is a 3-almost S-Key. The proposed key is minimal because it's a single set key. ✓
- **lang** alone, on the other hand, has 3 exceptions (the three “en” rows among themselves), therefore is a 3 – *almost* key. ✓
- **hasActor** alone has 5 exceptions (i_1, i_2, i_3, i_4, i_6) therefore it is a 5-almost S-Key, and, even though minimal, it doesn't qualify as 3-almost S-Key. ✗
- **(director, lang)** has 2 exceptions (i_2, i_5), therefore the 2-set property is a 2-almost S-key and consequently 3-almost S-key. ✓

• 3-almost F-Keys

- **title** alone has 2 exceptions, therefore it is a 2-almost F-Key, thus a 3-almost F-Key. The proposed key is minimal because it's a single set key. ✓
- **director** alone also has 3 exceptions (i_2, i_6 and possibly i_6), therefore is a 3-almost S-Key. The proposed key is minimal because it's a single set key. ✓

• 3-almost SF-Keys

- **title** alone has 2 exceptions, therefore it is a 2-almost SF-Key, thus a 3-almost SF-Key. The proposed key is minimal because it's a single set key. ✓
- **director** alone also has 2 exceptions (i_2, i_6), therefore is a 2-almost SF-Key, thus a 3-almost SF-Key. The proposed key is minimal because it's a single set key. ✓
- **(director, lang)** has 0 exceptions under SF semantics, therefore the 2-set property is a 0-almost SF-key and consequently 3-almost SF-key. ✓
- **(title, rDate)** has 2 exceptions under SF semantics (i_1, i_2), therefore the 2-set property is a 2-almost SF-key and consequently 3-almost SF-key. ✓

2.3 Question 3

Let's first define the 3 mentioned concepts:

- **S-Key:** A set of properties K is an S-Key if every row in the dataset has a unique combination of those properties (where a set-valued property is counted as the same if the two sets share at least one common element). Empty values are considered as different from the existing ones due to the optimistic approach.
- **F-Key:** A set of properties K is an F-Key if every row in the dataset has a unique combination of those properties (where a set-valued property is counted as the same if the two sets are exactly identical). Empty values are considered as possibly identical from the existing ones due to the pessimistic approach.
- **SF-Key:** A set of properties K is an SF-Key if every row in the dataset has a unique combination of those properties (where a set-valued property is counted as the same if the two sets are exactly identical). Empty values are considered as different from the existing ones due to the optimistic approach.

	FirstName	LastName	SSN	StudiedIn	HasSibling
p1	Marie	Brown	121558745	UCC, Yale	p2, p4
p2	John	Brown	232351234	–	p1, p4
p4	Marie	Roger	767960154	UCC, UCD	–
p4	Marc	Brown	–	UCD	p1, p2
p5	Helen	Roger	967960158	–	–

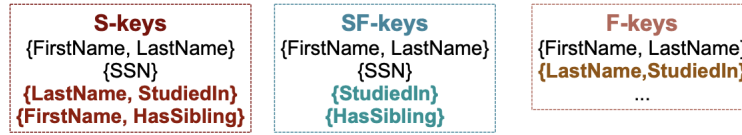


Figure 3: S-Keys, F-Keys and SF-Keys Example

In the previous theoretical example, there I believe there is a mistake as **{LastName, StudiedIn}** is not a 0-almost F-Key as p_1 and p_2 could be equal with the empty cell. I think instead of "**{LastName}**" it was meant to be "**{FirstName}**". Now, we proceed to answer the question, and we take into account that we can consider 2 exceptions, which is equivalent to say 2 – *almost* keys.

