Laboratory: Key Discovery

Pablo Mollá Chárlez

February 8, 2025

Contents

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 $K = hasKey(c(OP_1,\ldots,OP_m)(DP_1,\ldots,DP_n))$, its OWL 2 semantics is expressed as follows:

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

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

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

  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 $K1$ to $D1 \times D2$:
  - 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 paris 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 porperties, 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	–	–

S-keys
{FirstName, LastName}
{SSN}
{LastName, StudiedIn}
{FirstName, HasSibling}

SF-keys
{FirstName, LastName}
{SSN}
{StudiedIn}
{HasSibling}

F-keys
{FirstName, LastName}
{LastName,StudiedIn}
...

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 wich 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 $i = 1, \ldots, m$.
- DP_j: A data property (e.g., 'title', 'rDate'), for $j = 1, \ldots, n$.
- 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 $K_1 = \{hasActor, director\}$ (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 itelf:

	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 $y = i_1 2$, we have that both belong to the same class **Film** ($\longleftrightarrow C(x), C(y)$), 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 ($\longleftrightarrow$ **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**.

1 How to Evaluate Rules Using an SQL Interface

The goal of this section is to understand how to translate the computation of the support and confidence metrics of a rule using an SQL interface. We consider a generic rule R of the form:

$$R : B(\alpha, \beta) \implies r(\alpha, \beta)$$

where:

- B(α, β) represents the body of the rule, which can include one or more conditions.
- r(α, β) corresponds to the head relation of the rule.

1.1 Metric Formulas

The two main metrics used to evaluate a rule are defined as follows:

- Support: The support of a rule R measures the number of facts in the knowledge base $\mathcal{K}$ that validate the rule R. Mathematically, it is expressed by the following formula:

$$\text{support(R)} = |\{p : (\mathcal{K} \wedge R \vDash p) \wedge p \in \mathcal{K}\}|$$

- Confidence: The confidence of a rule R represents the proportion of correct predictions among all predictions made by the rule. It is given by:

$$\text{confidence(R)} = \frac{\text{support(R)}}{\text{support(R)} + |\text{cex(R)}|}$$

where:

  - **support(R)** is the number of correct predictions (true positives).
  - **cex(R)** represents the counterexamples, i.e., predictions made by R that are not valid in the knowledge base K.

Note: Confidence metrics vary depending on the adopted hypothesis. Therefore, we define:

  - **owa-conf:** Confidence based on the open-world assumption
  - **cwa-conf:** Confidence based on the closed-world assumption
  - **pca-conf:** Confidence based on the partial completeness assumption

1.2 Exercise: Evaluating a Rule on Wikidata

Objective: Apply the concepts of support and confidence to evaluate a rule based on data from Wikidata. Consider the following rule:

$$R : \text{positionHeld(x, UKPrimeMinister)} \implies \text{memberOf(x, BullingdonClub)}$$

SPARQL Query: UK Prime Ministers Who Were Not Members of the Bullingdon Club

```
SELECT DISTINCT ?primeMinister ?primeMinisterName
WHERE {
?primeMinister p:P39 ?statement.
?statement ps:P39 wd:Q14211. # Position:UK Prime Minister
FILTER NOT EXISTS {
    primeMinister wdt:P463 wd:Q469039. #Not a member of Bullingdon Club
}
OPTIONAL {
    ?primeMinister rdfs:label ?primeMinisterName.
    FILTER (LANG(?primeMinisterName ) = "en"). #Retrieve labels in English
    }
}
```

Testing Platforms: The Wikidata and YAGO platforms provide online SPARQL endpoints:

★ Wikidata SPARQL Endpoint ★ YAGO SPARQL Endpoint

To explore the predicates available on Wikidata, consult the complete list of properties: Wikidata Property List.

1.3 Solution

We have the following rule:

$$R : \text{positionHeld(x, UKPrimeMinister)} \implies \text{memberOf(x, BullingdonClub)}$$

and we want to know how well that rule holds in Wikidata. This rule basically states:

"All UK Prime Ministers are members of the Bullingdon Club."

We want to measure how true that rule actually is in Wikidata, and to do so, we need to apply the concepts of support and confidence. To find the support, we need to SPARQL query the true predictions, and we can use the following query:

```
SELECT DISTINCT ?primeMinister ?primeMinisterName
WHERE {
  # Must be a UK Prime Minister
  ?primeMinister wdt:P39 wd:Q14211 .

  # Must be a member of the Bullingdon Club
  ?primeMinister wdt:P463 wd:Q469039 .

  # Get an English label if available
  OPTIONAL {
    ?primeMinister rdfs:label ?primeMinisterName .
    FILTER(LANG(?primeMinisterName) = "en")
  }
}
```

The previous query produces 2 results, therefore the support is: $Support(R) = 2$.

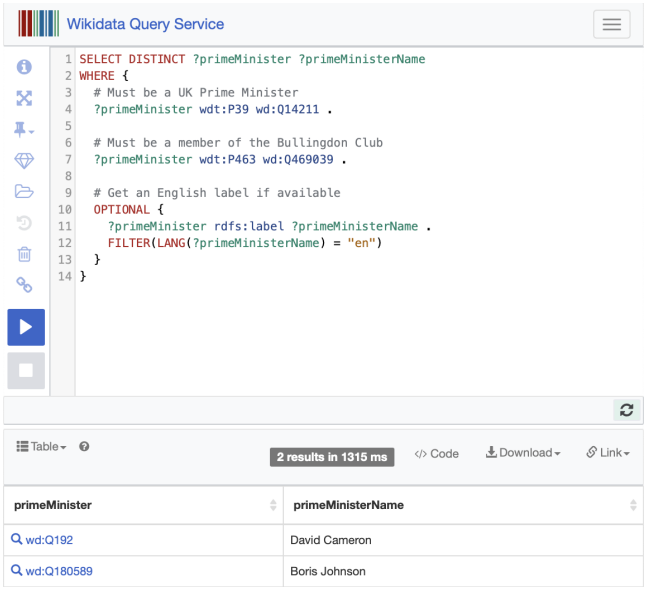

Figure 1: Support Query

Now, we need to compute the confidence of such rule, thus we need to determine the **cex(R)**, which is the **counterexamples** of the rule **R**, meaning the cases where the rule's prediction fails. The confidence is given by the formula:

$$\text{confidence(R)} = \frac{\text{support(R)}}{\text{support(R)+|cex(R)|}}$$

In order to determine the **counterexamples**, we need to decide which prime ministers are considered as **counterexamples**. In other words, "which prime ministers does the rule claim should be members of the Bullingdon Club, but in fact are not members, according to the knowledge base in Wikidata". We need to consider 3 different assumptions:

1. **Closed-World Assumption (CWA)**

   Under CWA, if the person is not stated to be a member, then he definitely is not a member. So every prime minister which is not in our support query is not a member. We can find all "non-members" simply by looking for prime ministers where there is no wdt:P463 (member of) = Q469039 (Bullingdon Club) statement. Then, the count of these results will be the **counterexamples cex(R)** under CWA.

```
SELECT DISTINCT ?primeMinister ?primeMinisterName
WHERE {
  # Must be a UK Prime Minister
  ?primeMinister wdt:P39 wd:Q14211 .

  # FILTER NOT EXISTS => "No membership in Bullingdon"
  FILTER NOT EXISTS {
    ?primeMinister wdt:P463 wd:Q469039 .
  }

  # Get an English label if available
  OPTIONAL {
    ?primeMinister rdfs:label ?primeMinisterName .
    FILTER(LANG(?primeMinisterName) = "en")
  }
}
```

   The previous SPARQL query produces 79 results, meaning that there are 79 prime ministers who are not members of the Bullingdon Club. Then,

$$\text{confidence}_{\text{CWA}}(R) = \frac{\text{support}(R)}{\text{support}(R) + \text{cex}(R)} = \frac{2}{2+79} \approx 0.0246.$$

