COMP318
Ontologies and Semantic Web

RDF - Part 3

Interoperability
Services
Systems
Semantic Web
SPARQL
Ontology Engineering
Information
Interaction
Ontology
RDF
Ontology
Linked Open Data
Semantic Web
Services
Ontology Alignment
Ontology
Agents
Knowledge
OWL
Knowledge Graph

Dr Valentina Tamma
V.Tamma@liverpool.ac.uk

How the Word Cloud Generator Works

The layout algorithm for positioning words without overlap is available on GitHub under an open source license as d3-cloud. Note that this is the the only the layout algorithm and any code for converting text into words and rendering the final output requires additional development.

As word placement can be quite slow for more than a few hundred words, the layout algorithm can be run asynchronously, with a configurable time step size. This makes it possible to animate words as they are placed without stuttering. It is recommended to always use a time step even without animations as it prevents the browser's event loop from blocking while placing the words.

The layout algorithm itself is incredibly simple. For each word, starting with the most "important":

Recap

- Limitation of XML

- RDF
  - Basic ideas behind RDF
  - Statements about resources
    - triples
    - graphs
  - Reification and higher order statements

RDF Serialisation formats

- RDF has been given a syntax in XML

  - This syntax inherits the benefits of XML

  - Other serialisations of RDF possible:

    - Notation 3 (N3)

      - Syntax for RDF

      - Logical language for RDF

    - N-Quads

      - Superset of N-triples for serialising multiple RDF graphs

    - Turtle

      - Refinement of N3

      - Just RDF representation

    - JSON-LD

      - JSON based serialisation

Why serialisation?

- RDF is not a data format, but a **data model** for describing resources in the forms of triples

- One of the use cases for RDF's design is to describe (*mark up*) the content of HTML web pages.

  - Either embedded in the HTML code

- RDFa is a syntactic variant introduced to help with that use case, and embeds RDF within the attributes of HTML tags

- or published in a separate file, typically in RDF/XML

  - conforming to the syntactic constraints deriving from the chosen syntax

  - and are processed:

    - in advance if dealing with static data set

    - on demand if dealing with dynamic data sets

RDF/XML syntax

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:dbpedia-owl="http://dbpedia.org/ontology/"

    xmlns:dbpedia="http://dbpedia.org/resource/"

    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"

    xmlns:swp= "http://www.swpExample.org/ontology/flats.ttl#" >

<rdf:Description rdf:about="http://www.swpExample.org/ontology/flats.ttl#BaronWayApartment">

    <swp:hasNumberOfBedrooms rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"> 3

    </swp:hasNumberOfBedrooms>

</rdf:Description>

<rdf:Description rdf:about="http://www.swpExample.org/ontology/flats.ttl#BaronWayFlat">

    <swp:isPartOf rdf:resource="http://www.swpExample.org/ontology/flats.ttl#BaronWayBuilding"/>

</rdf:Description>

<rdf:Description rdf:about="http://www.swpExample.org/ontology/flats.ttl#BaronWayBuilding">

    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Amsterdam"/>

    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Netherlands"/>

</rdf:Description>

</rdf:RDF>
```

RDF/XML syntax

- XML-based Syntax of RDF

  - All RDF/XML should be enclosed in an element `rdf:RDF`

  - Namespaces are used through `xmlns:` the namespace construct

  - The `MIME` type for the document

    - indicates the Internet media type and indicates the format of files on the Internet

    - for RDF/XML documents is `application/rdf+xml`

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:dbpedia-owl="http://dbpedia.org/ontology/"
    xmlns:dbpedia="http://dbpedia.org/resource/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:swp= "http://www.swpExample.org/ontology/flats.ttl#" >
<rdf:Description rdf:about="http://www.swpExample.org/ontology/
flats.ttl#BaronWayApartment">
    <swp:hasNumberOfBedrooms rdf:datatype="http://www.w3.org/2001/
XMLSchema#integer"> 3
    </swp:hasNumberOfBedrooms>
</rdf:Description>
<rdf:Description rdf:about="http://www.swpExample.org/ontology/
flats.ttl#BaronWayFlat">
    <swp:isPartOf rdf:resource="http://www.swpExample.org/ontology/
flats.ttl#BaronWayBuilding"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.swpExample.org/ontology/
flats.ttl#BaronWayBuilding">
    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Amsterdam"/>
    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Netherlands"/>
</rdf:Description>
</rdf:RDF>
```

RDF/XML syntax

- XML-based Syntax of RDF

  - Subjects are denoted within `rdf:Description` through `rdf:about,` that refers to the resource URI

  - Predicate and objects related to the subject are enclosed in the `rdf:Description` element