- **S-Keys**
 - **title** alone has 2 exceptions, therefore it is a 2-almost S-Key. The proposed key is minimal because it's a single set key. ✓
 - **hasActor** alone has 5 exceptions (i_1, i_2, i_3, i_4, i_6) therefore it is a 5-almost S-Key, and, even though minimal, it doesn't qualify as with 2 exceptions but would for at least 2 exceptions. ✗
 - **rDate** alone has exactly 4 exceptions (the rows i_1, i_2, i_4 , and i_5), so that is a 4-almost S-Key and doesn't qualify as with 2 exceptions, but would qualify for at least 2 exceptions. The proposed key is minimal because it's a single set key. ✗
 - **director** alone also has 3 exceptions (i_2, i_5, i_6), therefore is a 3-almost S-Key and wouldn't qualify for a S-Key with 2 exceptions but would for at least 2. The proposed key is minimal because it's a single set key. ✗
 - **lang** has 3 exceptions (the three "en" rows among themselves), therefore is a 3 – *almost* key and doesn't satisfy the condition but would verify the at least 2 exceptions. ✗
 - **(title, hasActor)** has 2 exceptions under S semantics (i_1, i_2), therefore the 2-set property is a 2-almost S-key. ✓

- **(title, rDate)** has 2 exceptions under S semantics (i_1, i_2) , therefore the 2-set property is a 2-almost S-key. ✓
- **(hasActor, rDate)** has 2 exceptions under S semantics (i_1, i_2) , therefore the 2-set property is a 2-almost S-key. ✓
- **(hasActor, director)** has 2 exceptions under S semantics (i_2, i_6) , therefore the 2-set property is a 2-almost S-key. ✓
- **(hasActor, lang)** has 2 exceptions under S semantics (i_2, i_4) , therefore the 2-set property is a 2-almost S-key. ✓
- **rDate, director** has 0 exceptions, therefore is a 0-almost key and doesn't satisfy the condition of 2 exceptions. ✗
- **(rDate, lang)** has 2 exceptions under S semantics (i_4, i_5) , therefore the 2-set property is a 2-almost S-key. ✓
- **(title, director)**: In question 1 we proved that is a 0-almost Key, so it doesn't satisfy the condition. ✗
- **(director, lang)** has 2 exceptions under S semantics (i_2, i_5) , therefore the 2-set property is a 2-almost S-key. ✓

• F-Keys

- **title** alone has 2 exceptions, therefore it is a 2-almost F-Key. The proposed key is minimal because it's a single set key. ✓
- **hasActor** alone has 2 exceptions (i_2, i_6) therefore it is a 2-almost F-Key, and minimal. ✓
- **rDate** alone has exactly 4 exceptions (the rows i_1, i_2, i_4 , and i_5), so that is a 4-almost F-Key and doesn't qualify as with 2 exceptions, but would qualify for at least 2 exceptions. The proposed key is minimal because it's a single set key. ✗
- **director** alone also has 3 exceptions considering the empty cell (i_2, i_3, i_6) , therefore is a 3-almost F-Key and wouldn't qualify for a F-Key with 2 exceptions but would for at least 2. The proposed key is minimal because it's a single set key. ✗
- **lang** has 6 exceptions (the three "en" rows among themselves and the three empty cells which could be equal to "en"), therefore is a 6-almost F-key and doesn't satisfy the condition but would verify the at least 2 exceptions. ✗
- **(title, hasActor)** has 0 exceptions under F semantics, therefore the 2-set property is a 0-almost F-key. ✗
- **(title, rDate)** has 2 exceptions under F semantics (i_1, i_2) , therefore the 2-set property is a 2-almost F-key. ✓
- **(hasActor, rDate)** has 0 exceptions under F semantics, therefore the 2-set property is a 0-almost F-key. ✗
- **(hasActor, director)** has 2 exceptions under F semantics (i_2, i_6) , therefore the 2-set property is a 2-almost F-key. ✓
- **(hasActor, lang)** has 2 exceptions under F semantics (i_2, i_6) , therefore the 2-set property is a 2-almost F-key. ✓
- **rDate, director** has 0 exceptions, therefore is a 0-almost F-key and doesn't satisfy the condition of 2 exceptions. ✗
- **(rDate, lang)** has 4 exceptions under S semantics (i_1, i_2, i_4, i_5) , therefore the 2-set property is a 4-almost F-key and doesn't satisfy the 2 exceptions condition. ✗
- **(title, director)**: In question 1 we proved that is a 0-almost F-Key, so it doesn't satisfy the condition. ✗

