Chapter 3
Deploying Linked Open Data
Methodologies and Software Tools

NIKOLAOS KONSTANTINOU

DIMITRIOS-EMMANUEL SPANOS

Outline

Introduction

Modeling Data

Software for Working with Linked Data

Software Tools for Storing and Processing Linked Data

Tools for Linking and Aligning Linked Data

Software Libraries for working with RDF

Introduction

Today's Web: Anyone can say anything about any topic

- Information on the Web cannot always be trusted

Linked Open Data (LOD) approach

- Materializes the Semantic Web vision
- A focal point is provided for any given web resource
  - Referencing (referring to)
  - De-referencing (retrieving data about)

Not All Data Can Be Published Online

Data has to be

- Stand-alone
  - Strictly separated from business logic, formatting, presentation processing
- Adequately described
  - Use well-known vocabularies to describe it, or
  - Provide de-referenceable URIs with vocabulary term definitions
- Linked to other datasets
- Accessed simply
  - HTTP and RDF instead of Web APIs

Linked Data-driven Applications (1)

Content reuse

- E.g. BBC's Music Store
  - Uses DBpedia and MusicBrainz

Semantic tagging and rating

- E.g. Faviki
  - Uses DBpedia

Linked Data-driven Applications (2)

Integrated question-answering

- E.g. DBpedia mobile
  - Indicate locations in the user's vicinity

Event data management

- E.g. Virtuoso's calendar module
  - Can organize events, tasks, and notes

Linked Data-driven Applications (3)

Linked Data-driven data webs are expected to evolve in numerous domains

- E.g. Biology, software engineering

The bulk of Linked Data processing is not done online

Traditional applications use other technologies

- E.g. relational databases, spreadsheets, XML files
- Data must be transformed in order to be published on the web

The O in LOD: Open Data

Open ≠ Linked

- Open data is data that is publicly accessible via internet
  - No physical or virtual barriers to accessing them
- Linked Data allows relationships to be expressed among these data

RDF is ideal for representing Linked Data

- This contributes to the misconception that LOD can only be published in RDF

Definition of openness by www.opendefinition.org

> Open means anyone can freely access, use, modify, and share for any purpose (subject, at most, to requirements that preserve provenance and openness)

Why should anyone open their data?

Reluctance by data owners

- Fear of becoming useless by giving away their core value

In practice the opposite happens

- Allowing access to content leverages its value
  - Added-value services and products by third parties and interested audience
  - Discovery of mistakes and inconsistencies
    - People can verify convent freshness, completeness, accuracy, integrity, overall value
  - In specific domains, data have to be open for strategic reasons
    - E.g. transparency in government data

Steps in Publishing Linked Open Data (1)

Data should be kept simple

- Start small and fast
- Not all data is required to be opened at once
- Start by opening up just one dataset, or part of a larger dataset
- Open up more datasets
  - Experience and momentum may be gained
  - Risk of unnecessary spending of resources
    - Not every dataset is useful

Steps in Publishing Linked Open Data (2)

Engage early and engage often

- Know your audience
- Take its feedback into account
- Ensure that next iteration of the service will be as relevant as it can be
- End users will not always be direct consumers of the data
  - It is likely that intermediaries will come between data providers and end users
    - E.g. an end user will not find use in an array of geographical coordinates but a company offering maps will
  - Engage with the intermediaries
    - They will reuse and repurpose the data

Steps in Publishing Linked Open Data (3)

Deal in advance with common fears and misunderstandings

- Opening data is not always looked upon favorably
- Especially in large institutions, it will entail a series of consequences and, respectively, opposition
- Identify, explain, and deal with the most important fears and probable misconceptions from an early stage

Steps in Publishing Linked Open Data (4)

It is fine to charge for access to the data via an API

- As long as the data itself is provided in bulk for free
- Data can be considered as open
  - The API is considered as an added-value service on top of the data
  - Fees are charged for the use of the API, not of the data
- This opens business opportunities in the data-value chain around open data

Steps in Publishing Linked Open Data (5)

Data openness ≠ data freshness

- Opened data does not have to be a real-time snapshot of the system data
  - Consolidate data into bulks asynchronously
  - E.g. every hour or every day
  - You could offer bulk access to the data dump and access through an API to the real-time data

Dataset Metadata (1)

