Building & Mining Knowledge Graphs

(KEN4256)

Lab 3: Constructing and linking KGs from structured data

Maastricht University

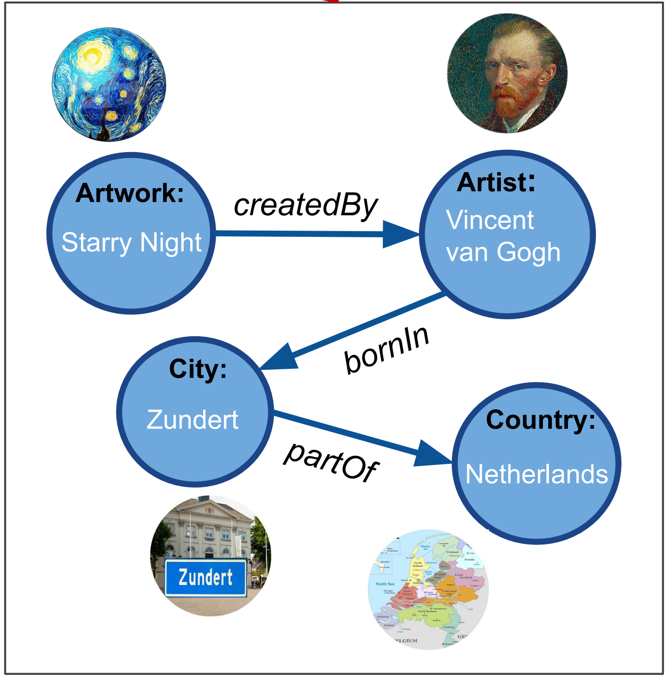
Institute of Data Science

https://tinyurl.com/rrvj6bl

Recap

“Vincent van Gogh was a Dutch artist born in Zundert, the Netherlands. One of the most famous artworks created by him is ‘The Starry Night’ oil on canvas painting.”

Data source



Conceptualisation

```
1.    wd:Q45585
              ex:createdBy ex:Vincent_van_Gogh ;
              rdf:type ex:Artwork ;
              rdfs:label “The Starry Night”@en .
2.    ex:Vincent_van_Gogh
              rdf:type ex:Artist ;
              ex:bornIn ex:Zundert ;
3.    ex:Zundert
              ex:partOf ex:Netherlands ;
              rdf:type ex:City ;
              ex:hasAge
              “37”^^xsd:nonNegativeInteger .
4.    ex:Netherlands rdf:type ex:Country .
```

(RDF) Knowledge Graph

Construction

Constructing KGs from existing data

Web pages

Research articles

Books

Relational Databases

Web APIs

Constructing KGs from existing data

Web pages

Research articles

Books

Relational Databases

Web APIs

Structured vs. unstructured

Constructing KGs from existing data

Web pages

Research articles

Books

Relational Databases

Web APIs

Unstructured: ?

Constructing KGs from existing data

Web pages

Research articles

Books

Relational Databases

Web APIs

Unstructured:

- Data which has no qualifying or contextual information (e.g. metadata or data model),
- Is specified in a language or format which has no specification

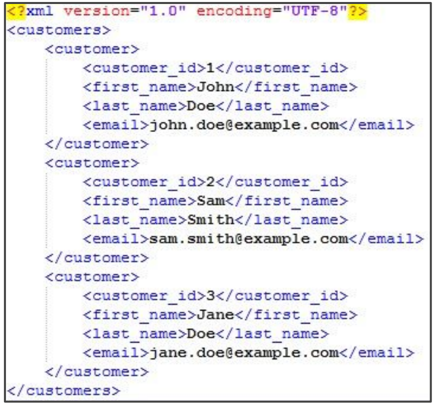
Constructing KGs from unstructured text

"Vincent van Gogh was a Dutch artist born in Zundert, the Netherlands. One of the most famous artworks created by him is 'The Starry Night' oil on canvas painting."