- (**director**, **lang**) has 2 exceptions under F semantics (i_2, i_6), therefore the 2-set property is a 2-almost F-key. ✓
- SF-Keys
 - **title** alone has 2 exceptions, therefore it is a 2-almost SF-Key. The proposed key is minimal because it's a single set key. ✓
 - **hasActor** alone has 2 exceptions (i_2, i_6) therefore it is a 2-almost SF-Key, and minimal. ✓
 - **rDate** alone has exactly 4 exceptions (the rows i_1, i_2, i_4 , and i_5), so that is a 4-almost SF-Key and doesn't qualify as with 2 exceptions, but would qualify for at least 2 exceptions. The proposed key is minimal because it's a single set key. ✗
 - **director** alone also has 2 exceptions considering the empty cell (i_2, i_6), therefore is a 2-almost SF-Key. The proposed key is minimal because it's a single set key. ✓
 - **lang** has 3 exceptions (the three “en” rows among themselves), therefore is a 3-almost SF-key and doesn't satisfy the condition but would verify the at least 2 exceptions. ✗
 - (**title**, **hasActor**) has 0 exceptions under SF semantics, therefore the 2-set property is a 0-almost SF-key. ✗
 - (**title**, **rDate**) has 2 exceptions under SF semantics (i_1, i_2), therefore the 2-set property is a 2-almost SF-key. ✓
 - (**hasActor**, **rDate**) has 0 exceptions under SF semantics, therefore the 2-set property is a 0-almost SF-key. ✗
 - (**hasActor**, **director**) has 2 exceptions under SF semantics (i_2, i_6), therefore the 2-set property is a 2-almost SF-key. ✓
 - (**hasActor**, **lang**) has 0 exceptions under SF semantics, therefore the 2-set property is a 0-almost SF-key. ✗
 - **rDate**, **director** has 0 exceptions, therefore is a 0-almost SF-key and doesn't satisfy the condition of 2 exceptions. ✗
 - (**rDate**, **lang**) has 2 exceptions under S semantics (i_4, i_5), therefore the 2-set property is a 2-almost F-key. ✓
 - (**title**, **director**): In question 1 we proved that is a 0-almost SF-Key, so it doesn't satisfy the condition. ✗
 - (**director**, **lang**) has 0 exceptions under SF semantics, therefore the 2-set property is a 0-almost F-key, thus it doesn't qualify. ✗

2.4 Question 4

Let's first explain both rules in order to answer the question by providing a recap of the symbology meaning:

- c : A class name (e.g., ‘Film’).
- OP_i : An object property (e.g., ‘hasActor’, ‘director’), for $i = 1, \dots, m$.
- DP_j : A data property (e.g., ‘title’, ‘rDate’), for $j = 1, \dots, n$.
- x, y : Arbitrary individuals (resources) in the ontology.
- z_i : A potential object-property filler—another individual—used for testing whether $OP_i(x, z_i)$ holds.
- w_j : A potential data-property filler—typically a literal value—used for testing whether $DP_j(x, w_j)$ holds.

The question requires to apply such rules to a specific set of properties $K_1 = \{\text{hasActor}, \text{director}\}$ (which makes sense as **hasActor** is a object property and **director** a data property) and to the datasets $D_1 \times D_2$: Let's proceed studying each axiom rule by itself:

	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	

	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		

Figure 4: Datasets D_1, D_2

- **S-Keys Axiom Rule:** Then, considering for example rows $x = i_1$ and $y = i_{12}$, we have that both belong to the same class **Film** ($\leftrightarrow C(x), C(y)$), the object property **hasActor** associates to the instance/row i_1 the set of values $\{J.Roberts, B.Pitt\}$ and the object property **hasActor** associates to the instance/row i_{12} the same set of values $\{J.Roberts, B.Pitt\}$ (z_i in the axiom rule), and the data property **director** associates to both, i_1 and i_{12} , the same value $S.Soderbergh$, then we can conclude that the instances $x = i_1$ and $y = i_{12}$ are the same ($\leftrightarrow \text{sameAs}(x, y)$).