Provenance

- Information about entities, activities and people involved in the creation of a dataset, a piece of software, a tangible object, a thing in general
- Can be used in order to assess the thing's quality, reliability, trustworthiness, etc.
- Two related recommendations by W3C
  - The PROV Data Model, in OWL 2
  - The PROV ontology

Dataset Metadata (2)

Description about the dataset

W3C recommendation

- DCAT
  - Describes an RDF vocabulary
  - Specifically designed to facilitate interoperability between data catalogs published on the Web

Dataset Metadata (3)

Licensing

- A short description regarding the terms of use of the dataset
- E.g. for the Open Data Commons Attribution License

> This {DATA(BASE)-NAME} is made available under the Open Data Commons Attribution License: http://opendatacommons.org/licenses/by/{version }.—See more at: http://opendatacommons.org/licenses/by/#sthash.9HadQzSW.dpuf

Bulk Access vs. API (1)

Offering bulk access is a requirement

- Offering an API is not

Bulk Access vs. API (2)

Bulk access

- Can be cheaper than providing an API
  - Even an elementary API entails development and maintenance costs
- Allows building an API on top of the offered data
  - Offering an API does not allow clients to retrieve the whole amount of data
- Guarantees full access to the data
  - An API does not

Bulk Access vs. API (3)

API

- More suitable for large volumes of data
  - No need to download the whole dataset when a small subset is needed

The 5-Star Deployment Scheme

★	Data is made available on the Web (whatever format) but with an open license to be Open Data
★★	Available as machine-readable structured data: e.g. an Excel spreadsheet instead of image scan of a table
★★★	As the 2-star approach, in a non-proprietary format: e.g. CSV instead of Excel
★★★★	All the above plus the use of open standards from W3C (RDF and SPARQL) to identify things, so that people can point at your stuff
★★★★★	All the above, plus: Links from the data to other people's data in order to provide context

Outline

Introduction

Modeling Data

Software Tools for Storing and Processing Linked Data

Tools for Linking and Aligning Linked Data

Software Libraries for working with RDF

The D in LOD: Modeling Content

Content has to comply with a specific model

- A model can be used
  - As a mediator among multiple viewpoints
  - As an interfacing mechanism between humans or computers
  - To offer analytics and predictions
- Expressed in RDF(S), OWL
- Custom or reusing existing vocabularies
- Decide on the ontology that will serve as a model
  - Among the first decisions when publishing a dataset as LOD
- Complexity of the model has to be taken into account, based on the desired properties
  - Decide whether RDFS or one of the OWL profiles (flavors) is needed

Reusing Existing Works (1)

Vocabularies and ontologies have existed long before the emergence of the Web

- Widespread vocabularies and ontologies in several domains encode the accumulated knowledge and experience

Highly probable that a vocabulary has already been created in order to describe the involved concepts

- Any domain of interest

Reusing Existing Works (2)

Increased interoperability

- Use of standards can help content aggregators to parse and process the information
  - Without much extra effort per data source
- E.g. an aggregator that parses and processes dates from several sources
  - More likely to support the standard date formats
  - Less likely to convert the formatting from each source to a uniform syntax
    - Much extra effort per data source
  - E.g. DCMI Metadata Terms
    - Field dcterms:created, value "2014-11-07"^^xsd:date

Reusing Existing Works (3)

Credibility

- Shows that the published dataset
  - Has been well thought of
  - Is curated
- A state-of-the-art survey has been performed prior to publishing the data

Ease of use

- Reusing is easier than rethinking and implementing again or replicating existing solutions
  - Even more, when vocabularies are published by multidisciplinary consortia with potentially more spherical view on the domain than yours

Reusing Existing Works (4)

In conclusion

- Before adding terms in our vocabulary, make sure they do not already exist
  - In such case, reuse them by reference
- When we need to be more specific, we can create a subclass or a subproperty of the existing
- New terms can be generated, when the existing ones do not suffice

Semantic Web for Content Modeling

Powerful means for system description

- Concept hierarchy, property hierarchy, set of individuals, etc.