2. **Open-World Assumption (OWA)**

   Under OWA, if a prime minister is not stated to be a member, that does not automatically mean "not a member" - it might just be unknown. So we only count as a counterexample if the data explicitly says the primer minister is not a member. In Wikidata, we do not usually have "negative" statements such as "X is definitely not in the Bullingdon Club." So in practice, your cex(R) under OWA may well be zero, unless Wikidata has some specific "excluded" membership property.

```
SELECT DISTINCT ?primeMinister ?primeMinisterName
WHERE {
  ?primeMinister wdt:P39 wd:Q14211 .   # is a UK Prime Minister

  # -- Hypothetical: Some kind of explicit "excluded membership" statement
  # e.g. a specialized property or a "no value" statement for wdt:P463 = Q469039

  ?primeMinister p:P463 ?membershipStatement .
  ?membershipStatement ps:P463 wd:Q469039 .
  ?membershipStatement wikibase:rank  wd:deprecatedRank .

  OPTIONAL {
    ?primeMinister rdfs:label ?primeMinisterName .
    FILTER(LANG(?primeMinisterName) = "en")
  }
}
```

That query would return all prime ministers that are explicitly flagged as "not members". In most Wikidata contexts, though, the database does not store such explicit negatives. You'll probably get zero results, meaning **cex(R)** under OWA = 0. Then:

$$\text{confidence}_{\text{OWA}}(R) = \frac{\text{support}(R)}{\text{support}(R) + \text{cex}(R)} = \frac{\text{support}(R)}{\text{support}(R) + 0} = 1.0 \quad \text{(if no explicit negation is found)}$$

3. **Partial-Completeness Assumption (PCA)**

   Under PCA, we only consider an entity as "negative" if Wikidata does list some membership(s) for that entity, but not the Bullingdon Club. If Wikidata is completely silent about all clubs for that prime minister, we skip them (we say "unknown," not negative). Thus we need two pieces:

   (a) Who are the prime ministers that have some wdt:P463 membership statement?

   (b) Among those, which do not mention **wd:Q469039**?

   Here's a query that returns prime ministers who definitely have membership data for at least one organization, but not Q469039:

```
SELECT DISTINCT ?primeMinister ?primeMinisterName
WHERE {
  # must be a UK Prime Minister
  ?primeMinister wdt:P39 wd:Q14211 .

  # has at least one membership in some org
  ?primeMinister wdt:P463 ?someOrg .

  # but that org is not Bullingdon
  FILTER(?someOrg != wd:Q469039)

  # Now, we also want to ensure that the PM is never stated
  # to be in Bullingdon. So we do a NOT EXISTS check:
  FILTER NOT EXISTS {
    ?primeMinister wdt:P463 wd:Q469039 .
  }

  OPTIONAL {
    ?primeMinister rdfs:label ?primeMinisterName .
    FILTER(LANG(?primeMinisterName) = "en")
  }
}
```

The count of these results corresponds to the **cex(R)** under PCA. Using Wikidata, we obtain 39 prime ministers satisfying the previous conditions. They are prime ministers for whom we know at least one membership (so presumably the membership data is "partially complete"), yet we see no membership in Bullingdon. Then:

$$\text{confidence}_{\text{PCA}}(R) = \frac{\text{support}(R)}{\text{support}(R) + \text{cex}(R)} = \frac{2}{2 + 39} = 0.0488.$$

Finally, note that any prime ministers who has no membership statements at all for **wdt:P463** simply does not appear in that PCA counterexample list. That is exactly how we avoid penalizing "unknown membership" under PCA.

2 Computing Metrics on a Given Graph

Consider the relations created and createdBy, along with the following knowledge graph: The thick red line

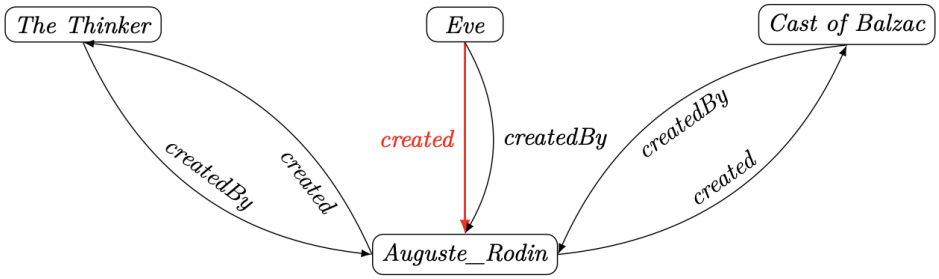

Figure 2: Relations **created** and **createdBy**

represents an erroneous fact, extracted incorrectly instead of its inverse. The domains of the two relations should normally be disjoint, as these relations are inverses of each other and are not symmetric. However, due to this error, the domains of **createdBy** and **created** share the element **Eve** in common.

1. Compute:
   - **pca-conf(createdBy(x, y) $\Longrightarrow$ created(x, y)) =?**
   - **pca-conf(created(x, y) $\Longrightarrow$ createdBy(x, y)) =?**
2. What do you observe?

   A concise way to see what happens is to list exactly which "created" and "createdBy" facts appear in the little graph (including the one erroneous triple). Then we can count support and counterexamples under the **partial-completeness assumption**.

   - **The Facts in the Graph**

     From the figure, we have the following triples (the thick red one is erroneous but does appear in the Knowledge Base):

     $$TheThinker \Longrightarrow createdBy \Longrightarrow AugusteRodin$$
     $$CastofBalzac \Longrightarrow createdBy \Longrightarrow AugusteRodin$$
     $$Eve \Longrightarrow createdBy \Longrightarrow AugusteRodin$$
     $$AugusteRodin \Longrightarrow created \Longrightarrow TheThinker$$
     $$AugusteRodin \Longrightarrow created \Longrightarrow CastofBalzac$$
     $$Eve \Longrightarrow created \Longrightarrow AugusteRodin \text{ (erroneous but present)}$$

     Hence the domain of "createdBy" includes three works ("The Thinker," "Cast of Balzac," and "Eve"), and the domain of "created" (artist $\Longrightarrow$ artwork) should have been just Auguste Rodin, but because of the error, "Eve" also appears as a creator. Next, we will study once again the 3 assumptions as practice:

     – $\boxed{R_1 : \textbf{createdBy(x, y)} \Longrightarrow \textbf{created(x, y)}}$

       (a) Open-World Assumption (OWA):

       * **Support**: The support of the rule corresponds to the number of (x, y) for which both createdBy(x,y) and created(x,y) appear in the Knowledge Base, which is 1, because we have for **x** = Eve, **y** = Auguste Rodin:
         · createdBy(Eve, Auguste_Rodin) and
         · created(Eve, Auguste_Rodin) (the erroneous triple for us but not for the system).