We can as well identify the following **sameAs** links:

- **sameAs**(i_1, i_{12}): ✓
- **sameAs**(i_1, i_{22}): ✓
- **sameAs**(i_1, i_{32}): ✓
- **sameAs**(i_2, i_{22}): ✓
- **sameAs**(i_2, i_{32}): ✓
- **sameAs**(i_6, i_{22}): ✓
- **sameAs**(i_6, i_{32}): ✓

	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	

	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		

Figure 5: F-Keys sameAs Links - Some links only (Not All)

- **F-Keys Axiom Rule:** All the links are:

- `sameAs(i_1 , i_{12}):` ✓
- `sameAs(i_2 , i_{62}):` ✓
- `sameAs(i_3 , i_{32}):` ✓
- `sameAs(i_6 , i_{62}):` ✓
- `sameAs(i_6 , i_{52}):` ✓

- **SF-Keys Axiom Rule:** We would obtain:

- `sameAs(i_1 , i_{12}):` ✓
- `sameAs(i_3 , i_{32}):` ✗

3 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 hands-on session: [Ecampus](#)

You are asked to deliver your work on sub-sections 2.2, 2.3 and section 3 (as a bonus). The submission link is: [Submission Link](#).

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
```

3.1 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**.

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

3.2 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**.
3. Compare the results in terms of number of keys and non-keys. If no keys found, try with a higher value of n .

3.3 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 Yasgui. 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 foaf: <http://xmlns.com/foaf/0.1/>
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(?labelDBP1 != ?labelDBP2)
}
limit 1

```

3.4 Going further

Redo the exercise 3.1 and 3.2 using the three tools **SAKey**, **Vickey** and **Rocker** and compare the results in terms of:

- Number of keys,
- Run-time,
- The average size of keys (if possible)

For more details on how to run **Vickey**, please use the following Github Link and for futher information on **Rocker**, use this Rocker Link. This is an example of how to use **Vickey**:

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

```

----- READ ME -----
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.
-----

```

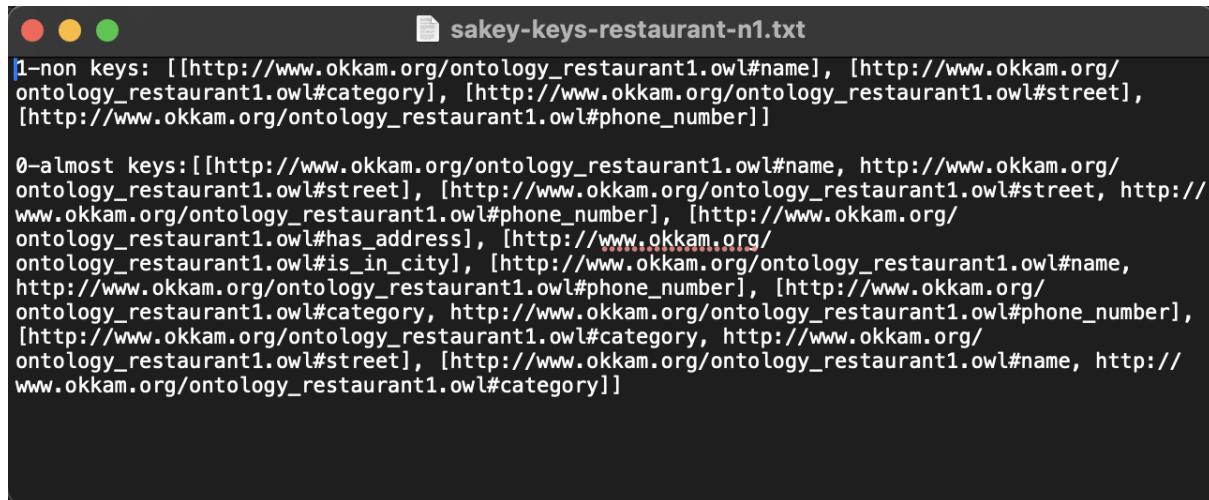
4 Solution

4.1 SAKey on a small dataset

- 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**.