Beyond description

- Model checking
  - Use of a reasoner assures creation of coherent, consistent models
- Semantic interoperability
- Inference
- Formally defined semantics
- Support of rules
- Support of logic programming

Assigning URIs to Entities

Descriptions can be provided for

- Things that exist online
- Items/persons/ideas/things (in general) that exist outside of the Web

Example: Two URIs to describe a company

- The company's website
- A description of the company itself
  - May well be in an RDF document

A strategy has to be devised in assigning URIs to entities

- No deterministic approaches

Assigning URIs to Entities: Challenges

Dealing with ungrounded data

Lack of reconciliation options

Lack of identifier scheme documentation

Proprietary identifier schemes

Multiple identifiers for the same concepts/entities

Inability to resolve identifiers

Fragile identifiers

Assigning URIs to Entities: Benefits

Semantic annotation

Data is discoverable and citable

The value of the data increases as the usage of its identifiers increases

URI Design Patterns (1)

Conventions for how URIs will be assigned to resources

Also widely used in modern web frameworks

In general applicable to web applications

Can be combined

Can evolve and be extended over time

Their use is not restrictive

- Each dataset has its own characteristics

Some upfront thought about identifiers is always beneficial

URI Design Patterns (2)

Hierarchical URIs

- URIs assigned to a group of resources that form a natural hierarchy
  - E.g. :collection/:item/:sub-collection/:item

Natural keys

- URIs created from data that already has unique identifiers
  - E.g. identify books using their ISBN

URI Design Patterns (3)

Literal keys

- URIs created from existing, non-global identifiers
  - E.g. the dc:identifier property of the described resource

Patterned URIs

- More predictable, human-readable URIs
- E.g. /books/12345
  - /books is the base part of the URI indicating “the collection of books”
  - 12345 is an identifier for an individual book

URI Design Patterns (4)

Proxy URIs

- Used in order to deal with the lack of standard identifiers for third-party resources
- If for these resources, identifiers do exist, then these should be reused
  - If not, use locally minted Proxy URIs

Rebased URIs

- URIs constructed based on other URIs
  - E.g. URIs rewritten using regular expressions
  - From http://graph1.example.org/document/1 to http://graph2.example.org/document/1

URI Design Patterns (5)

Shared keys

- URIs specifically designed to simplify the linking task between datasets
- Achieved by a creating *Patterned URIs* while applying the *Natural Keys* pattern
- Public, standard identifiers are preferable to internal, system-specific

URL slugs

- URIs created from arbitrary text or keywords, following a certain algorithm
- E.g. lowercasing the text, removing special characters and replacing spaces with a dash
  - A URI for the name “Brad Pitt” could be http://www.example.org/brad-pitt

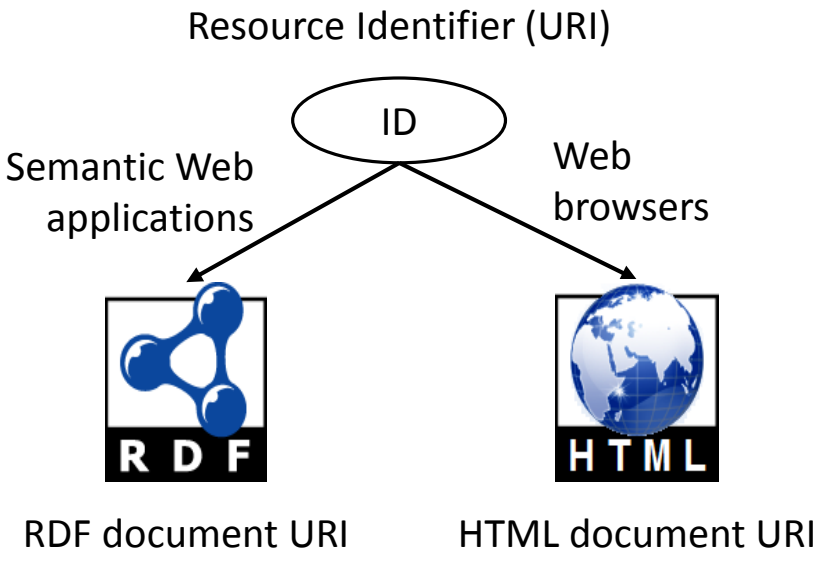
Assigning URIs to Entities

Desired functionality

- Semantic Web applications retrieve the RDF description of things
- Web browsers are directed to the (HTML) documents describing the same resource

Two categories of technical approaches for providing URIs for dataset entities

- Hash URIs
- 303 URIs



Hash URIs

URIs contain a fragment separated from the rest of the URI using '#'