* **Counterexample**: If we call $B \vDash$ *createdBy* (body) and $H \vDash$ *created* (head), the counterexample corresponds to: cex_OWA = $|B \cap \neg H|$, then it's easy to deduce that there are no relations stating **not created**, therefore cex_OWA = $|B \cap \varnothing| = |\varnothing| = 0$.
* **Confidence:** Confidence_OWA = $\frac{\text{support}(R_1)}{\text{support}(R_1)+\text{cex}_{OWA}(R_1)} = \frac{1}{1+0} = 1.0$

(b) Closed-World Assumption (CWA):

* **Support**: The support of the rule still is 1.
* **Counterexample**: If we call $B \vDash$ *createdBy* (body) and $H \vDash$ *created* (head), the counterexample corresponds to: cex_CWA = $|\{(x, y) \in B : (x, y) \notin H\}|$, then it's easy to deduce that there are 2 relations satisfying it, therefore cex_CWA = 2.
* **Confidence:** Confidence_CWA = $\frac{\text{support}(R_1)}{\text{support}(R_1)+\text{cex}_{CWA}(R_1)} = \frac{1}{1+2} = 0.33$

(c) Partial-Completeness Assumption (PCA):

* **Support**: The support of the rule still is 1.
* **Counterexample**: If we call $B \vDash$ *createdBy* (body) and $H \vDash$ *created* (head), the counterexample corresponds to: cex_PCA = $|\{(x, y) \in B : x \in H \wedge (x, y) \notin H\}|$, then it's easy to deduce that there is no relation satisfying it (from the system point of view, the erroneous triple is not wrong even if we know it is), therefore cex_CWA = 0.
* **Confidence:** Confidence_CWA = $\frac{\text{support}(R_1)}{\text{support}(R_1)+\text{cex}_{CWA}(R_1)} = \frac{1}{1+0} = 1.0$

– $\boxed{R_2 : \textbf{created(x, y)} \implies \textbf{createdBy(x, y)}}$

(a) Open-World Assumption (OWA):

* **Support**: The support of the rule corresponds to the number of (x, y) for which both created(x,y) and createdBy(x,y) appear in the Knowledge Base, which is 1, because we have for **x** = Eve, **y** = Auguste Rodin:
  · created(Eve, Auguste_Rodin) (the erroneous triple for us but not for the system) and
  · createdBy(Eve, Auguste_Rodin)
* **Counterexample**: If we call $B \vDash$ *created* (body) and $H \vDash$ *createdBy* (head), the counterexample corresponds to: cex_OWA = $|B \cap \neg H|$, then it's easy to deduce that there are no relations stating **not createdBy**, therefore cex_OWA = $|B \cap \varnothing| = |\varnothing| = 0$.
* **Confidence:** Confidence_OWA = $\frac{\text{support}(R_1)}{\text{support}(R_1)+\text{cex}_{OWA}(R_1)} = \frac{1}{1+0} = 1.0$

(b) Closed-World Assumption (CWA):

* **Support**: The support of the rule still is 1.
* **Counterexample**: If we call $B \vDash$ *created* (body) and $H \vDash$ *createdBy* (head), the counterexample corresponds to: cex_CWA = $|\{(x, y) \in B : (x, y) \notin H\}|$, then it's easy to deduce that there are 2 relations satisfying it, therefore cex_CWA = 2.
* **Confidence:** Confidence_CWA = $\frac{\text{support}(R_1)}{\text{support}(R_1)+\text{cex}_{CWA}(R_1)} = \frac{1}{1+2} = 0.33$

(c) Partial-Completeness Assumption (PCA):

* **Support**: The support of the rule still is 1.
* **Counterexample**: If we call $B \vDash$ *created* (body) and $H \vDash$ *createdBy* (head), the counterexample corresponds to: cex_PCA = $|\{(x, y) \in B : x \in H \wedge (x, y) \notin H\}|$, then it's easy to deduce that there is no relation satisfying it (from the system point of view, the erroneous triple is not wrong even if we know it is), therefore cex_CWA = 0.
* **Confidence:** Confidence_CWA = $\frac{\text{support}(R_1)}{\text{support}(R_1)+\text{cex}_{CWA}(R_1)} = \frac{1}{1+0} = 1.0$

In this toy example, both inverse rules end up having exactly the same confidence under all three assumptions:

- OWA and PCA each yield a confidence of 1.0 for both rules, because only the single (erroneous) pair (Eve, Rodin) is in both relations, and there are no "explicit negatives" or "partially complete subjects" that force counterexamples.

- CWA yields a confidence of 1/3 for both rules, since each rule has the same number of "body-only" pairs (2) and one "body+head" pair (1), giving $1/(1+2) = 1/3$.

So even though the two rules are truly inverse (and one is obviously incorrect), they end up with the same confidence values under each assumption given the particular facts and error in this KB.

3 Detection of Subproperty Relationships

In the context of description logic, the concept of a subproperty is essential for modeling hierarchical inclusion between properties. If a property r_1 is a subproperty of r_2 (denoted $r_1 \subseteq r_2$), it means that all instances of r_1 are also instances of r_2. In other words, the subproperty relationship expresses that r_1 is a specialization of r_2, thereby formalizing an inheritance relationship between the two properties. The subproperty relationship, denoted $r_1 \subseteq r_2$, can be expressed as a Horn clause as follows:

$$r_1(x, y) \implies r_2(x, y)$$

This rule states that if a pair (x, y) satisfies the relation $r_1(x, y)$, then it must also satisfy the relation $r_2(x, y)$. In other words, every instance of the property r_1 implies a corresponding instance of the property r_2, thus modeling the hierarchical inclusion between the two properties. Consider the following property:

Property: Let r_1 and r_2 be two functional relations with $r_1 \subseteq r_2$. Then,

$$\text{pca-conf}(r_1 \subseteq r_2) = 1 \text{ and } \text{pca-conf}(r_2 \subseteq r_1) = 1.$$

4 Query Expressiveness

4.1 Objective: Comparing SPARQL and SQL

The goal of this section is to analyze the SQL equivalents of the **OPTIONAL** clause, **FILTER NOT EXIST** clause, and path patterns found in SPARQL queries. We assume that the knowledge base is stored in a single table, **KB**, with three attributes:

S (subject), **P (predicate)**, and **O (object)**.

This schema, which may seem inefficient, has proven to be highly performant. In particular, it is the schema adopted by the RDF3X system, which has demonstrated superiority compared to a partitioned schema where each predicate is stored in a separate table **P(S, O)—P for predicate**, **S for subject**, and **O for object**.

4.2 Exercises

Translate the following SPARQL queries into SQL equivalents for the **OPTIONAL** clause and path queries.

1. **Query 1:**

```
SELECT DISTINCT ?scientist
WHERE {
    ?scientist rdf:type yago:Scientist.
    FILTER NOT EXISTS {
        ?scientist schema:award yago:NobelPrize.
    }
}
```

SQL Version of Query 1:

```sql
SELECT DISTINCT t1.subject AS scientist
FROM Triples AS t1
WHERE t1.predicate = 'rdf:type'
  AND t1.object = 'yago:Scientist'
  AND NOT EXISTS (
    SELECT 1
    FROM Triples AS t2
    WHERE t2.subject = t1.subject
      AND t2.predicate = 'schema:award'
      AND t2.object = 'yago:NobelPrize'
  );
```

2. **Query 2:**

```
SELECT ?pName ?date
WHERE {
    ?p schema:award yago:NobelPrize.
    ?p rdfs:label ?pName .
    OPTIONAL { ?p schema:birthDate ?date.}}
```

SQL Version of Query 2:

```sql
SELECT DISTINCT lbl.object AS pName, bdate.object AS date
FROM Triples AS aw
JOIN Triples AS lbl
  ON lbl.subject = aw.subject
  AND lbl.predicate = 'rdfs:label'
LEFT JOIN Triples AS bdate
  ON bdate.subject = aw.subject
  AND bdate.predicate = 'schema:birthDate'
WHERE aw.predicate = 'schema:award'
AND aw.object = 'yago:NobelPrize';
```

3. **Query 3:**

```
SELECT ?Nname
WHERE {
    ?p rdfs:label ?pName.
    ?p rdf:type yago:Scientist.}
```

SQL Version of Query 3:

```sql
SELECT DISTINCT label.object AS Nname
FROM Triples AS t
JOIN Triples AS label
  ON t.subject = label.subject
  AND label.predicate = 'rdfs:label'
WHERE t.predicate = 'rdf:type'
  AND t.object    = 'yago:Scientist';
```

4. **Query 4:**

```
SELECT ?pName
WHERE {
    ?p rdfs:label ?pName.
    ?p rdf:type/rdfs:subClassOf yago:Scientist.}
```

SQL Version of Query 4:

```sql
SELECT DISTINCT lbl.object AS pName
FROM Triples AS type
JOIN Triples AS lbl
   ON lbl.subject = type.subject
   AND lbl.predicate = 'rdfs:label'
JOIN Triples AS sc
   ON sc.subject = type.object
   AND sc.predicate = 'rdfs:subClassOf'
   AND sc.object    = 'yago:Scientist'
WHERE type.predicate = 'rdf:type';
```

4.3 Testing the Solutions

We suggest using the online SQL database tool: https://sqliteonline.com/.