Information Extraction (IE) techniques:

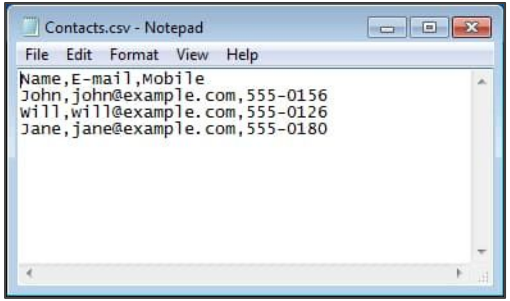
- Named Entity Recognition (NER)
- Relation Extraction (RE)

Constructing KGs from structured data



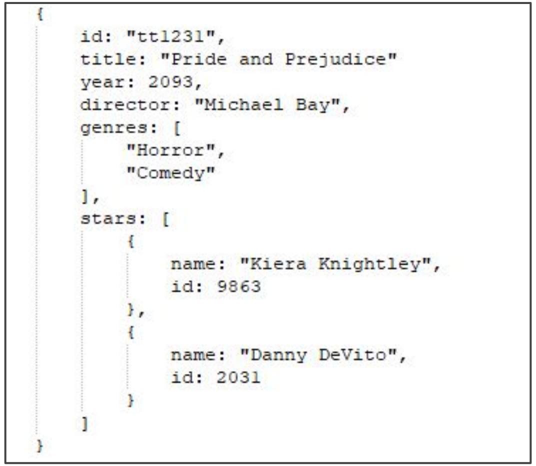
```
<?xml version="1.0" encoding="UTF-8"?>
<customers>
    <customer>
        <customer_id>1</customer_id>
        <first_name>John</first_name>
        <last_name>Doe</last_name>
        <email>john.doe@example.com</email>
    </customer>
    <customer>
        <customer_id>2</customer_id>
        <first_name>Sam</first_name>
        <last_name>Smith</last_name>
        <email>sam.smith@example.com</email>
    </customer>
    <customer>
        <customer_id>3</customer_id>
        <first_name>Jane</first_name>
        <last_name>Doe</last_name>
        <email>jane.doe@example.com</email>
    </customer>
</customers>
```

XML (eXtensible Markup Language)



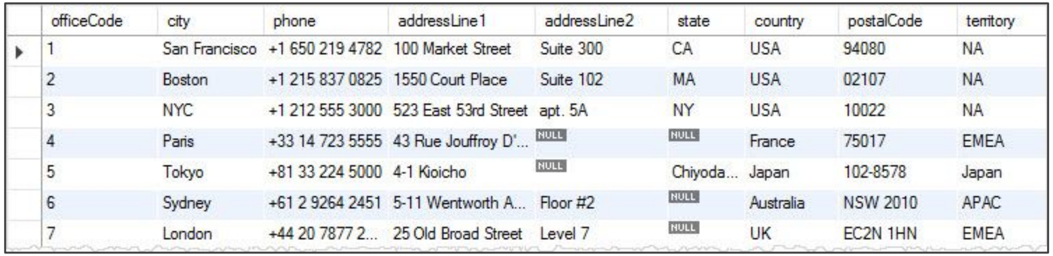
```
Name,E-mail,Mobile
John,john@example.com,555-0156
Will,will@example.com,555-0126
Jane,jane@example.com,555-0180
```

CSV (Comma-Separated Values)



```
{
    id: "tt1231",
    title: "Pride and Prejudice"
    year: 2093,
    director: "Michael Bay",
    genres: [
        "Horror",
        "Comedy"
    ],
    stars: [
        {
            name: "Kiera Knightley",
            id: 9863
        },
        {
            name: "Danny DeVito",
            id: 2031
        }
    ]
}
```

JSON (JavaScript Object Notation)



	officeCode	city	phone	addressLine1	addressLine2	state	country	postalCode	territory
▸	1	San Francisco	+1 650 219 4782	100 Market Street	Suite 300	CA	USA	94080	NA
	2	Boston	+1 215 837 0825	1550 Court Place	Suite 102	MA	USA	02107	NA
	3	NYC	+1 212 555 3000	523 East 53rd Street	apt. 5A	NY	USA	10022	NA
	4	Paris	+33 14 723 5555	43 Rue Jouffroy D'...	NULL	NULL	France	75017	EMEA
	5	Tokyo	+81 33 224 5000	4-1 Kioicho	NULL	Chiyoda...	Japan	102-8578	Japan
	6	Sydney	+61 2 9264 2451	5-11 Wentworth A...	Floor #2	NULL	Australia	NSW 2010	APAC
	7	London	+44 20 7877 2...	25 Old Broad Street	Level 7	NULL	UK	EC2N 1HN	EMEA

Relational databases

Converting structured data to RDF

- Variety of technologies available to do this
- Choice of technology depends on which format(s) we are converting from: e.g. CSV, XML, SQL, JSON etc.
- Possible to create own custom programming scripts (Python, Java, R, PHP) to do this
- Extract Transform Load (ETL) tools: no perfect solution or standard for RDF

Converting structured data to RDF: rel. databases

R2RML: Relational Databases to RDF Mapping Language (W3C recommendation)

Consists of a standard language to define mappings between entities in database and entities in an output KG, and tools to execute the mapping on the database to generate the KG:

- Easier to write
- Easier to share
- Easier to maintain

Why use this as opposed to custom scripts?