- E.g. URIs for the descriptions of two companies
  - http://www.example.org/info#alpha
  - http://www.example.org/info#beta
- The RDF document containing descriptions about both companies
  - http://www.example.org/info
- The original URIs will be used in this RDF document to uniquely identify the resources
  - Companies Alpha, Beta and anything else

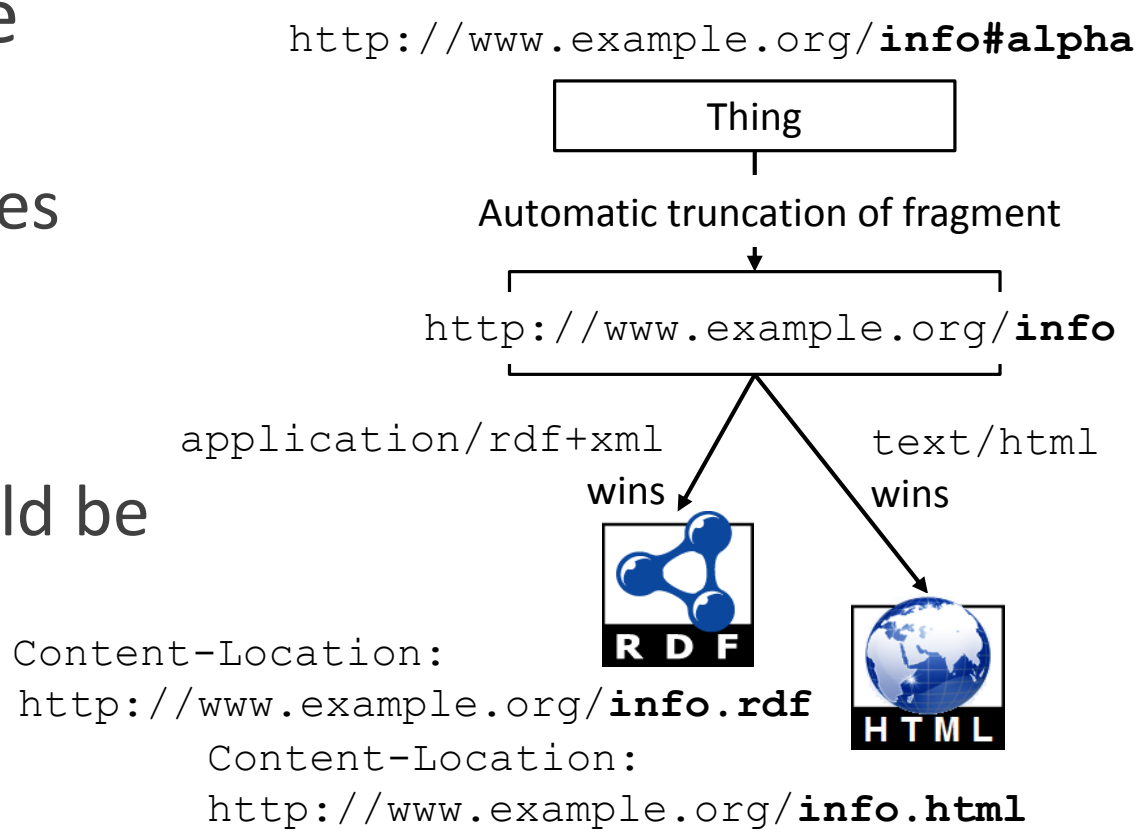
Hash URIs with Content Negotiation

Redirect either to the RDF or the HTML representation

- Decision based on client preferences and server configuration

Technically

- The Content-Location header should be set to indicate where the hash URI refers to
  - Part of the RDF document (info.rdf)
  - Part of the HTML document (info.html)



Hash URIs without Content Negotiation

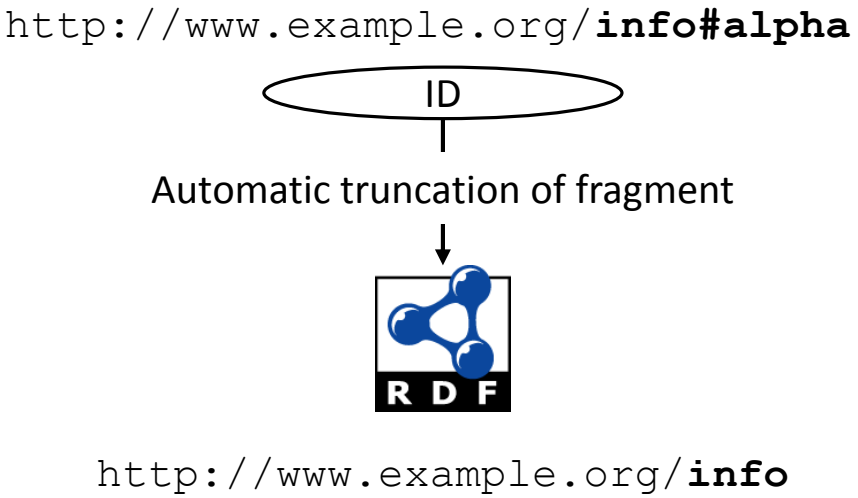
Can be implemented by simply uploading static RDF files to a Web server