Creating and Dropping the Table: Before inserting data, we need to create the table. To avoid duplication errors, drop the table if it already exists using:

- **Dropping the Table:**

```
DROP TABLE IF EXISTS KB;
```

- **Creating the KB Table:**

```
CREATE TABLE KB (
S TEXT,
P TEXT,
O TEXT):
```

- **Inserting Data:** Insert the following data into the table.

```
INSERT INTO KB (S, P, O) VALUES
('yago:Marie_Curie', 'rdfs:label', '"Marie Curie"@en'),
('yago:Max_Planck', 'rdfs:label', '"Max Planck"@en'),
('yago:Marie_Curie', 'rdf:type', 'yago:Scientist'),
('yago:Dmitri_Mendeleev', 'rdf:type', 'yago:Scientist'),
('yago:Max_Planck', 'rdf:type', 'yago:TheoreticalPhysicist'),
('yago:TheoreticalPhysicist', 'rdfs:subClassOf', 'yago:Physicist'),
('yago:Physicist', 'rdfs:subClassOf', 'yago:Scientist'),
('yago:Marie_Curie', 'schema:birthDate', '"1867-11-07"^^xsd:date'),
('yago:Marie_Curie', 'rdfs:label', '"Marie Curie"@fr'),
('yago:Max_Planck', 'rdfs:label', '"Max Planck"@de'),
('yago:Marie_Curie', 'schema:award', 'yago:NobelPrize'),
('yago:Marie_Curie', 'schema:award', 'yago:AlbertMedal'),
('yago:Marie_Curie', 'schema:award', 'yago:NobelPrize'),
('yago:Max_Planck', 'schema:award', 'yago:NobelPrize');
```

These rows represent **RDF** facts expressed as triples (subject, predicate, object).

- **Verifying the Data:** After insertion, you can verify the table's content using:

```
SELECT  FROM KB;
```

Laboratory: Constraint Rules

Pablo Mollá Chárlez

January 31, 2025

1 SHACL Validation of an RDF Graph

SHACL Validation Processor: We will use the open-access processor available at the following address: https://shacl.org/playground/.

New Dataset: Consider the graph given in the file rdf-graph.ttl as an example. The **dataset** contains the following information:

```
@prefix ex: <http://example.org/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# Define the individual Paul as an instance of schema:Student
ex:Paul a schema:Student ;
    ex:hasSSN "123-45-6789"^^xsd:string ;
    schema:email "paul@example.org"^^xsd:string ;
    ex:doctoralAdviser ex:Pierre ;
    ex:doctoralAdviser ex:Anne ;
    ex:phdCandidateIn ex:CS ;
    schema:affiliation ex:UPS .

# Define the supervisor Pierre
ex:Pierre a schema:Person ;
    ex:phdIn ex:CS ;
    schema:affiliation ex:IPP .

# Define the supervisor Anne
ex:Anne a schema:Person ;
    ex:phdIn ex:Mathematics ;
    schema:affiliation ex:UPS .

# Define the university UPS
ex:UPS a ex:ResearchOrganization ;
    schema:name "Université Paris-Saclay"@fr ;
    schema:locatedin ex:France.

# Define the organization IPP
ex:IPP a schema:ResearchOrganization ;
    schema:name "Institut Polytechnique de Paris"@fr ;
     schema:locatedin ex:France.
```

Propose SHACL shapes (constraints) to express the following constraints. Verify the validity of the solutions by extending the graph. In the report, it is important to include these extensions in addition to the SHACL constraints.

What Are We Doing?

- We have an RDF file (rdf-graph.ttl) describing people (students and supervisors), their universities, and affiliations. The goal is to express a rule (constraint) that every object, students, supervisors, universities or affiliations must satisfy. We use the SHACL (Shapes Constraint Language) to define these rules in a structured way, so that any RDF data must respect these constraints to be considered valid. By uploading both our RDF file and our SHACL shapes file to the SHACL Playground, we can see if our data is valid or if something is missing.

Let's solve each one of the following constraints:

1. **Exercise** Define a constraint indicating that each student must have an email and a Social Security Number (SSN).

   Currently, Paul is our only **schema:Student** and he already has both **schema:email** and **ex:hasSSN**. He should pass the new SHACL constraint that we are going to define as follows:

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

# This line declares ex:StudentShape as a SHACL NodeShape, i.e.
# it defines the set of constraints applied to schema:Student
ex:StudentShape a sh:NodeShape ;
    sh:targetClass schema:Student ;          # 1. This rule applies to all schema:Student
    sh:property [
        sh:path schema:email ;               # 2. Must have an email
        sh:datatype xsd:string ;             #    which is of type xsd:string
        sh:minCount 1 ;                      #    at least 1 occurrence
    ] ;
    sh:property [
        sh:path ex:hasSSN ;                  # 3. Must have an SSN
        sh:datatype xsd:string ;             #    which is of type xsd:string
        sh:minCount 1 ;                      #    at least 1 occurrence
    ] .
```

   In order to verify whether the rule is appropiately working, we can use the SHACL Playground. As it can be observed, no violations of the rule appear as outcome.

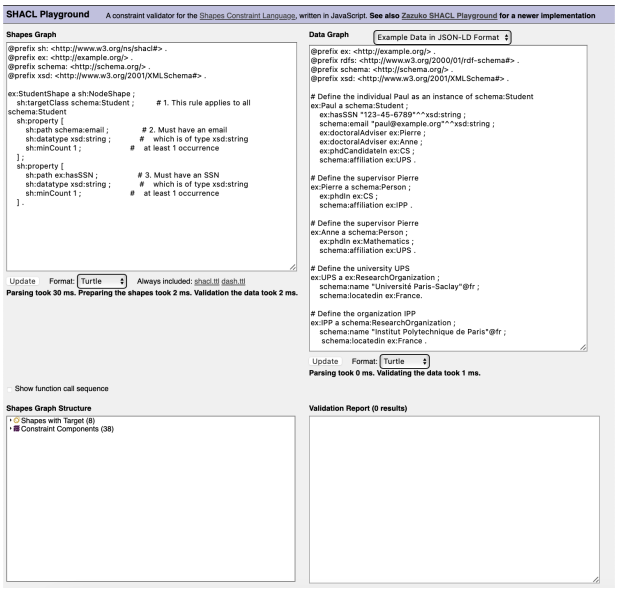

Figure 1: Check Query 1

However, if we introduce, for instance a new student in our RDF file, we would obtain the following error:

```
ex:John a schema:Student ;
    schema:affiliation ex:UPS .
```

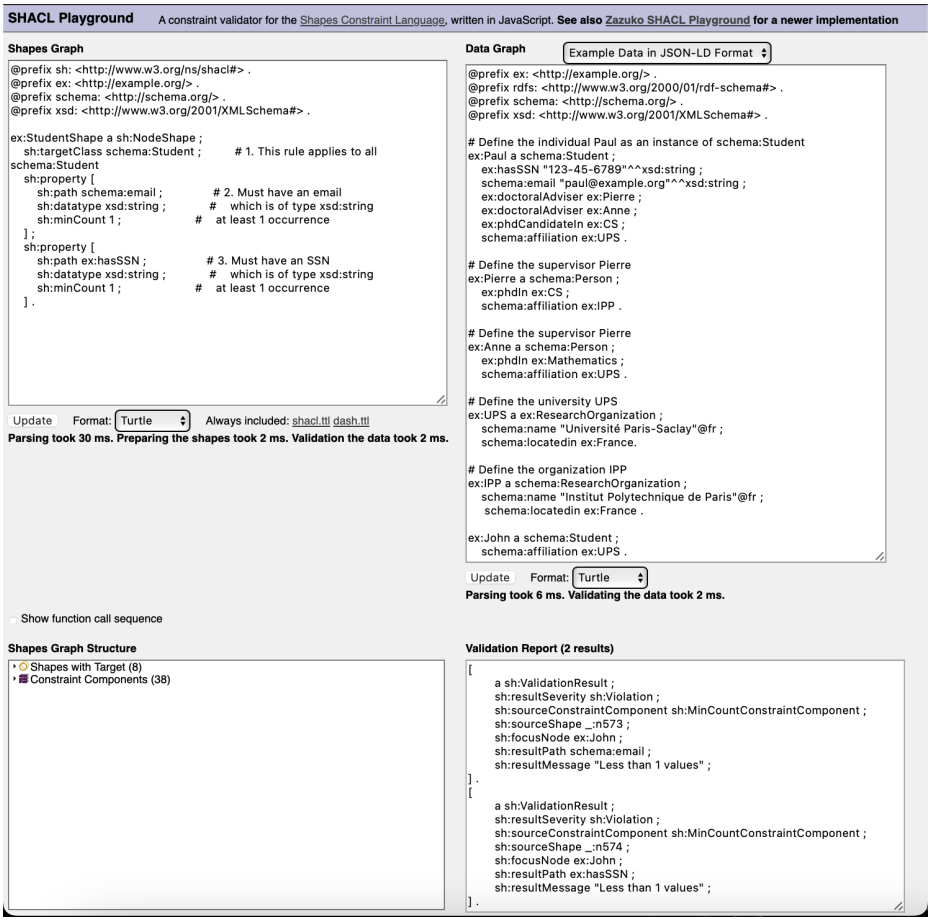

Figure 2: Error Query 1

2. **Exercise** Ensure that the **SSN is a string** and that the **email is a string**.

```
ex:StudentShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:hasSSN ;
    sh:datatype xsd:string
  ] ;
  sh:property [
    sh:path schema:email ;
    sh:datatype xsd:string
  ] .
```

3. **Exercise** Add a constraint to enforce that each student must have **exactly one SSN**.

```
ex:ExactlyOneSSNShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:hasSSN ;
    sh:minCount 1 ;
    sh:maxCount 1
  ] .
```

4. **Exercise** Add a constraint to specify that a **student may have at most one email**.

```
ex:MaxOneEmailShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path schema:email ;
    sh:maxCount 1
  ] .
```

5. **Exercise** Specify a constraint to enforce that the **email must follow a valid format** (e.g., name@example.com).

```
ex:ValidEmailShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path schema:email ;
    sh:pattern "^[^@]+@[^@]+\\.[^@]+$" ;
    sh:message "Email must match a name@domain pattern."
  ] .
```

6. **Exercise** Add a constraint ensuring that the **SSN** is unique—two students cannot have the same **SSN**.

```
ex:UniqueSSNShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "SSN must be unique among all students.";
    sh:select """
      SELECT ?this
      WHERE {
        ?this ex:hasSSN ?ssn .
        ?other ex:hasSSN ?ssn ;
               a schema:Student .
        FILTER (?other != ?this)
      }
    """
  ] .
```

Another solution for this constraint would be:

```
ex:InverseSSNConstraintShape a sh:NodeShape ;
  sh:targetSubjectsOf ex:hasSSN ;
  sh:sparql [
    a sh:SPARQLConstraint ;
    sh:message "The SSN must be unique." ;
    sh:select """
      SELECT $this
      WHERE {
        $this ex:hasSSN ?ssn .
        ?other ex:hasSSN ?ssn .
        FILTER ($this != ?other)
      }
    """ ;
  ] .
```

They both enforce a "unique SSN" rule with SPARQL, but the targets differ:

- **ex:InverseSSNConstraintShape**: applies to any subject that has **ex:hasSSN**.
- **ex:UniqueSSNShape**: applies only to instances of **schema:Student** and checks uniqueness among other **schema:Students**.

So they are not exactly the same constraint.

7. **Exercise** Add a constraint to enforce that a student must have a supervisor and be affiliated with a **schema:ResearchOrganization**.

```
ex:StudentMustHaveSupervisorShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:doctoralAdviser ;
    sh:minCount 1
  ] ;
  sh:property [
    sh:path schema:affiliation ;
    sh:class schema:ResearchOrganization ;
    sh:minCount 1
  ] .
```

In this exercise, when we define a student like Michel with no affiliation (**schema:ResearchOrganization**) nor doctoral adviser (**ex:doctoralAdviser**), we obtain:

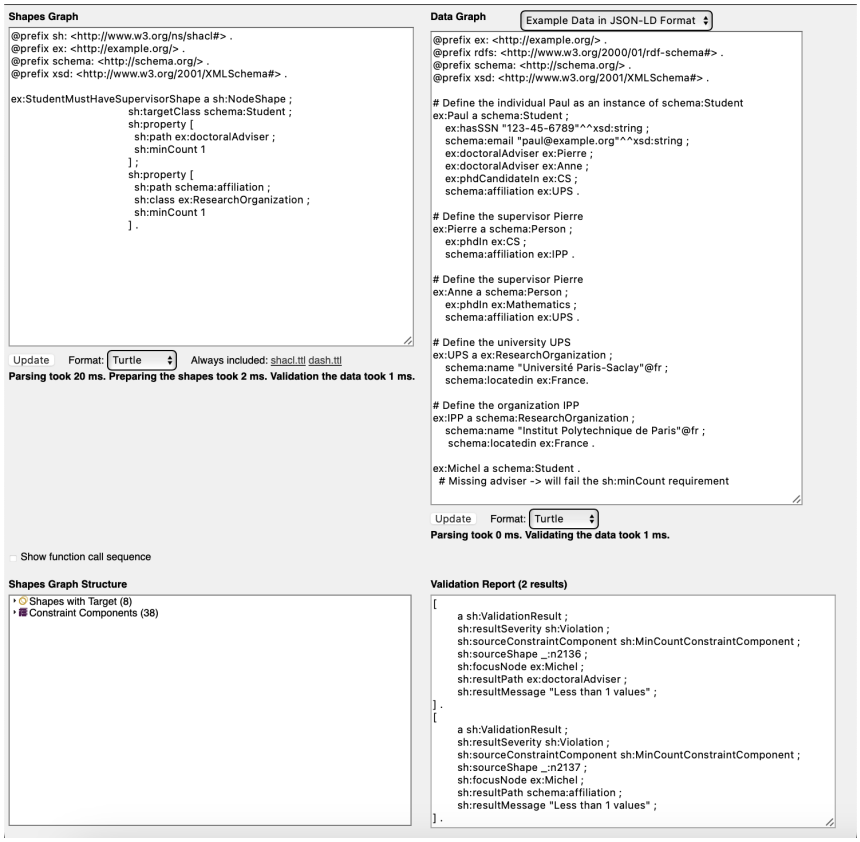

Figure 3: Check Query 7

8. **Exercise** Ensure that a student's supervisor is a person affiliated with a research unit **schema:ResearchOrganizati**