Only for **Relational** Databases.

A unified solution: RML (RDF Mapping Language)

http://rml.io/

A mapping language to rule them all.

Extends R2RML spec. to allow conversion to RDF from additional data formats:

- XML
- JSON
- CSV

Drawback: current implementations do not have scalable performance to deal with very large datasets.

RML workflow

1. Define a **mapping file** to map your chosen data source (in a given format e.g. CSV) to RDF triples
- **Conceptualise** how you want your triples to look (which elements of the data should be mapped to **subjects**, which to **predicates** and which to **objects**?
- Define this in your mapping file using RML rules
2. Execute the RML processor to apply your mapping to the input data (requires Java Runtime Environment installed): https://github.com/RMLio/rmlmapper-java/releases/download/v4.3.1/rmlmapper.jar

```
java -jar rmlmapper.jar -m /data/rml/mapping.ttl -o /data/rml/output.nt
```

- -m: mapping file path
- -o: output file path

Windows path syntax: C:\data\rml\mapping.ttl

RML mapping files

Uses Turtle syntax to express series of user-specified rules for converting data to RDF triples

Components of an RML mapping file:

- Prefix section:
- Triples Map:
  - Logical Source:
    - Data sources:
    - Reference formulation:
    - Iterator:
  - Subject Map:
  - Predicate Object Map:
    - Predicate Map:
    - Object Map:

RML mapping files

Uses Turtle syntax to express series of user-specified rules for converting data to RDF triples

Components of an RML mapping file:

- Prefix section: usually at the top of the file like in RDF Turtle
- Triples Map: one or more - for defining rules (“patterns”) to generate RDF triples
  - Logical Source:
    - Data sources: name and path of the input data file(s)
    - Reference formulation: tells RML how to read the elements of your dataset (what kind of format is this data in)
    - Iterator: tells how to iterate over the elements of the data
  - Subject Map: URI pattern stating how these triples’ subjects (and types) should be generated
  - Predicate Object Map:
    - Predicate Map: URI pattern stating how these triples’ predicates should be generated
    - Object Map: URI pattern stating how these triples’ subjects (and types) should be generated

Converting CSV to RDF using RML

Logical Source

```
<TriplesMapCsv>
    a rr:TriplesMap;
    rml:logicalSource [
      rml:source "/data/rml/countryInfo.csv";
      rml:referenceFormulation ql:CSV
    ];
```

short for rdf:type !

Path to the source of the data (filepath)

Our input file is a CSV file

NB: the rr, rml and ql prefixes refer to terms (entities) within the RML specification
NNB: RML is following the linked data principles!

@prefix rr: <http://www.w3.org/ns/r2rml#>.
@prefix rml: <http://semweb.mmlab.be/ns/rml#>.
@prefix ql: <http://semweb.mmlab.be/ns/ql#>.

Look up any RML URI from the spec! What do you see?

Converting CSV to RDF using RML

Subject Map

```
<TriplesMapCsv>
   a rr:TriplesMap;
   rml:logicalSource [
     rml:source "/data/rml/countryInfo.csv";
     rml:referenceFormulation ql:CSV
   ];
   rr:subjectMap [
      rr:template "http://geonames.org/country/{ISO3}" ;
      rr:class gn:country
   ];
```

ISO3	Country	Population	Continent
FRA	France	70,000,000	EU

<http://example.com/country/FRA> a gn:country .

Create the subject URI

ISO3	Country	Population	Continent
FRA	France	70,000,000	EU

<http://geonames.org/country/FRA> a gn:country .

