Semantic Web

Tutorial 7-8

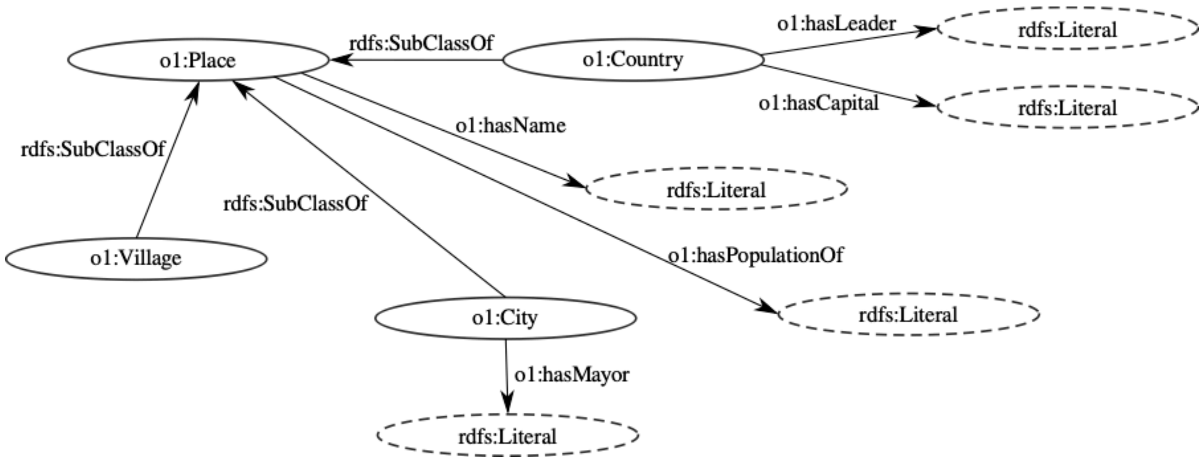
Iryna Dubrovska

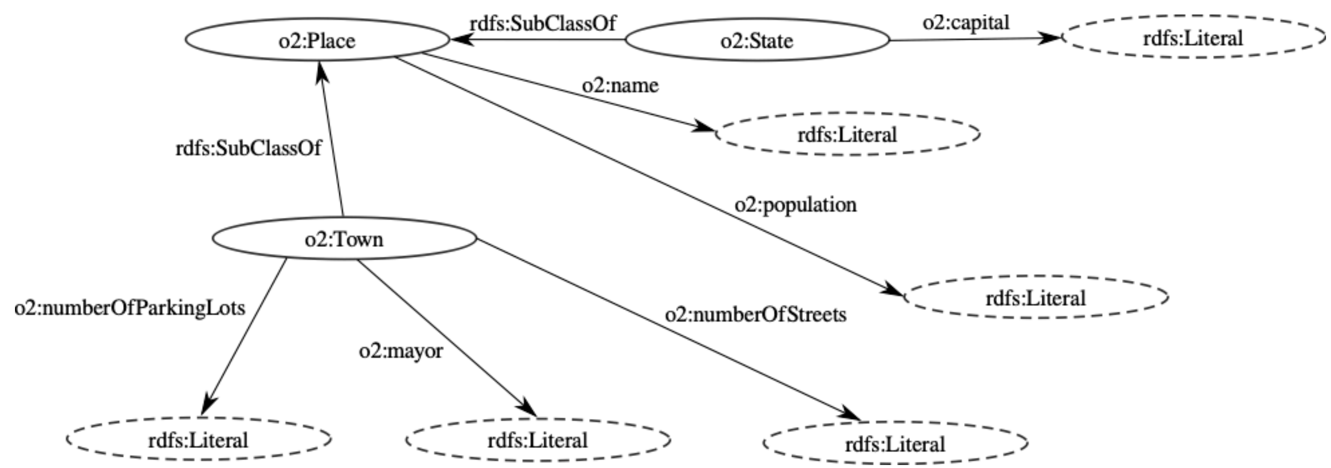
WeST
People and Knowledge Networks

Tutorial 7

Task 1. Ontology Matching

Consider two ontologies O1 and O2:





Task 1. Ontology Matching

Levenshtein Distance

The Levenshtein Distance, or Edit Distance, measures the difference between two sequences by counting the *minimum* number of edit operations required to transform one sequence into the other.