```
ex:SupervisorIsPersonAffiliatedShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:property [
    sh:path ex:doctoralAdviser ;
    sh:class schema:Person ;
    sh:node [
      sh:property [
        sh:path schema:affiliation ;
        sh:class schema:ResearchOrganization
      ]
    ]
  ] .
```

In this query, as we are forcing to check that the supervisor of the student is a person affiliated with a research unit **schema:ResearchOrganization** (as IPP) and not **ex:ResearchOrganization** (as UPS), the SHACL processor finds a violation with one of the supervisor's affiliation.

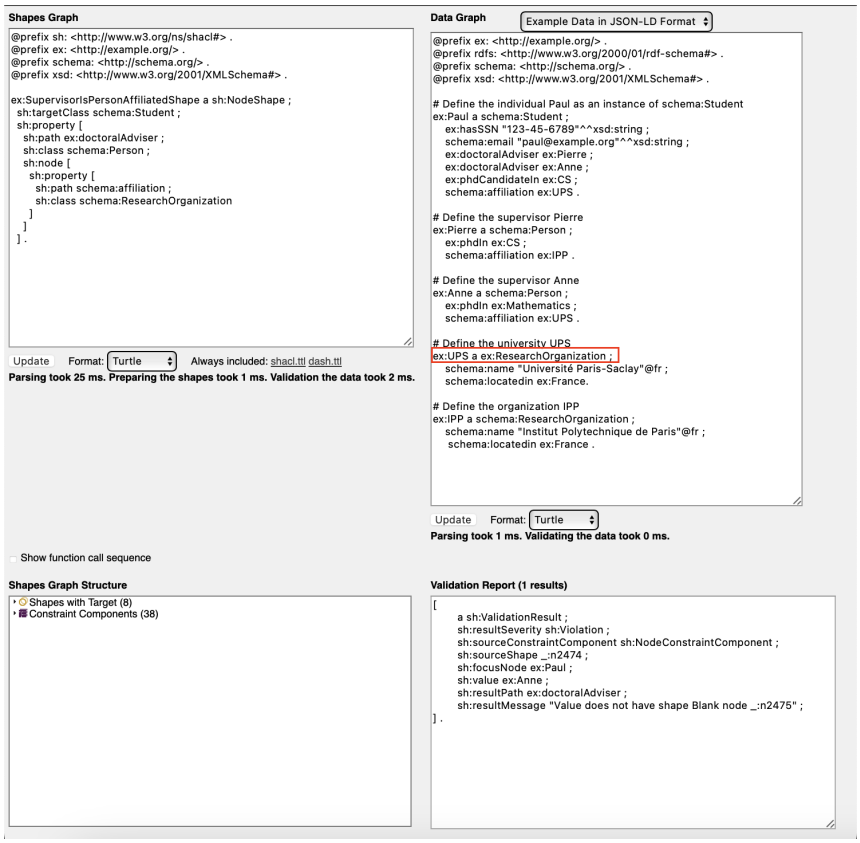

Figure 4: **schema:ResearchOrganization** vs. **ex:ResearchOrganization**

9. **Exercise** Define a constraint to enforce that an **organization must have a name** (rdfs:label).

```
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix ex: <http://example.org/> .
@prefix schema: <http://schema.org/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .


ex:OrganizationShape a sh:NodeShape ;
  sh:targetClass schema:ResearchOrganization ;
  sh:property [
    sh:path rdfs:label ;
    sh:minCount 1
  ] .
```

In this query, make sure to bind **rdfs** prefix to corresponding link. Besides, notice that this rule will produce a violation as in the original data we don't have such property **rdfs:label**, instead we have **schema:name**.

10. **Exercise** Ensure that an organization has only **one denomination per language**.

```
ex:OrganizationLabelLangShape a sh:NodeShape ;
  sh:targetClass schema:ResearchOrganization ;
  sh:property [
    sh:path rdfs:label ;
    sh:uniqueLang true
  ] .
```

To visualize whether the rule works or not, we need to modify the original data, in particular, only the **IPP** object to ease what we want to show:

```
# Defining new version of the organization IPP
# Make sure ex:IPP is a schema:ResearchOrganization
ex:IPP a schema:ResearchOrganization ;
    rdfs:label "Institut Polytechnique de Paris"@fr ;
    # Same language -> will cause a violation
    rdfs:label "Institut Polytechnique de Paris (duplicate)"@fr ;
    schema:locatedin ex:France .
```

Another option to see the violation would be to just replace in the rule **rdfs:label** by **schema:name** and duplicate the name of any of the 2 institutions (use a different name). Something to notice is that by duplicating the name, **schema:name "Institut Polytechnique de Paris"@fr ;** and only changing the language (fr → en) still doesn't produce any violation.

11. **Exercise** All PhD advisors (see the predicate **ex:doctoralAdviser**) must hold a doctorate.

```
ex:PhDAdvisorMustHoldDoctorateShape a sh:NodeShape ;
  sh:targetObjectsOf ex:doctoralAdviser ;
  sh:property [
    sh:path ex:phdIn ;
    sh:minCount 1
  ] .
```

12. **Exercise All supervisors** of a PhD student **must be affiliated with the same university** as the student.

```
ex:SameAffiliationShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "All advisers must have the same affiliation as the student." ;
    sh:select """
      SELECT ?this
      WHERE {
        ?this schema:affiliation ?uni ;
              ex:doctoralAdviser ?adv .
        ?adv schema:affiliation ?advUni .
        FILTER(?advUni != ?uni)
      }
    """
  ] .
```

13. **Exercise At least one of the supervisors** of a PhD student must be **affiliated with the same university** as the student.

```
ex:AtLeastOneSameUniShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "At least one supervisor must share the same affiliation as the student." ;
    sh:select """
      SELECT ?this
      WHERE {
        ?this schema:affiliation ?uni .
        FILTER NOT EXISTS {
          ?this ex:doctoralAdviser ?adv .
          ?adv schema:affiliation ?uni .
        }
      }
    """
  ] .
```

14. **Exercise** PhD students enrolled in a German university cannot have more than one supervisor.