<http://geonames.org/country/FRA> rdfs:label “France” .

```
<TriplesMapCsv>
   a rr:TriplesMap;
   rml:logicalSource [
     rml:source "/data/rml/countryInfo.csv";
     rml:referenceFormulation ql:CSV
   ];
   rr:subjectMap [ rr:template "http://example.com/country/{ISO3}" ;
     rr:class ex:country ];
   rr:predicateObjectMap [
     rr:predicate rdfs:label ;
     rr:objectMap [ rml:reference "Country" ]
   ] .
```

Path to the source of the data (filepath)

Create the subject URI

Map the Country column as object

ISO3	Country	Population	Continent
FRA	France	70,000,000	EU

<http://geonames.org/country/FRA> a gn:country .
<http://geonames.org/country/FRA> rdfs:label “France” .
<http://geonames.org/country/FRA> gn:population “70000000”^^xsd:integer .

```
<TriplesMapCsv>
   a rr:TriplesMap;
   rml:logicalSource [
     rml:source "/data/rml/countryInfo.csv";
     rml:referenceFormulation ql:CSV
   ];
   rr:subjectMap [ rr:template "http://geonames.org/country/{ISO3}" ;
     rr:class ex:country ];
   rr:predicateObjectMap [
     rr:predicate gn:population ;
     rr:objectMap [ rml:reference "Population" ;
                    rr:datatype xsd:integer ]
   ] ;
```

Path to the source of the data (filepath)

Create the subject URI

The object is an integer

ISO3	Country	Population	Continent
FRA	France	70,000,000	EU

```
<http://geonames.org/country/FRA> a gn:country .
<http://geonames.org/country/FRA> rdfs:label “France” .
<http://geonames.org/country/FRA> gn:population “70000000”^^xsd:integer .
<http://geonames.org/country/FRA> gn:partOf
<http://geonames.org/continent/EU> .
```

```
<TriplesMapCsv>
    a rr:TriplesMap;
    rml:logicalSource [
      rml:source "/data/rml/countryInfo.csv";
      rml:referenceFormulation ql:CSV
    ];
    rr:subjectMap [ rr:template "http://geonames.org/country/{ISO3}" ;
      rr:class gn:country ];
    rr:predicateObjectMap [
      rr:predicate gn:continent ;
      rr:objectMap [ rr:template "http://geonames.org/continent/{Continent}" ]
    ] ;
```

Path to the source of the data (filepath)

Create the subject URI

Create URI as object

```
<Root >
   <data>
    <record>
         <country key="FRA">France</country>
         <year>1960</year>
         <value>62651474946.6007</value>
     </record>
```

XML

<http://data.wordbank.org/country/FRA> a wb:country .
<http://data.wordbank.org/country/FRA> rdfs:label “France” .

```
<TriplesMapXml>
   a rr:TriplesMap;
  rml:logicalSource [
    rml:source "/data/rml/gdp_worldbank.xml";
    rml:referenceFormulation ql:XPath;
    rml:iterator "/Root/data/record"
  ] ;
```

Path to the source of the data (filepath)
And XPath iteration

Create yearly entry for each country

```
  rr:subjectMap [ rr:template "http://data.worldbank.org/{country/@key}/gdp/{year}" ;
    rr:class wb:GdpEntry ];
  rr:predicateObjectMap [
    rr:predicate wd:country ;
    rr:objectMap [ rml:reference "country/@key" ]
  ] .
```

Define the country of the GDP entry

Your Tasks

Task 1

- Download “**example.csv**” from Student Portal (under Lab 3)
- Create an RML mapping file to generate the RDF triples from this data
- Execute RML mapper on the mapping file and input data to generate an output triples document
- Check the output file to verify if the data was successfully converted

How to execute the mapping file:

```
java -jar rmlmapper.jar -m mapping.ttl -o output.nt
```

Material

- RML Specification: http://rml.io
- http://rml.io/RML_examples.html

https://tinyurl.com/rrviCbl

Using a shared vocabulary (ontology)

Using a commonly defined term for types in your KG is a best practice to allow people to more easily reuse and interpret the meaning of your KG:

- DBpedia ontology: http://mappings.dbpedia.org/server/ontology/classes/

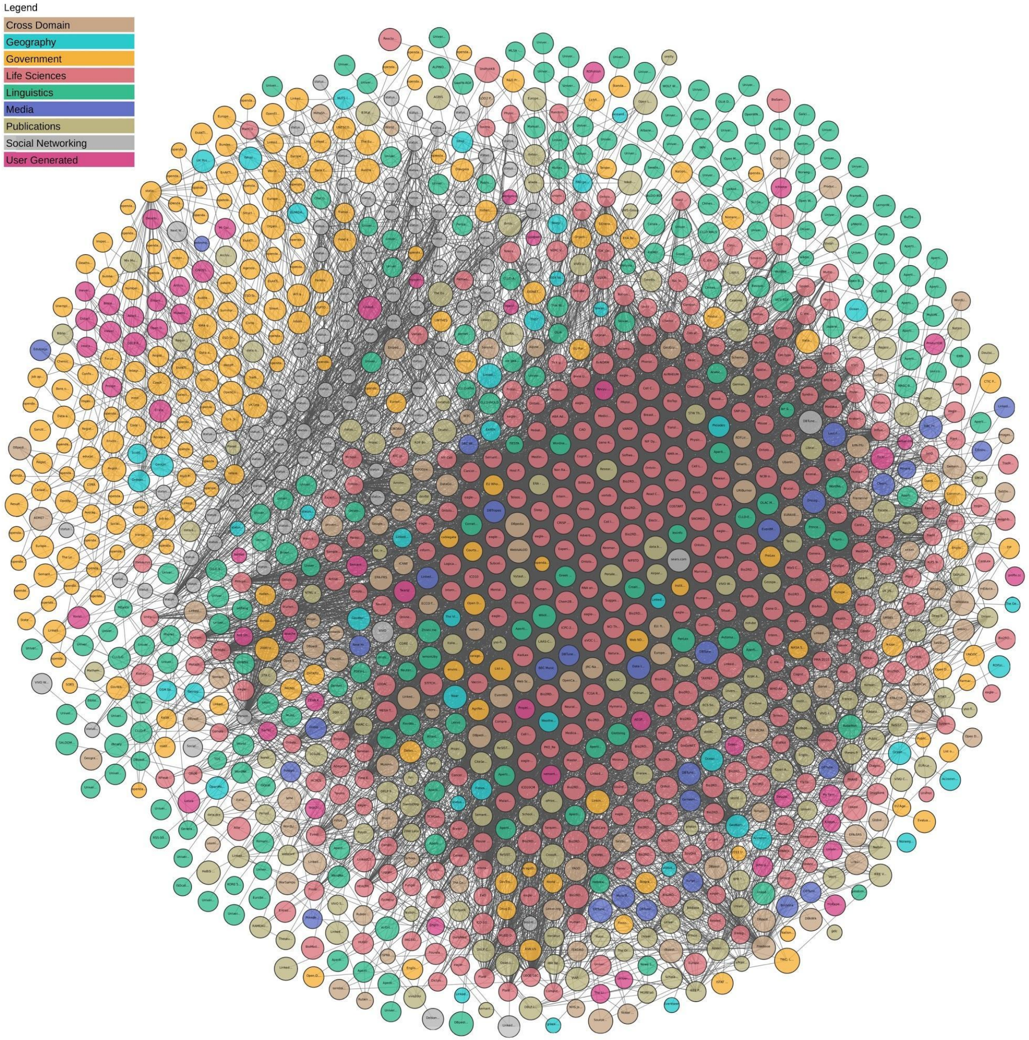
Which class best defines the meaning of your subject?

Questions?

Linking

Linked Data Principles

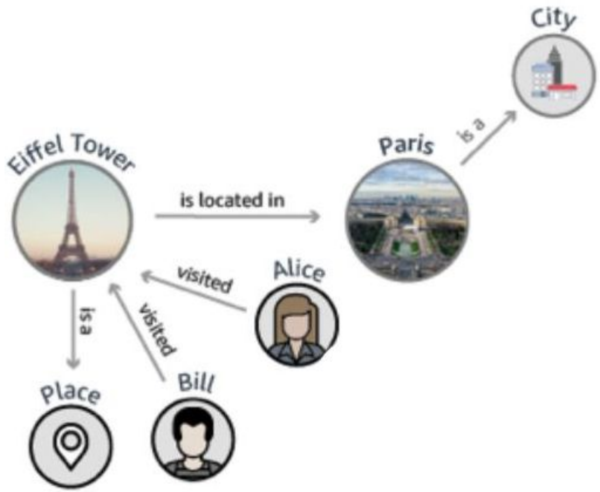
- Use **Uniform Resource Identifiers (URIs)** as names for things.
- Use **HTTP URIs**, so that people can look up those names.
- When someone looks up a URI, provide **useful information**, using the standards (RDF, RDFS, OWL, SPARQL).
- Include **links** to other URIs, so that they can discover more things.