Executing the following command produced the file in the figure 6:

```
java -jar ../Sakey-20250118/sakey.jar datasets/OAEI_2011_Restaurant_1.nt 1 >
sakey-keys-restaurant-n1.txt
```



```

sakey-keys-restaurant-n1.txt
1-non keys: [[http://www.okkam.org/ontology_restaurant1.owl#name], [http://www.okkam.org/ontology_restaurant1.owl#category], [http://www.okkam.org/ontology_restaurant1.owl#street], [http://www.okkam.org/ontology_restaurant1.owl#phone_number]]

0-almost keys: [[http://www.okkam.org/ontology_restaurant1.owl#name, http://www.okkam.org/ontology_restaurant1.owl#street], [http://www.okkam.org/ontology_restaurant1.owl#street, http://www.okkam.org/ontology_restaurant1.owl#phone_number], [http://www.okkam.org/ontology_restaurant1.owl#has_address], [http://www.okkam.org/ontology_restaurant1.owl#is_in_city], [http://www.okkam.org/ontology_restaurant1.owl#name, http://www.okkam.org/ontology_restaurant1.owl#phone_number], [http://www.okkam.org/ontology_restaurant1.owl#category, http://www.okkam.org/ontology_restaurant1.owl#phone_number], [http://www.okkam.org/ontology_restaurant1.owl#category, http://www.okkam.org/ontology_restaurant1.owl#street], [http://www.okkam.org/ontology_restaurant1.owl#name, http://www.okkam.org/ontology_restaurant1.owl#category]]

```

Figure 6: sakey-keys-restaurant-n1.txt

- 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**.

Executing the following command produced the file in the figure 7:

```
java -jar ../Sakey-20250118/sakey.jar datasets/OAEI_2011_Restaurant_1.nt 7 >
sakey-keys-restaurant-n7.txt
```

```

sakey-keys-restaurant-n7.txt
7-non keys: [[http://www.okkam.org/ontology_restaurant1.owl#name], [http://www.okkam.org/ontology_restaurant1.owl#category]]

6-almost keys:[http://www.okkam.org/ontology_restaurant1.owl#has_address], [http://www.okkam.org/ontology_restaurant1.owl#street], [http://www.okkam.org/ontology_restaurant1.owl#is_in_city], [http://www.okkam.org/ontology_restaurant1.owl#phone_number], [http://www.okkam.org/ontology_restaurant1.owl#name, http://www.okkam.org/ontology_restaurant1.owl#category]]

```

Figure 7: 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).

Let's first analyze the first file [sakey-keys-restaurant-n1.txt](#). According to [SAKey](#), we have obtained the following **4 single sets of properties which are not keys**:

[name] [category] [street] [phone_number]

On the other hand, the minimal sets of properties obtained that are almost keys (0 – *almost* keys) are:

[has_address] [is_in_city] [name, street] [street, phone_number]
[name, phone_number] [category, phone_number] [category, street] [name, category]

The total count for 0 – *almost* keys adds up to **2 single-property keys** and **6 two-property key sets**.

In the second file [sakey-keys-restaurant-n7.txt](#), we have obtained **2 non-keys**:

[name] [category]

On the other hand, the minimal sets of properties obtained that are almost keys (6 – *almost*) are:

[has_address] [is_in_city] [street] [phone_number] [name, category]

The total count for **6 – almost keys** adds up to **4 single-properties** and **1 two-property set**. Now, let's compare both results. The number of discovered keys for [sakey-keys-restaurant-n1.txt](#) is **8 keys** (0 – *almost* keys) and for [sakey-keys-restaurant-n7.txt](#) is **5 keys** (6 – *almost* keys). In terms of minimality (size of the key sets), we have that:

- [sakey-keys-restaurant-n1.txt](#): Although there are more keys in total, only 2 of them are single-property ([has_address], [is_in_city]); the rest are pairs.
- [sakey-keys-restaurant-n7.txt](#): Fewer total keys (only 5), but 4 of them are single-property.

4.2 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**.

Executing the following command produced the file in the figure 8:

- **3-properties:** 4 occurrences
- **4-properties:** 1 occurrence
- **5-properties:** 1 occurrence
- **6-properties:** 6 occurrences
- **7-properties:** 3 occurrences
- **9-properties:** 1 occurrence

On the other hand, the minimal sets of properties obtained that are almost keys ($0 - almost$ keys) are:

- **1-property:** 17 keys
- **2-properties:** 43 keys
- **3-properties:** 2 keys

The total count for $0 - almost$ keys adds up to **62 keys properties**.