    - grouping by subject, in a similar way to Turtle

```xml
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:dbpedia-owl="http://dbpedia.org/ontology/"
    xmlns:dbpedia="http://dbpedia.org/resource/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:swp= "http://www.swpExample.org/ontology/flats.ttl#" >
<rdf:Description rdf:about="http://www.swpExample.org/ontology/
flats.ttl#BaronWayApartment">
    <swp:hasNumberOfBedrooms rdf:datatype="http://www.w3.org/2001/
XMLSchema#integer"> 3
    </swp:hasNumberOfBedrooms>
</rdf:Description>
<rdf:Description rdf:about="http://www.swpExample.org/ontology/
flats.ttl#BaronWayFlat">
    <swp:isPartOf rdf:resource="http://www.swpExample.org/ontology/
flats.ttl#BaronWayBuilding"/>
</rdf:Description>
<rdf:Description rdf:about="http://www.swpExample.org/ontology/
flats.ttl#BaronWayBuilding">
    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Amsterdam"/>
    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Netherlands"/>
</rdf:Description>
</rdf:RDF>
```

rdf:about vs rdf:ID

- The `rdf:Description` element makes a statement about a resource, identified in 3 ways:

  - `rdf:about` - this attribute references an existing resource, "defined" elsewhere

  - `rdf:ID` - this attribute is used to create a new resource

  - without a name, creating an anonymous resource

- Formally, there is no such thing as "defining" an object in one place and referring to it elsewhere

  - But, sometimes is useful (for human readability) to have a defining location, while other locations state "additional" properties

Property Elements

- The content of `rdf:Description`:

```
<rdf:Description rdf:about="http://www.swpExample.org/ontology/flats.ttl#BaronWayBuilding">
    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Amsterdam"/>
    <dbpedia-owl:country rdf:resource="http://dbpedia.org/resource/Netherlands"/>
</rdf:Description>
```

- **`dbpedia-owl:location`** and **`dbpedia-owl:country`** define two property-value pairs for the resource BaronWay Building

  - two RDF statements (triples) to be read conjunctively

rdf:resource

- The relationship between the resource *Baron Way Building* and the resource *Amsterdam* exists through referencing a resource with the given name

  - But how do we know which Amsterdam we refer to?

    - Is it Amsterdam capital of the Netherlands?