- No special server configuration is needed
- Not as technically challenging as the previous one

Popular for quick-and-dirty RDF publication

Major problem: clients will be obliged to load (download) the whole RDF file

- Even if they are interested in only one of the resources



303 URIs (1)

Approach relies on the “303 See Other” HTTP status code

- Indication that the requested resource is not a regular Web document

Regular HTTP response (200) cannot be returned

- Requested resource does not have a suitable representation

However, we still can retrieve description about this resource

- Distinguishing between the real-world resource and its description (representation) on the Web

303 URIs (2)

HTTP 303 is a redirect status code

- Server provides the location of a document that represents the resource

E.g. companies Alpha and Beta can be described using the following URIs

- http://www.example.org/id/alpha
- http://www.example.org/id/beta

Server can be configured to answer requests to these URIs with a 303 (redirect) HTTP status code

303 URIs (3)

Location can contain an HTML, an RDF, or any alternative form, e.g.

- http://www.example.org/doc/alpha
- http://www.example.org/doc/beta

This setup allows to maintain bookmarkable, de-referenceable URIs

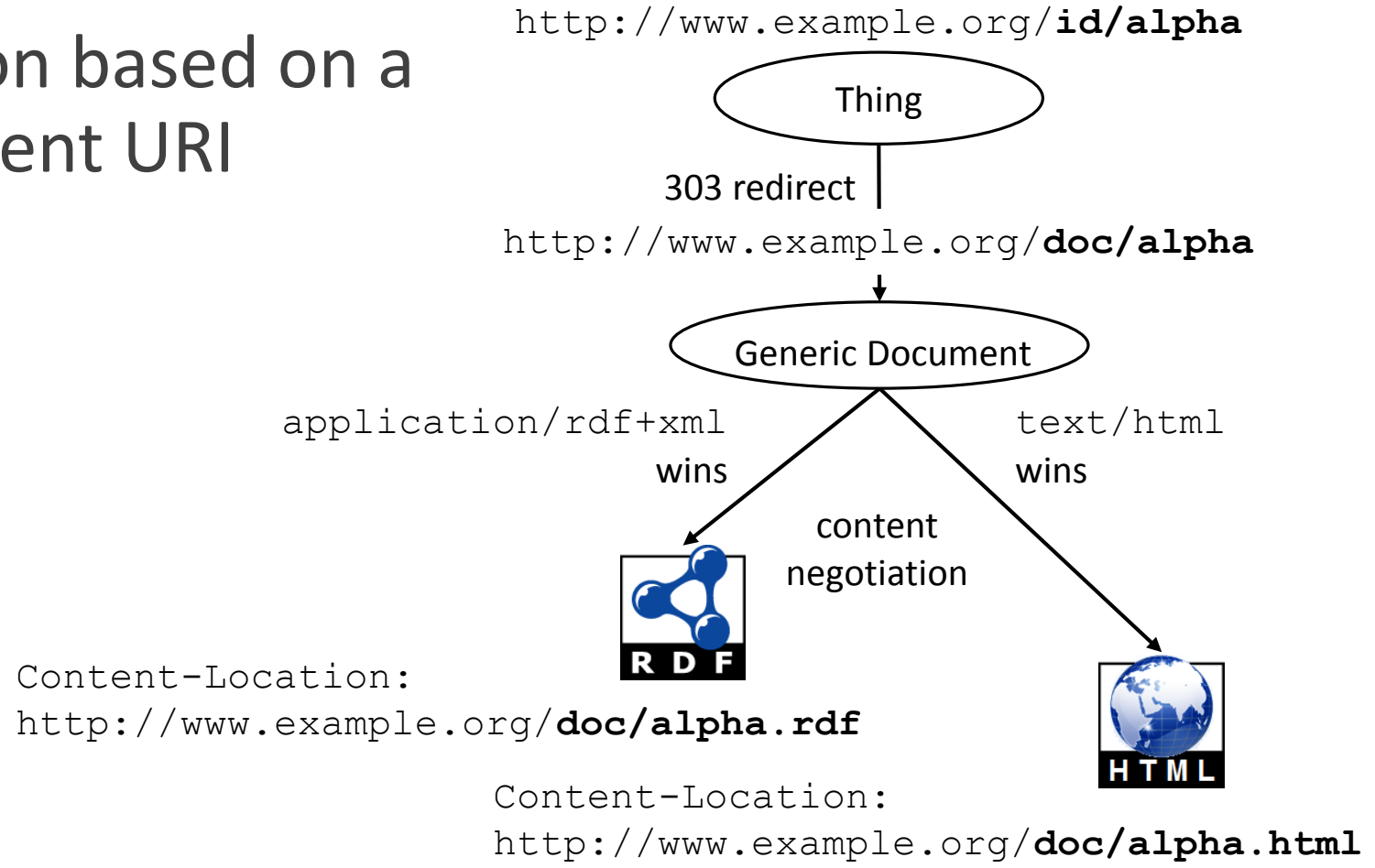
- For both the RDF and HTML views of the same resource