```
ex:GermanUniSupervisorLimitShape a sh:NodeShape ;
  sh:targetClass schema:Student ;
  sh:sparql [
    sh:message "PhD students in a German university cannot have more than one supervisor." ;
    sh:select """
      SELECT ?this
      WHERE {
        ?this schema:affiliation ?org ;
              ex:doctoralAdviser ?adv .
        ?org schema:locatedin ex:Germany .
      }
      GROUP BY ?this
      HAVING (COUNT(?adv) > 1)
    """
  ] .
```

1 Part 1: Key Discovery for Data Linking [12 pts]

1.1 Question 1 [2.5 pts]

Give three different quality measures that are used to evaluate the quality of discovered keys that are proposed by [Soru et al. 15] and [Atencia et al. 12]. For each measure, give an informal definition.

1.1.1 Answer

The three measures commonly used to assess discovered keys are support, discriminability, and score. In the context of **F-Keys** as proposed by Soru et al. (2015), and **SF-Keys** (or Preside-Keys) as discussed by Atencia et al. (2012), these measures can be described informally as follows:

- Support: This measure tells you how widely the key applies by computing the fraction of instances in the dataset that actually provide values for the key's properties. In other words, support gauges the "coverage" of the key across the data, indicating its overall applicability. Formally, the support is defined as:

$$\text{support(P)} = \frac{\text{\# instances described by P}}{\text{\# all instances}}$$

- Discriminability: Discriminability looks at how effectively the key distinguishes individual entities. It does this by partitioning the dataset according to the key's values and then checking what proportion of these groups contain exactly one instance. A discriminability of 1 means every group is a singleton—i.e., the key perfectly differentiates every record. Formally, the discriminability is defined as:

$$\text{dis(P)} = \frac{\text{\# singleton partitions}}{\text{\# partitions}}$$

- Score: Often derived as the ratio of discriminability to the total number of instances, the score provides a normalized measure of the key's quality. A score of 1 indicates a perfect key (all instances are uniquely identified), while a score below 1 shows that some duplicates exist, meaning the key is only a pseudo-key. This measure is especially useful for comparing keys across datasets. Formally, the score is defined as:

$$\text{score(P)} = \frac{\text{discriminality(P)}}{\text{\# instances}}$$

  If $score(P) = 1$, then P is a perfect key, and if $score(P) < 1$, then P is only a pseudo-key.

These quality measures together help evaluate both how broadly a key can be applied (support) and how precisely it differentiates between entities (discriminability and score).

1.2 Question 2 [2.5 pts]

To distinguish between the three key semantics S-Keys, SF**-Keys** and F-Keys that are studied in the course,

- (a) What are the main data characteristics that should be taken into account ?
- (b) How these characteristics are considered in the S-Keys, SF**-Keys** and F-Keys semantics?

1.2.1 Answer

- (a) The main data characteristics to consider are how multi-valued properties and missing values (or incomplete descriptions) are treated. In other words, one must decide whether to compare the values for a property by looking at overlaps between sets or by requiring an exact match, and whether an empty (or missing) value should be regarded as distinct from any non-empty value or possibly identical to other empty values.

- (b) In the S-Keys semantics, multi-valued properties are compared in a relaxed way—that is, two sets are considered equal if they share at least one common element—and empty values are treated as different from any existing value (an optimistic stance). In contrast, F-Keys require an exact match between the sets of values, so two instances' multi-valued properties must be identical for them to be deemed equal; moreover, empty values are treated pessimistically, meaning they might be considered identical. Finally, SF-**Keys** also require an exact match for multi-valued properties like F-Keys but adopt an optimistic approach to empty values by treating them as different from non-empty values. This combination leads to a subtle yet important difference in how keys are discovered and verified.

1.3 Question 3 [7 pts]

In **table 1** we give an extract of some book descriptions. These books are described by six properties {**title**, **hasAuthor**, **genre**, **pages**, **publisher**, **lang**}. Given these data if we apply SAKey, a key discovery tool that allows to discover n-almost keys (under the S-key semantics):

- (a) Give a **2-almost** key of one property that can be discovered. [**1.5pts**]
- (b) Give a **3-almost** key composed of two properties. [**1.5pts**]
- (c) Give a S-Key, composed of two properties, that is not an F-Key that can be discovered in the data presented in **Table 1**. [1pts]

	title	author	genre	pages	publisher	lang
b_1	The Age of Wrath	E. Abraham, Oram Andy	history	198	Penguin	en
b_2	The Trial	Kafka Frank, J. Clarck	fiction	198	R. House	
b_3	Statistical Decision Theory	Pratt John, Tao Terence	data_science	236	MIT Press	en, de
b_4	Data Mining Handbook		data_science	242	Apress	
b_5	The New Machiavelli	Wells H. G.	fiction	198	Penguin	en
b_6	Analysis & Vol I	Tao Terence, N. Robert	science	250	Apress	en
b_7	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_8	Making Software		computer_science	232	O'Reilly	
b_9	Analysis & Vol I	Tao Terence	mathematics	248	HB	en

Figure 1: Extract of Books Descriptions **(D1)**

- Let consider a key $K1 = hasKey(Book)$(**title**, **hasAuthor**) and $K2 = hasKey(Book)$(**hasAuthor**, **lang**). Let **D2** be a second dataset given in **Table 2**. What would be the sameAs links that can be inferred when applying **K1** and **K2** to $D1 \times D2$ S-Key semantics. Give separated results for each key. [**3pts**]

	title	author	genre	pages	publisher	lang
b_{21}	The Age of Wrath	Eraly Abraham	history	198	Penguin	
b_{22}	Statistical Decision Theory	Pratt John	data_science	236	MIT Press	en, de
b_{23}	The New Machiavelli		fiction	198	Penguin	en
b_{24}	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_{25}	Analysis & Vol I	Tao Terence	mathematics	248	HB	

Figure 2: Book Descriptions **(D2)**

1.3.1 Answer

- (a) The 2-almost key of one property that can be discovered is **{title}**. Notice that the 2 exceptions are in the instances b_6 and b_9.
- (b) The 3-almost key composed of two properties that can be discovered is **{genre, pages}**. Notice that with those 2 properties we have an 2-almost S-Key, which qualifies obviously as a 3-almost S-Key.
- (c) A S-Key, composed of two properties, that is not an F-Key that can be discovered in the data is **{author, genre}**. Notice that, even though the **author** property has on its won 3 exceptions (b_3, b_6, b_9) when adding the property **genre**, there are no visible exceptions under S-Key semantics because of the

optimisitc approach. However, under **F-Key** semantics, as the approach is pessimistic, the empty cells are considered as possible identical matches, and therefore the instance b_4 in the property **author** could have the exact same value **{Pratt John, Tao Terence}** (as in instance b_3) which would make an exception because **genre** has already the same value for b_3 and b_4 instances.

	title	author	genre	pages	publisher	lang
b_1	The Age of Wrath	E. Abraham, Oram Andy	history	198	Penguin	en
b_2	The Trial	Kafka Frank, J. Clarck	fiction	198	R. House	
b_3	Statistical Decision Theory	Pratt John, Tao Terence	data_science	236	MIT Press	en, de
b_4	Data Mining Handbook		data_science	242	Apress	
b_5	The New Machiavelli	Wells H. G.	fiction	198	Penguin	en
b_6	Analysis & Vol I	Tao Terence, N. Robert	science	250	Apress	en
b_7	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_8	Making Software		computer_science	232	O'Reilly	
b_9	Analysis & Vol I	Tao Terence	mathematics	248	HB	en

Figure 3: Solutions

- (d) The following image highlights the **sameAs** links that can be inferred using key K_1 = **hasKey(Book)(title, author)**. Notice that, there are 2 different colours for the instance b_{25} as it can be inferred the **sameAs**(b_6, b_{25}) and **sameAs**(b_9, b_{25}).

	title	author	genre	pages	publisher	lang
b_1	The Age of Wrath	E. Abraham, Oram Andy	history	198	Penguin	en
b_2	The Trial	Kafka Frank, J. Clarck	fiction	198	R. House	
b_3	Statistical Decision Theory	Pratt John, Tao Terence	data_science	236	MIT Press	en, de
b_4	Data Mining Handbook		data_science	242	Apress	
b_5	The New Machiavelli	Wells H. G.	fiction	198	Penguin	en
b_6	Analysis & Vol I	Tao Terence, N. Robert	science	250	Apress	en
b_7	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_8	Making Software		computer_science	232	O'Reilly	