    - Is it Amsterdam, NY, USA?
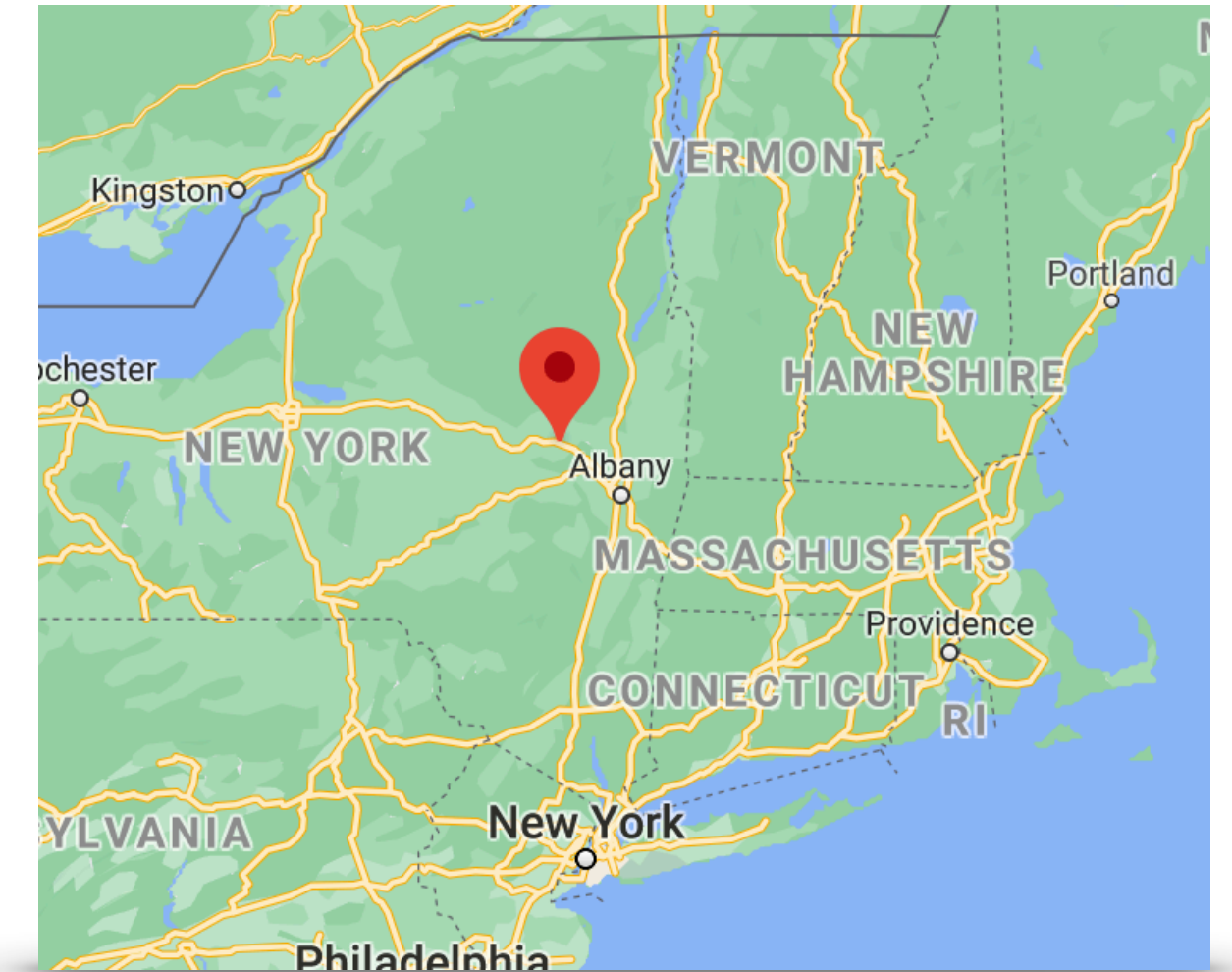
rdf:ID

- A description with an `rdf:ID` defines a fragment URI

  - can reference the defined description

- The use of the same name can be a coincidence

  - possible if you do not assume that all data is asserted locally

  - Some resources are declared externally

```
<?xml version="1.0" encoding="utf-8"?>
<rdf:RDF xmlns:swp = "http://www.swpExample.org/ontology/flats.ttl#"
    ....>
  <rdf:Description rdf:about="http://www.swpExample.org/ontology/flats.ttl#BaronWayBuilding">
    <dbpedia-owl:location rdf:resource="http://dbpedia.org/resource/Amsterdam"/>
    <dbpedia-owl:country rdf:resource="http://dbpedia.org/resource/Netherlands"/>
    <swp:tenant rdf:resource = flats.ttl#jeff>
</rdf:Description>
<rdf:Description rdf:ID="flats.ttl#jeff">
    <swp:rents rdf:resource = flats.ttl#BaronWayBuilding>
</rdf:Description>
```

More on URIs

- **URI = Uniform Resource Identifier**
  - allow for denoting resources in a general, unambiguous way

- **Resource: any object that possesses a clear identity (within the context of a given application)**
  - books, cities, humans, publishers, but also relations between those, abstract concepts, etc.
  - already realised in some domains:
    - ISBN for books

- URIs do not need to correspond to an actual location

- but it is good practice if they do
  - a picture, a FOAF description, a map…

- Builds on concept of URLs but…
  - not every URI refers to a Web document
  - however, often the URL of a document is used as its URI

- URI starts with so-called URI schema separated from the following part by ":"
  - e.g, http, ftp, mailto
    - but starting with http does not necessarily mean http-accessible…
  - mostly hierarchically organised

Self-defined URIs?

- Necessary if no URI exists (yet) for a resource
  - or it is not known

- strategy for avoiding unwanted clashes:
  - use http URIs of webspace you control

- AND provide some documentation about the URI

- Need to distinguish the URI of a resource from URI of the associated documents describing it:
  - Example: URI for "Lord of the Ring"

```
http://www.wikipedia.org/wiki/LordOfTheRing#URI
```

```
http://www.wikipedia.org/wiki/LordOfTheRing
```

Data types

- The attribute `rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"`

```
<rdf:Description rdf:about="http://www.swpExample.org/ontology/flats.ttl#BaronWayApartment">
    <swp:hasNumberOfBedrooms
         rdf:datatype="http://www.w3.org/2001/XMLSchema#integer"> 3
    </swp:hasNumberOfBedrooms>
</rdf:Description>
```

- `swp:hasNumberOfBedrooms` defines the number of bedrooms property and has `http://www.w3.org/2001/XMLSchema#integer` as its range

  - It is required to indicate the type of the property every time it is used, to ensure that an RDF processor can assign the correct type of value even when the schema is not known

COMP318
Ontologies and Semantic Web

End of RDF - Part 3

Dr Valentina Tamma

`V.Tamma@liverpool.ac.uk`

How the Word Cloud Generator Works

The layout algorithm for positioning words without overlap is available on GitHub under an open source license as d3-cloud. Note that this is the the only the layout algorithm and any code for converting text into words and rendering the final output requires additional development.

As word placement can be quite slow for more than a few hundred words, the layout algorithm can be run asynchronously, with a configurable time step size. This makes it possible to animate words as they are placed without stuttering. It is recommended to always use a time step even without animations as it prevents the browser's event loop from blocking while placing the words.

The layout algorithm itself is incredibly simple. For each word, starting with the most "important":