A very flexible approach

- Redirection target can be configured separately per resource
- There could be a document for each resource, or one (large) document with descriptions of all the resources

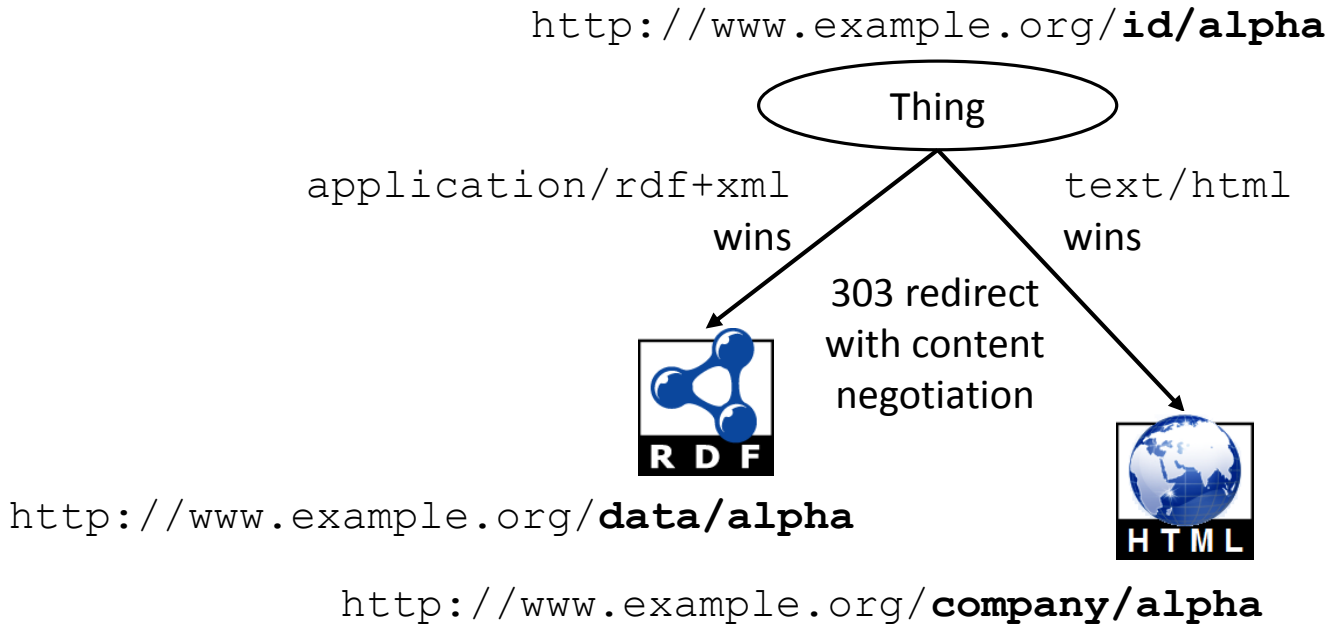
303 URIs (4)