b_9	Analysis & Vol I	Tao Terence	mathematics	248	HB	en

Figure 4: K_1 sameAs Links (D1)

	title	author	genre	pages	publisher	lang
b_{21}	The Age of Wrath	Eraly Abraham	history	198	Penguin	
b_{22}	Statistical Decision Theory	Pratt John	data_science	236	MIT Press	en, de
b_{23}	The New Machiavelli		fiction	198	Penguin	en
b_{24}	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_{25}	Analysis & Vol I	Tao Terence	mathematics	248	HB	

Figure 5: K_1 sameAs Links (D2)

The following image highlights the **sameAs** links that can be inferred using key K_2 = **hasKey(Book)(author, lang)**.

	title	author	genre	pages	publisher	lang
b_1	The Age of Wrath	E. Abraham, Oram Andy	history	198	Penguin	en
b_2	The Trial	Kafka Frank, J. Clarck	fiction	198	R. House	
b_3	Statistical Decision Theory	Pratt John, Tao Terence	data_science	236	MIT Press	en, de
b_4	Data Mining Handbook		data_science	242	Apress	
b_5	The New Machiavelli	Wells H. G.	fiction	198	Penguin	en
b_6	Analysis & Vol I	Tao Terence, N. Robert	science	250	Apress	en
b_7	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_8	Making Software		computer_science	232	O'Reilly	
b_9	Analysis & Vol I	Tao Terence	mathematics	248	HB	en

Figure 6: K_2 sameAs Links (D1)

	title	author	genre	pages	publisher	lang
b_{21}	The Age of Wrath	Eraly Abraham	history	198	Penguin	
b_{22}	Statistical Decision Theory	Pratt John	data_science	236	MIT Press	en, de
b_{23}	The New Machiavelli		fiction	198	Penguin	en
b_{24}	Philosophie der Physik	Heisenberg Werner	science	197	Penguin	de
b_{25}	Analysis & Vol I	Tao Terence	mathematics	248	HB	

Figure 7: K_2 sameAs Links (D2)

2 Part 2: Rule Discovery [8 pts]

2.1 Question 4 [3 pts]

- (a)) What are the two main families of approaches of rule discovery presented in the course and what are the main steps of each of them?
- (b) What is the main challenge that raises for rule discovery under the open world assumption (OWA) and how this challenge is dealt with in AMIE system?

2.1.1 Answer

- (a) Two main families of approaches for rule discovery presented in the course are the top-down and bottom-up strategies. In a top-down approach, one begins with very general rules that cover a large part of the data and then progressively specializes them by adding more atoms to the rule's body. The main steps involve (1) **generating a broad, general rule**, (2) **incrementally specializing the rule by adding conditions**, (3) **evaluating each specialized candidate with quality metrics** (like support and confidence), and (4) **pruning those that do not meet predetermined thresholds**. In contrast, the bottom-up approach starts with specific patterns or candidate rule fragments derived directly from the data and then generalizes or merges them into broader rules. Its steps include (1) **identifying specific, highly reliable rule fragments from observed facts**, (2) **generalizing these fragments by combining them and allowing for slight variations**, and (3) **evaluating and iteratively merging these fragments until high-quality rules emerge**.
- (b) The primary challenge under the open world assumption (OWA) is dealing with incompleteness—namely, the absence of a fact in the knowledge base does not necessarily mean it is false. This lack of negative evidence can lead to an overestimation of counterexamples when evaluating a rule's confidence, thus underestimating the rule's actual quality. The AMIE system addresses this challenge by adopting the Partial Completeness Assumption (PCA). Under PCA, if an entity has at least one recorded fact for a given relation, then it is assumed that the knowledge base is complete with respect to that relation for that entity; hence, missing facts can be treated as genuine negatives. Conversely, if no such fact exists, the entity is not penalized. This PCA-based confidence measure allows AMIE to more accurately reflect the quality of rules in the face of incomplete data.

2.2 Question 5 [3.5 pts]

Let consider the following rules r_1 and r_2 that we assume to be discovered by AMIE tool on the data presented in **Table 8**. The atoms **predicate(x, y)** are written as **(?x predicate ?y)**:

$$r_1 : \textbf{(?b spouse ?a)} \implies \textbf{(?a spouse ?b)}$$
$$r_2 : \textbf{(?a nationality ?b)} \implies \textbf{(?a deathplace ?b)}$$

	spouse	birthplace	deathplace	nationality
#Bob	#Mary	France		France
#Mary		Greece	France	France
#Momo	#Yue	Algeria	Algeria	
#Katsu	#Yori	Senegal	Italy	Italy
#Yue	#Momo	China	China	China
#Yori		Ukraine		France

Figure 8: People Descriptions Dataset

Considering people descriptions presented in **Table 8** compute the support, the standard confidence and the PCA-confidence for r_1 and r_2.

2.2.1 Answer

The first rule is:

$$r_1 : \overbrace{\textbf{(?b spouse ?a)}}^{Body} \implies \overbrace{\textbf{(?a spouse ?b)}}^{Head}$$

From these, the actual triples bodies (?b spouse ?a) in the data are:

(Bob, Mary), **(Momo, Yue)**, **(Katsu, Yori)**, **(Yue, Momo)**

We check whether the **head** "(?a spouse ?b)" also appears in the data:

1. spouse(Mary, Bob): ✗
2. spouse(Yue, Momo): ✓
3. spouse(Yori, Katsu): ✗
4. spouse(Momo, Yue): ✓

So, we have 2 successes out of 4 body matches, then we conclude that the support(r_1) = 2.

The standard confidence follows by the fact that the body is satisfied 4 times, and the head is satisfied 2 times, therefore the standard_conf(r_1) = $\frac{2}{4}$ = 0.5. Finally, we study the PCA-Confidence(r_1). Under the Partial Completeness Assumption, we only treat "missing" head facts as genuine negatives if the subject in the head has at least one known spouse triple. Concretely:

1. spouse(Bob, Mary) ✓ and as spouse(Mary, Bob) is not present, we dont count it as negative.
2. spouse(Momo, Yue) ✓ and spouse(Yue, Momo) ✓, therefore this is a sucess rule inference.
3. spouse(Katsu, Yori) ✓ and as spouse(Yori, Katsy) is not present, we don't count it as negative.

Then, there are no counter examples under PCA, which means that the confidence is:

$$\text{PCA-Confidence}(r_1) = \frac{\text{support}(r_1)}{\text{support}(r_1) + cex_{PCA}} = \frac{2}{2+0} = 1$$

The second rule is:

$$r_2 : \overbrace{\textbf{(?a nationality ?b)}}^{Body} \implies \overbrace{\textbf{(?a deathplace ?b)}}^{Head}$$

From these, the actual triples bodies (?a nationality ?b) in the data are:

(Bob, France), **(Mary, France)**, **(Katsu, Italy)**, **(Yue, China)**, **(Yori, France)**

We check whether the **head** "(?a deathplace ?b)" also appears in the data:

1. deathplace(Bob, France): ✗
2. deathplace(Mary, France): ✓
3. deathplace(Katsu, Italy): ✓
4. deathplace(Yue, China): ✓
5. deathplace(Yori, France): ✗

So, we have 3 successes out of 5 body matches, then we conclude that the support(r_2) = 3.

The standard confidence follows by the fact that the body is satisfied 5 times, and the head is satisfied 3 times, therefore the standard_conf(r_5) = $\frac{3}{5}$ = 0.6. Finally, we study the PCA-Confidence(r_2). Under the Partial Completeness Assumption, we only treat "missing" head facts as genuine negatives if the subject in the head has at least one known deathplace triple. Concretely:

1. nationality(Bob, France) ✓ and as deathplace(Bob, France) is not present, we dont count it as negative.
2. nationality(Mary, France) ✓ and deathplace(Mary, France) ✓, therefore this is a sucess rule inference.
3. nationality(Katsu, Italy) ✓ and as deathplace(Katsu, Katsy) ✓, therefore this is another sucess rule inference.
4. nationality(Yue, China) ✓ and as deathplace(Yue, China) ✓, therefore this is a sucess rule inference.
5. nationality(Yori, France) ✓ and as deathplace(Yori, France) is not present, we dont coun't it as negative.

Then, there are no counter examples under PCA, which means that the confidence is:

$$\text{PCA-Confidence}(r_2) = \frac{\text{support}(r_2)}{\text{support}(r_2)+cex_{PCA}} = \frac{3}{3+0} = 1$$

2.3 Question 6 [1.5 pts]

Let us consider the following rule:

$$r_3\text{: (?a birthplace ?b) and (?a deathplace ?b)} \implies \text{(?a nationality ?b)}$$

that is discovered on data described in **Table 8**. Give the SPARQL query that translates the rule r_3 for allowing using it for predicting new facts for **nationality** predicate.

2.3.1 Answer

A convenient way to express the rule in SPARQL is via a CONSTRUCT query that finds all **subjects** ?a and **objects** ?b satisfying both birthplace and deathplace, while ensuring ?a nationality ?b does not already exist. For instance:

```
CONSTRUCT {
?a :nationality ?b
}
WHERE {
  ?a :birthplace ?b .
  ?a :deathplace ?b .
  FILTER NOT EXISTS {
    ?a :nationality ?b
  }
}
```

This query "discovers" new triples (?a :nationality ?b) whenever both (?a :birthplace ?b) and (?a :deathplace ?b) are present in the data.