In the second file [sakey-keys-library-196k-n1.txt](#), we have obtained 522 non-keys, following the distribution described:

- **1-property:** 3 occurrences
- **2-properties:** 12 occurrences
- **3-properties:** 24 occurrences
- **4-properties:** 38 occurrence
- **5-properties:** 55 occurrence
- **6-properties:** 55 occurrences
- **7-properties:** 69 occurrences
- **8-properties:** 74 occurrence
- **9-properties:** 39 occurrence
- **10 properties:** 35 occurrences
- **11 properties:** 33 occurrences
- **12 properties:** 20 occurrences
- **13 properties:** 24 occurrences
- **14 properties:** 14 occurrences
- **15 properties:** 11 occurrences
- **16 properties:** 13 occurrences
- **17 properties:** 2 occurrences
- **18 properties:** 1 occurrences

To find the previous distribution, the following Python code was implemented:

```
# Re-read the content and directly process the keys
with open(file_path, 'r') as file:
    data = file.read()

# Split the data to identify 1-non keys and count their occurrences
# First, we remove the part before the first key list
keys_section = data.split("1-non keys:")[1]

# Extract the 1-non keys by capturing all occurrences of the pattern '[]'
matches = re.findall(r'\[.*?\]', keys_section)

# Now, count the distribution of 1-non keys based on the number of properties
distribution = {}

for match in matches:
    # Clean the match and convert it to a Python list using eval
    match_cleaned = match.strip()[1:-1] # Remove leading and trailing brackets
    keys = match_cleaned.split(', ') # Split the keys by the comma and space
    num_properties = len(keys) # Number of properties
    if num_properties in distribution:
        distribution[num_properties] += 1
    else:
        distribution[num_properties] = 1

# Total number of 1-non keys
total_1non_keys = len(matches)

# Display the results
total_1non_keys, distribution
```


On the other hand, the minimal sets of properties obtained that are almost keys (*0-almost*) are distributed as follows:

- **1-properties:** 540 occurrences
- **2-properties:** 15798 occurrences
- **3-properties:** 2709 occurrences
- **4-properties:** 965 occurrences
- **5-properties:** 133 occurrences
- **6-properties:** 2 occurrences

The previous distribution adds up to **20147** which are **0-almost keys**. To find the previous distribution, the following Python code was implemented:

```
from collections import Counter

# Load the file content
file_path = '/mnt/data/zeroalmostkeys196.txt'
with open(file_path, 'r') as file:
    data = file.read()

# Parse the data to find sets of properties (each set is surrounded by square brackets [])
import re

# Extracting sets of properties
property_sets = re.findall(r'\[[^\]]+\]', data)
# Count the number of properties in each set
property_count = [len(props.split(',')) for props in property_sets]

# Calculate the distribution of property counts
property_distribution = Counter(property_count)

# Format the results as LaTeX
latex_distribution = "\n".join([
    f"\\item \\textbf{{{count}}-properties:}} \\$\\{occurrences\\}\\$ occurrences"
    for count, occurrences in sorted(property_distribution.items())
])

latex_distribution
```

Now, let's compare both results. The number of discovered keys for **sakey-keys-library-6k-n1.txt** is **62 keys** (*0-almost* keys) and for **sakey-keys-library-196k-n1.txt** is **20147 keys** (*0-almost* keys). In terms of minimality (size of the key sets), we have that:

- **sakey-keys-library-6k-n1.txt:** Most keys consist of 1 or 2 properties, with very few requiring 3 properties. This suggests high minimality.
- **sakey-keys-library-196k-n1.txt:** While the majority of keys have 2 or 3 properties, the presence of keys requiring up to 6 properties indicates a broader distribution and less minimality compared to the 6k dataset.

4.3 Use the keys for data linking

The following **SPARQL** queries have been considered to study the retrieved information from **SAKey**:

- **Query 1**

To check if **dbp:etymology** is a valid *0-almost* key (meaning it can uniquely identify library entities with no exceptions), we can create a **SPARQL** query that retrieves pairs of libraries that share the same **etymology**

value but are distinct entities. If no such pairs exist, it would suggest that **dbp:etymology** is functioning as a valid key. The considered **SPARQL** query is defined as follows:

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT DISTINCT ?library1 ?library2
WHERE {
    ?library1 rdf:type dbo:Library .
    ?library1 dbp:etymology ?etymology1 .

    ?library2 rdf:type dbo:Library .
    ?library2 dbp:etymology ?etymology2 .