303 URI solution based on a generic document URI



303 URIs (5)

303 URI solution without the generic document URI



303 URIs (6)

Problems of the 303 approach

- Latency caused by client redirects
- A client looking up a set of terms may use many HTTP requests
  - While everything that could be loaded in the first request is there and is ready to be downloaded
- Clients of large datasets may be tempted to download the full data via HTTP, using many requests
  - In these cases, SPARQL endpoints or comparable services should be provided in order to answer complex queries directly on the server

303 URIs (7)

303 and Hash approaches are not mutually exclusive

- Contrarily, combining them could be ideal
  - Allow large datasets to be separated into multiple parts and have identifiers for non-document resources

Outline

Introduction

Modeling Data

Software for Working with Linked Data

Software Tools for Storing and Processing Linked Data

Tools for Linking and Aligning Linked Data

Software Libraries for working with RDF

Software for Working with Linked Data (1)

Working on small datasets is a task that can be tackled by manually authoring an ontology, however

Publishing LOD means the data has to be programmatically manipulated

Many tools exist that facilitate the effort

Software for Working with Linked Data (2)

The most prominent tools listed next

- Ontology authoring environments
- Cleaning data
- Software tools and libraries for working with Linked Data
- No clear lines can be drawn among software categories
  - E.g. graphical tools offering programmatic access, or software libraries offering a GUI

Ontology Authoring (1)

Not a linear process

- It is not possible to line up the steps needed in order to complete its authoring

An iterative procedure

- The core ontology structure can be enriched with more specialized, peripheral concepts
  - Further complicating concept relations
  - The more the ontology authoring effort advances, the more complicated the ontology becomes

Ontology Authoring (2)

Various approaches

- Start from the more general and continue with the more specific concepts
- Reversely, write down the more specific concepts and group them

Can uncover existing problems

- E.g. concept clarification
- Understanding of the domain in order to create its model
- Probable reuse and connect to other ontologies

Ontology Editors (1)

Offer a graphical interface

- Through which the user can interact
- Textual representation of ontologies can be prohibitively obscure

Assure syntactic validity of the ontology

Consistency checks

Ontology Editors (2)

Freedom to define concepts and their relations

- Constrained to assure semantic consistency

Allow revisions

Several ontology editors have been built

- Only a few are practically used
- Among them the ones presented next

Protégé (1)

Open-source

Maintained by the Stanford Center for Biomedical Informatics Research

Among the most long-lived, complete and capable solutions for ontology authoring and managing

A rich set of plugins and capabilities

Protégé (2)

Customizable user interface

Multiple ontologies can be developed in a single frame workspace

Several Protégé frames roughly correspond to OWL components

- Classes
- Properties
  - Object Properties, Data Properties, and Annotation Properties
- Individuals

A set of tools for visualization, querying, and refactoring

Protégé (3)

Reasoning support
- Connection to DL reasoners like HermiT (included) or Pellet

OWL 2 support

Allows SPARQL queries

WebProtégé
- A much younger offspring of the Desktop version
- Allows collaborative viewing and editing
- Less feature-rich, more buggy user interface

TopBraid Composer

RDF and OWL authoring and editing environment

Based on the Eclipse development platform

A series of adapters for the conversion of data to RDF

- E.g. from XML, spreadsheets and relational databases

Supports persistence of RDF graphs in external triple stores

Ability to define SPIN rules and constraints and associate them with OWL classes

Maestro, Commercial, Free edition

- Free edition offers merely a graphical interface for the definition of RDF graphs and OWL ontologies and the execution of SPARQL queries on them

The NeOn Toolkit

Open-source ontology authoring environment

- Also based on the Eclipse platform

Mainly implemented in the course of the EC-funded NeOn project

- Main goal: the support for all tasks in the ontology engineering life-cycle

Contains a number of plugins

- Multi-user collaborative ontology development
- Ontology evolution through time
- Ontology annotation
- Querying and reasoning
- Mappings between relational databases and ontologies
  - ODEMapster plugin

Platforms and Environments

Data that published as Linked Data is not always produced primarily in this form

- Files in hard drives, relational databases, legacy systems etc.

Many options regarding how the information is to be transformed into RDF

Many software tools and libraries available in the Linked Data ecosystem

- E.g. for converting, cleaning up, storing, visualizing, linking etc.
- Creating Linked Data from relational databases is a special case, discussed in detail in the next Chapter

Cleaning-Up Data: OpenRefine (1)

Data quality may be lower than expected

- In terms of homogeneity, completeness, validity, consistency, etc.