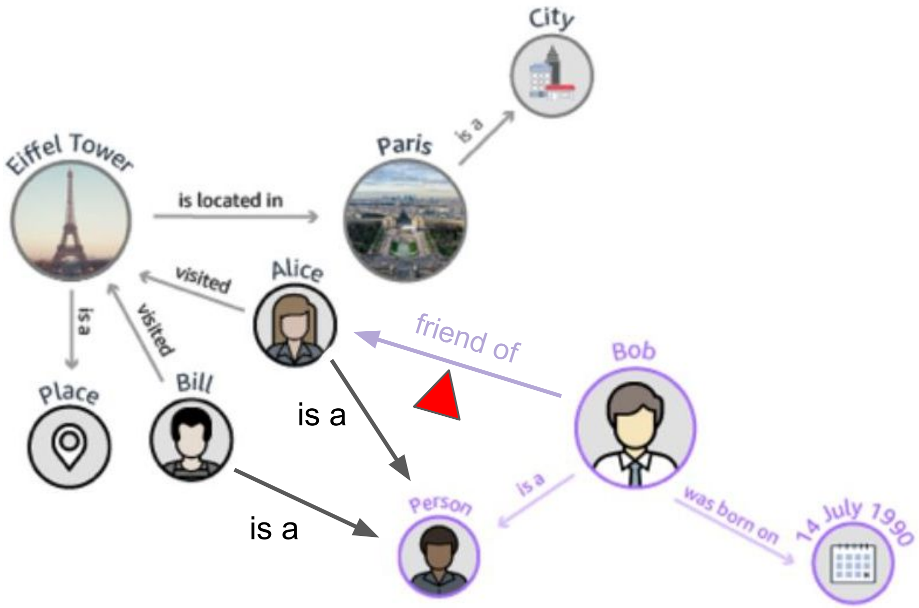
The Linked Open Data Cloud from lod-cloud.net

https://lod-cloud.net/

1,224 datasets with **16,113 links** (as of June 2018) travel network



travel network



social network

▲ linking relations

Interlinking Datasets

an *external RDF link* is an **RDF triple** in which the

subject of the triple is a URI reference in the namespace of one data set,

while the **predicate** and/or **object** of the triple are URI references pointing into the namespaces of other data sets.

Example:

<http://worldbank.org/India>

<http://www.w3.org/2002/07/owl#sameAs>

<http://dbpedia.org/resource/India>

Types of Links

- **Relationship Links** point at related things in other data sources, for instance, other people, places or genes.
  - Example: ex:Amrapali foaf:knows ex:Tim-Berners-Lee
- **Identity Links** point at URI aliases used by other data sources to identify the *same* real-world object or abstract concept.
  - Example: wb:India owl:sameAs db:India
- **Vocabulary Links** point from data to the *definitions of the vocabulary terms* that are used to represent the data, as well as from these definitions to the definitions of related terms in other vocabularies. Vocabulary links make data self-descriptive and enable Linked Data applications to understand and integrate data across vocabularies.
  - Example: dbo:Country rdfs:subClassOf dbo:PopulatedPlace
  - Example: ex:Amrapali rdf:type schema:Person

Considerations before Interlinking

- What is the added value of the new data in the target KG?
- Is the target KG and its namespace under stable ownership and active maintenance? **Why is it important?**
- Are the URIs in the data set stable and unlikely to change? **Why is it important?**
- Are there outgoing links to other KGs so that applications can tap into a network of interconnected graphs?

Choosing Predicates for Linking

- How *widely* is the predicate already used for linking by other KGs?
- Is the vocabulary well *maintained* and properly published with de-referenceable URIs? What does “de-referencable” mean?
- How semantically accurate is the relationship? Do the URIs refer to the *same* thing or are they *related*?

Examples:

- owl:sameAs
- skos:broader (similar to rdfs:subClassOf)
- skos:narrower

Automatic Interlinking

Link Discovery - Similarity-based Approaches

Goal: Discover related entities across knowledge bases

- Use similarity-based linkage heuristics, which may compare multiple properties of the entities that are to be interlinked as well as properties of related entities

Example: Linking entities (geographical places) in GeoNames and DBpedia by comparing their:

- names using a **string similarity** function
- longitude and latitude values using a **geographic matcher**
- name of geographical region (e.g. country/continent) in which the places are located
- population count
- etc.