    # Ensure the two libraries are distinct
    FILTER (?library1 != ?library2)

    # Ensure they share the same etymology value
    FILTER (?etymology1 = ?etymology2)
}
LIMIT 10
```

The results from the query are in the following figure 10:



Figure 10: Query 1 - Results

• Query 2

For this next **SPARQL** query, we considered the 0 – *almost* keys **location** and **established**, which appear on the results from **SAKey**.

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
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/>
PREFIX dbp: <http://dbpedia.org/property/>

SELECT DISTINCT ?library1 ?library2
WHERE {
    ?library1 rdf:type dbo:Library .
    ?library1 dbp:established ?established1 .
    OPTIONAL { ?library1 dbp:location ?location1 . }
```

```

?library2 rdf:type dbo:Library .
?library2 dbp:established ?established2 .
OPTIONAL { ?library2 dbp:location ?location2 . }

FILTER(
  (?established1 != ?established2) ||
  (BOUND(?location1) && BOUND(?location2) && ?location1 != ?location2)
)
}
LIMIT 10

```

However, the results of the **SPARQL** query in figure 11 indicate that the combination of **location** and **established** is not strictly unique for identifying libraries, as multiple libraries are associated with the same library. This doesn't align with the concept of a 0 – *almost* key in **SAKey**.

Query: <http://dbpedia.org/sparql>

```

1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
4 PREFIX dbo: <http://dbpedia.org/ontology/>
5 PREFIX dbp: <http://dbpedia.org/property/>
6
7 SELECT DISTINCT ?library1 ?library2
8 WHERE {
9   ?library1 rdf:type dbo:Library .
10  ?library1 dbp:established ?established1 .
11  OPTIONAL { ?library1 dbp:location ?location1 . }
12
13  ?library2 rdf:type dbo:Library .
14  ?library2 dbp:established ?established2 .
15  OPTIONAL { ?library2 dbp:location ?location2 . }
16
17  FILTER(
18    (?established1 != ?established2) ||
19    (BOUND(?location1) && BOUND(?location2) && ?location1 != ?location2)
20  )
21 }
22 LIMIT 10
23

```

Showing 1 to 10 of 10 entries (in 0.045 seconds)

library1	library2
http://dbpedia.org/resource/Cabarrus_County_Public_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Cal_Poly_Pomona_University_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Caleis_Free_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Calgary_Tool_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/California_Area_Public_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Cambridge_University_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Camden_Public_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Cantonment_Public_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Capitol_View_Neighborhood_Library	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives
http://dbpedia.org/resource/Carl_Albert_Center	http://dbpedia.org/resource/California_Ethnic_and_Multicultural_Archives

Showing 1 to 10 of 10 entries (in 0.045 seconds)

Figure 11: Query 2 - Results

In particular, the results highlight potential issues in the dataset. Specifically, the repetition of **California Ethnic and Multicultural Archives** suggests either incomplete or inconsistent data for the **location** or **established** properties. This could be due to shared or overlapping values for these attributes, or errors in the data.

Further investigation into the dataset is necessary to understand and possibly address these anomalies, ensuring that the combination of properties achieves a higher degree of uniqueness.

• Query 3

Finally, we consider a third **SPARQL** query, which retrieves pairs of distinct library entities (of type **dbo:Library**) that share the same **location** and **country** values, and once again, in theory such pair of properties is an 0 – *almost* key as shown in figure 12. The **SPARQL** query is the following:

```

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

SELECT DISTINCT ?dbpediaID1 ?dbpediaID2
WHERE {
  # First entity

```

[illegible]

The results obtained are the following:

Figure 13: Query 3 - Results

19

4.4 Going Further: Vickey & Rocker

4.4.1 Vickey

4.4.1.1 OAEI_2011_Restaurant_1.nt

Executing the following **command** produced the following results:

```
java -jar Vickey-20250118/vickey.jar -mins 5 -p datasets-20250118/datasets/OAEI_2011_Restaurant_1.nt > vickey-restaurant_1.nt.txt
```

Results:

- Computing the non-keys: 4 non-keys found.
- Facts loaded: 1130.
- Minimum support: 5.0%.
- Instances found: 339.
- Unique Conditional Keys: 0 found in 1 ms.