Edit operations:

- Replace: “aa” → “ab”
- Insert: “aa” → “aab”
- Delete: “aa” → “a”
-

Properties of Levenshtein distance:

- It is zero if and only if two strings are equal: $dist(a,b) = 0 \iff a = b$
- It’s symmetric: $dist(a,b) = dist(b,a)$
- The value is at most the length of the longer string: $dist(a,b) \leq max(|a|,|b|)$
- The value is at least the size difference of the strings: $abs(|a| - |b|) \leq dist(a,b)$
- Triangle inequality - the distance between two strings is no greater than the sum of their distances from some other string: $dist(a,b) \leq dist(a,c) + dist(b,c)$

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ							
P							
A							
G							
E							

Assuming we are in a cell x, the following key can be used:

replace	*insert*
delete	x

Value in x is equal to:

- The minimum of the values in those three cells, if the two corresponding characters match.
- The minimum of the values in those three cells + 1, if the two corresponding characters do not match.

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1						
A	2						
G	3						
E	4						

Key —>

replace	*insert*
delete	x

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1						
A	2						
G	3						
E	4						

Key —>

replace	*insert*
delete	x

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1	0					
A	2						
G	3						
E	4						

Key —>

replace	*insert*
delete	x

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1	0					
A	2						
G	3						
E	4						

Key —>

replace	*insert*
delete	x

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1	0	1				
A	2						
G	3						
E	4						

Key —>

replace	*insert*
delete	*x*

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1	0	1	2			
A	2						
G	3						
E	4						

Key —>

replace	*insert*
delete	x

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1	0	1	2	3	4	5
A	2	1	1	1	2	3	4
G	3	2	2	2	2	3	4
E	4	3	3	3	3	2	**3**

Key —>

replace	*insert*
delete	x

Task 1. Ontology Matching

Levenshtein Distance

Levenshtein distance through the table:

	ϵ	P	L	A	C	E	S
ϵ	0	1	2	3	4	5	6
P	1	0	1	2	3	4	5
A	2	1	1	1	2	3	4
G	3	2	2	2	2	3	4
E	4	3	3	3	3	2	**3**

p	l	a	c	e	s
p	a	g	e		
0	**1**	**2**	**3**	**4**	**5**

p	l	a	c	e	s
p	*	a	g	e	*
0	**1**	**0**	**2**	**0**	**3**

Key —>

replace	*insert*
delete	*x*

Task 1. Ontology Matching

Levenshtein Distance

1.1 What are the Levenshtein distances between all possible pairs of *o1:hasLeader*, *o1:hasName*, and *o1:hasCapital*?

Answer:

(hasLeader, hasName) = 4

(hasLeader, hasCapital) = 7

(hasName, hasCapital) = 6

replace	*insert*
delete	*x*

		h	a	s	L	e	a	d	e	r
	0	1	2	3	4	5	6	7	8	9
h	1	0	1	2	3	4	5	6	7	8
a	2	1	0	1	2	3	3	4	5	6
s	3	2	1	0	1	2	3	4	5	6
N	4	3	2	1	1	2	3	4	5	6
a	5	4	2	2	2	2	2	3	4	5
m	6	5	3	3	3	3	3	3	4	5
e	7	6	4	4	4	3	4	4	3	4

Task 1. Ontology Matching

Levenshtein Distance

1.1 What are the Levenshtein distances between all possible pairs of *o1:hasLeader*, *o1:hasName*, and *o1:hasCapital*?

Answer:

(hasLeader, hasName) = 4

(hasLeader, hasCapital) = 7

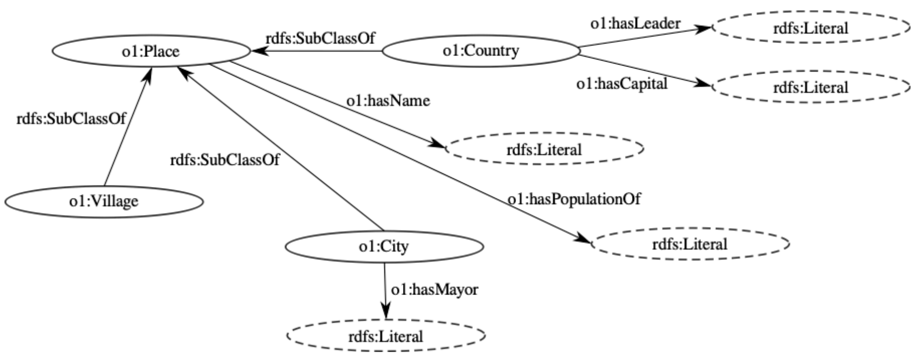
(hasName, hasCapital) = 6

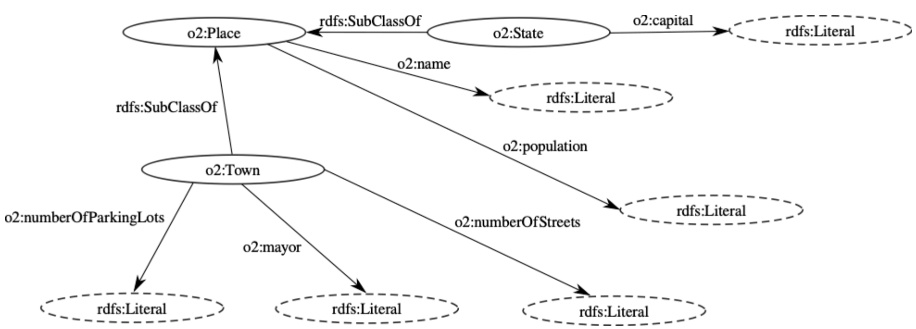
h	a	s	L	e	a	d	e	r
h	a	s	N	*	a	m	e	*
0	**0**	**0**	**1**	**2**	**0**	**3**	**0**	**4**