Link Discovery Tools

- **LIMES** – Link Discovery Framework for Metric Spaces provides time-efficient approaches for discovery and computing the results of link specifications.
- **Silk** - A Link Discovery Framework for the Web of Data tool for discovering relationships between data items within different Linked Data sources. Data publishers can use Silk to set RDF links from their data sources to other data sources on the Web.
- **TopBraid Composer** (ontology editor made by TopQuadrant) has a wizard for linking ontology instances to corresponding DBpedia concepts.
- **SemMF** is a framework for calculating semantic similarity between objects that are represented as arbitrary RDF graphs. The framework allows taxonomic and non-taxonomic concept matching techniques to be applied to selected object properties.

Interlinking using LIMES tool (**Li**nk Discovery Framework for **Me**tric **S**paces)

Download latest JAR file from:

https://github.com/dice-group/LIMES/releases

User manual: http://dice-group.github.io/LIMES/user_manual/

XML Configuration File

Metadata/header

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE LIMES SYSTEM "limes.dtd">
<LIMES>
```

Prefixes

```
  <PREFIX>
      <NAMESPACE>http://www.w3.org/1999/02/22-rdf-syntax-ns#</NAMESPACE>
      <LABEL>rdf</LABEL>
  </PREFIX>
```

XML Configuration File

Data Sources - Source

```
<SOURCE>
    <ID>dbpedia</ID>
    <ENDPOINT>http://dbpedia.org/sparql</ENDPOINT>
    <VAR>?y</VAR>
    <PAGESIZE>5000</PAGESIZE>
    <RESTRICTION>?y rdf:type dbo:City</RESTRICTION>
    <PROPERTY>rdfs:label</PROPERTY>
    <TYPE>sparql</TYPE>
</SOURCE>
```

XML Configuration File

Data Sources - Target

```
<TARGET>
    <ID>graphdb</ID>

<ENDPOINT>http://linkedgeodata.org/sparql</ENDPOINT>
    <VAR>?x</VAR>
    <PAGESIZE>5000</PAGESIZE>
    <RESTRICTION>?x rdf:type lgd:City</RESTRICTION>
    <PROPERTY>rdfs:label</PROPERTY>
</TARGET>
```

XML Configuration File

Metric

```
<METRIC>
    levenshtein(x.rdfs:label, y.rdfs:label)
</METRIC>
```

http://dice-group.github.io/LIMES/#/user_manual/configuration_file/defining_link_specifications?id=string-measures

Other measures to try!

XML Configuration File

Acceptance & Review Conditions

```
<ACCEPTANCE>
    <THRESHOLD>0.95</THRESHOLD>
    <FILE>accepted.nt</FILE>
    <RELATION>owl:sameAs</RELATION>
</ACCEPTANCE>

<REVIEW>
    <THRESHOLD>0.60</THRESHOLD>
    <FILE>reviewme.nt</FILE>
    <RELATION>owl:sameAs</RELATION>
</REVIEW>
```

XML Configuration File

Output Format

```
<OUTPUT>N3</OUTPUT>
```

End file

```
</LIMES>
```

Execute LIMES

RUN this command (it might take a minute or so since the KGs are large):

```
java -jar path/to/limes-core-${version}.jar path/to/{configuration-file}.xml
```

Remember to delete your output N-triples (.nt) file each time you want to re-run the LIMES tool with different parameters for experimentation

Output

```
<http://dbpedia.org/resource/Amsterdam>
<http://www.w3.org/2002/07/owl#sameAs>
<http://linkedgeodata.org/triplify/node268396336>
.

<http://dbpedia.org/resource/Berlin>
<http://www.w3.org/2002/07/owl#sameAs>
<http://linkedgeodata.org/triplify/node240109189>
.
```

Review the results

- What matches did you get?
- Are they all accurate?
- What happens if you experiment with the threshold parameter?

Questions?

More detailed instructions at
https://github.com/MaastrichtU-IDS/UM_KEN4256_KnowledgeGraphs

Resources

- Linked Data Book http://linkeddatabook.com/editions/1.0/
- When owl:sameAs isn’t the Same: An Analysis of Identity Links on the Semantic Web https://www.w3.org/2009/12/rdf-ws/papers/ws21
- DBpedia interlinks: https://wiki.dbpedia.org/services-resources/interlinking
- How to publish Linked Data on the Web: http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/