4.4.1.2 Library-6k.nt

Executing the following **command** produced the following results:

```
java -jar Vickey-20250118/vickey.jar -mins 5 -p datasets-20250118/library-6k.nt > vickey-keys-library-6k-n1.txt
```

Results:

- Computing the non-keys: 23 non-keys found.
- Facts loaded: 6098.
- Minimum support: 5.0%.
- Instances found: 96.
- Unique Conditional Keys: 0 found in 2 ms.

4.4.1.3 Library-196k.nt

Executing the following **command** produced the following results:

```
java -jar Vickey-20250118/vickey.jar -mins 5 -p datasets-20250118/library-196k.nt > vickey-keys-library-196k.nt.txt
```

Results:

- Computing the non-keys: 522 non-keys found.
- Facts loaded: 196,501.
- Minimum support: 5.0%.
- Instances found: 2687.
- Unique Conditional Keys: 0 found in 5 ms.

4.4.2 Rocker

4.4.2.1 OAEI_2011_Restaurant_1.nt

Executing the following **command** produced the following results:

```
java -Xmx8g -jar Vickey-20250118/rocker-1.2.1-full.jar "OAEI_2011_Restaurant_1"
"file:///Users/pablomollacharlez/Downloads/Key_Discovery/datasets-
20250118/datasets/OAEI_2011_Restaurant_1.nt"
"http://www.okkam.org/ontology_restaurant1.owl#Restaurant" false true 10.0 >
rocker-keys-OAEI_2011_Restaurant_1.txt
```

Results:

- Instances: 113.
- Properties: 4 properties analyzed.
- Parsing Errors: 0.
- Exact Keys:
 - http://www.okkam.org/ontology_restaurant1.owl#has_address (Score: 1.0).
 - http://www.okkam.org/ontology_restaurant1.owl#name (Score: 1.0).
- Composite Key:
 - [http://www.okkam.org/ontology_restaurant1.owl#category,
http://www.okkam.org/ontology_restaurant1.owl#phone_number] (Score: 1.0).
- High-Scoring 0-Almost Key:
 - http://www.okkam.org/ontology_restaurant1.owl#phone_number: 0.991.
- Low-Scoring Property:
 - http://www.okkam.org/ontology_restaurant1.owl#category: 0.168.

4.4.2.2 Library-6k.nt

Executing the following **command** produced the following results:

```
java -Xmx8g -jar Vickey-20250118/rocker-1.2.1-full.jar "library-6k.nt"
"file:///Users/pablomollacharlez/Downloads/Key_Discovery/datasets-20250118/library-6k.nt"
"http://dbpedia.org/ontology/Library" false true 10.0 > rocker-keys-library-6k.txt
```

Results:

- Instances: 95.
- Properties: 221 unique properties analyzed.
- Parsing Errors: 1 (unclosed triple).
- Exact Keys:
 - <http://www.w3.org/ns/prov#wasDerivedFrom>.
 - <http://dbpedia.org/ontology/wikiPageRevisionID>.
 - <http://dbpedia.org/ontology/wikiPageLength>.

- <http://www.w3.org/2000/01/rdf-schema#label>.
- High-Scoring 0-Almost Keys:
 - <http://xmlns.com/foaf/0.1/homepage>: 0.505.
 - <http://dbpedia.org/property/location>: 0.452.
 - <http://dbpedia.org/ontology/wikiPageOutDegree>: 0.421.

4.4.2.3 Library-196k.nt

Executing the following **command** produced the following results:

```
java -Xmx8g -jar Vickey-20250118/rocker-1.2.1-full.jar "library-196k.nt"
"file:///Users/pablomollacharlez/Downloads/Key_Discovery/datasets-20250118/library-196k.nt"
"http://dbpedia.org/ontology/Library" false true 10.0 > rocker-keys-library-196k.txt
```

Results:

- Instances: 2687.
- Properties: 744 unique properties analyzed.
- Parsing Errors: 0.
- High-Scoring 0-Almost Keys:
 - <http://dbpedia.org/property/website>: 0.362.
 - <http://dbpedia.org/property/type>: 0.073.
 - <http://dbpedia.org/property/itemsCollected>: 0.064.
- Many properties had low scores (e.g., 0.001 or lower), reflecting weak potential as 0-almost keys.