Task 1. Ontology Matching

Suffix Similarity

1.2 Calculate the suffix similarity for both *o2:mayor* and *o2:population* to elements in O1. Indicate the pair(s) with the confidence value higher than zero.





Answer:

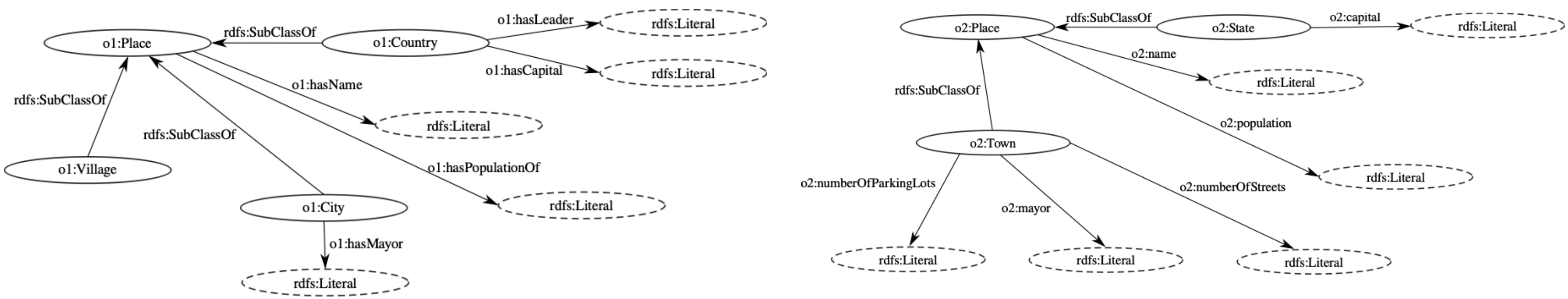
Confidence value of (o2:mayor, o1:hasMayor) = 0.62 or 5/8

(o2:population, o1:hasPopulationOf) —> No suffix similarity with confidence value > 0

Task 1. Ontology Matching

Graph-based Techniques

1.3 Using graph-based techniques, identify two pairs of entities between non-leaf elements of O1 and O2 that are similar. For each pair (e1, e2) that you provide, explain why they are similar.



Answer:

(o1:City, o2:State)

(o1:Place, o2:Place)

Task 2. Ontology Alignment

Given two strings s1 and s2, let N be the size of the longest string, and L be the Levenshtein distance between s1 and s2. The normalised Levenshtein distance is (N – L)/N.

http://example.org/places/Berlin:

s = *http://example.org/places/Berlin*
s' = *http://dbpedia.org/page/Berlin*

The normalized Levenshtein distance:
(N - L) / N = (32 - 10) / 32 = 0.6875

s = *http://example.org/places/Berlin*
s' = *https://en.wikipedia.org/wiki/Berlin*

The normalized Levenshtein distance:
(N - L) / N = (36 - 17) / 36 = 0.5278

http://example.org/institutions/Reichstag:

s = *http://example.org/institutions/Reichstag*
s' = *https://en.wikipedia.org/wiki/Reichstag_building*

The normalized Levenshtein distance:
(N - L) / N = (48 - 30) / 48 = 0.375

s = *http://example.org/institutions/Reichstag*
s' = *https://en.dbpedia.org/page/Reichstag_building*

The normalized Levenshtein distance:
(N - L) / N = (42 - 28) / 42 = 0.333

Tutorial 8

1. SPARQL

Consider the following excerpts of RDF datasets containing patient information from a single hospital (1.1) and information about flights (1.2).

Study the data structure apparent from the snippets. Based on that structure, formulate SPARQL queries for each of the specified tasks.

1. SPARQL

1.1 Hospital

1.1.1 List all the patients that have been assigned to `x:DrCarolineSmith` for their admissions.

```
SELECT ?patient
WHERE {
?patient rdf:type xo:Patient.
?patient xo:admission ?adm.
?adm xo:physician x:DrCarolineSmith }
```

1.1.2 Count the number of medical reports for all the patients ever admitted to `xo:ICU`.

```
SELECT ?patient (COUNT(?report) as ?numReports)
WHERE {
?patient rdf:type xo:Patient.
?patient xo:report ?report.
?patient xo:admission ?adm.
?adm xo:careUnit xo:ICU.
}
GROUP BY ?patient
```

1. SPARQL

1.1 Hospital

1.1.3 Return the 5 most costly care units of the hospital together with these costs. For that, sum up the total costs of care known for each care unit respectively.

```
SELECT ?careUnit (SUM(?cost) as ?totalCost)
WHERE {
?adm xo:careUnit ?careUnit.
?adm xo:medicalBill ?bill.
?bill xo:cost ?cost
}
GROUP BY ?careUnit
ORDER BY DESC(?totalCost)
LIMIT 5
```

1. SPARQL

1.2 Flights

1.2.1 Write a query that lists flights from Rome to destination London that take between 2 and 2.5 hours.

```
SELECT ?flight
WHERE {
?flight rdf:type ex:Flight.
ex:start ex:Rome.
ex:destination ex:London.
ex:duration ?duration.
FILTER (?duration >= "2"^^xsd:nonNegativeInteger && ?duration <=
“2.5"^^xsd:nonNegativeInteger)
}
```

1. SPARQL

1.2 Flights

1.2.2 Write a query that retrieves a list of all flights that have 2 different intermediate stops (at which they have a layover).

```
SELECT ?flight
WHERE {
?flight rdf:type ex:Flight.
ex:connectsTo ?cflight.
ex:layoverAt ?l1, ?l2.
?cflight rdf:type ex:Flight.
FILTER (?l1 != ?l2)
}
```

1. SPARQL

1.2 Flights

1.2.3 Write a query that lists all the flights with destination Los Angeles starting from Hamburg or Berlin.

```
SELECT ?flight
WHERE {
?flight rdf:type ex:Flight.
?flight ex:destination ex:Los_Angeles.
{?flight ex:start ex:Hamburg.}
UNION
{?flight ex:start ex:Berlin.}
}
```

2. Querying DBPedia

2.1 List all the countries (instances of class dbo:Country) that have a title of their leader containing the word "president".
Note: pay attention to case sensitivity.

```
SELECT distinct ?country
WHERE {
?country rdf:type dbo:Country.
?country dbp:leaderTitle ?title
 filter contains(lcase(str(?title)), "president" )
}
ORDER BY ?country
```

2. Querying DBPedia

2.2 *Zombies*: List the persons (instances of dbo:Person) born before 1867 that are still alive today (if DBPedia is to be believed). List their names and birth dates. Count, how many zombies there are.

Hint: create a filter also checking, whether an optional pattern (e.g. for matching a death date) was bound.

```
SELECT (count) DISTINCT ?person ?name ?birth
WHERE {
?person rdf:type dbo:Person.
?person foaf:name ?name.
?person dbo:birthDate ?birth.
OPTIONAL { ?person dbo:deathDate ?death.}
FILTER (?birth < "1867-01-01" ^^xsd:dateTime && !bound(?death))
}
```

2. Querying DBPedia

2.3 *Bound by fate:* List 10 pairs of (distinct) persons that share the dates for both birth and death (according to DBPedia). They should all have been born in 1927 or after.

```
SELECT DISTINCT ?person ?name ?birth ?death
                             ?person2 ?name2 ?birth2 ?death2
WHERE {
?person rdf:type dbo:Person.
?person foaf:name ?name.
?person dbo:birthDate ?birth.
?person dbo:deathDate ?death.
?person2 rdf:type dbo:Person.
?person2 foaf:name ?name2.
?person2 dbo:birthDate ?birth2.
?person2 dbo:deathDate ?death2.

FILTER ("1927-01-01" ^^xsd:dateTime < ?birth && ?birth = ?birth2  &&
                 ?death = ?death2  && ?person != ?person2)
}
LIMIT 10
```

0. Exercise

List authors who wrote only books with least 1000 pages, i.e., they did not write any books with less than 1000 pages.

`Author` is a `type` `Person`.

`Book` is a `type` `Book`.

`Book` has an `author`.

`Book` has `numberOfPages`.

0. Exercise

List authors who wrote only books with least 1000 pages, i.e., they did not write any books with less than 1000 pages.

```
SELECT DISTINCT ?author
        WHERE {
?author rdf:type dbo:Person.
?book dbo:author ?author.
?book rdf:type dbo:Book.
?book dbo:numberOfPages ?numPages.
FILTER (?numPages > 999)
}
     GROUP BY ?author
```

```
SELECT DISTINCT ?author
         WHERE {
?author rdf:type dbo:Person.
?book dbo:author ?author.
?book rdf:type dbo:Book.
?book dbo:numberOfPages ?numPages.
}
     GROUP BY ?author
     HAVING ( MIN(?numPages) > 999 )
```

Questions?