Prior processing has to take place before publishing

It is not enough to provide data as Linked Data

- Published data must meet certain quality standards

Cleaning-Up Data: OpenRefine (2)

Initially developed as “Freebase Gridworks”, renamed “Google Refine” in 2010, “OpenRefine” after its transition to a community-supported project in 2012

Created specifically to help working with messy data

Used to improve data consistency and quality

Used in cases where the primary data source are files

- In tabular form (e.g. TSV, CSV, Excel spreadsheets) or
- Structured as XML, JSON, or even RDF.

Cleaning-Up Data: OpenRefine (3)

Allows importing data into the tool and connect them to other sources

It is a web application, intended to run locally, in order to allow processing sensitive data

Cleaning data

- Removing duplicate records
- Separating multiple values that may reside in the same field
- Splitting multi-valued fields
- Identifying errors (isolated or systematic)
- Applying ad-hoc transformations using regular expressions

OpenRefine: The RDF Refine Extension (1)

Allows conversion from other sources to RDF

- RDF export
- RDF reconciliation

RDF export part

- Describe the shape of the generated RDF graph through a template
- Template uses values from the input spreadsheet
- User can specify the structure of an RDF graph
  - The relationships that hold among resources
  - The form of the URI scheme that will be followed, etc.

OpenRefine: The RDF Refine Extension (2)

RDF reconciliation part

- Offers a series of alternatives for discovering related Linked Data entities
- A reconciliation service
- Allows reconciliation of resources
  - Against an arbitrary SPARQL endpoint
    - With or without full-text search functionality
    - A predefined SPARQL query that contains the request label (i.e. the label of the resource to be reconciled) is sent to a specific SPARQL endpoint
  - Via the Sindice API
    - A call to the Sindice API is directly made using the request label as input to the service

Outline

Introduction

Modeling Data

Software Tools for Storing and Processing Linked Data

Tools for Linking and Aligning Linked Data

Software Libraries for working with RDF

Tools for Storing and Processing Linked Data

Storing and processing solutions

- Usage not restricted to these capabilities

A mature ecosystem of technologies and solutions

- Cover practical problems such as programmatic access, storage, visualization, querying via SPARQL endpoints, etc.

Sesame

An open source, fully extensible and configurable with respect to storage mechanisms, Java framework for processing RDF data

Transaction support

RDF 1.1 support

Storing and querying APIs

A RESTful HTTP interface supporting SPARQL

The Storage And Inference Layer (Sail) API

- A low level system API for RDF stores and inferences, allowing for various types of storage and inference to be used

OpenLink Virtuoso (1)

RDF data management and Linked Data server solution

- Also a web application/web services/relational database/file server

Offers a free and a commercial edition

Implements a quad store

- (graph, subject, predicate, object)

OpenLink Virtuoso (2)

Graphs can be

- Directly uploaded to Virtuoso
- Transient (not materialized) RDF views on top of its relational database backend
- Crawled from third party RDF (or non-RDF, using Sponger) sources

Offers several plugins

- Full-text search, faceted browsing, etc.

Apache Marmotta

The LDClient library

- A Linked Data Client
- A modular tool that can convert data from other formats into RDF
- Can be used by any Linked Data project, independent of the Apache Marmotta platform

Can retrieve resources from remote data sources and map their data to appropriate RDF structures

A number of different backends is included

- Provide access to online resources
  - E.g. Freebase, Facebook graph API, RDFa-augmented HTML pages

Callimachus

An open source platform

- Also available in an enterprise closed-source edition

Development of web applications based on RDF and Linked Data

A Linked Data Management System

- Creating, managing, navigating and visualizing Linked Data through appropriate front-end components

Relies on XHTML and RDFa templates

- Populated by the results of SPARQL queries executed against an RDF triple store
  - Constitute the human-readable web pages

Visualization software

Users may have limited or non-existent knowledge of Linked Data and the related ecosystem

LodLive

- Provides a navigator that uses RDF resources, relying on SPARQL endpoints

CubeViz

- A faceted browser for statistical data
- Relies on the RDF Data Cube vocabulary for representing statistical data in RDF

Gephi

GraphViz

} Open-source, generic graph visualization platforms

Apache Stanbol

A semantic content management system

Aims at extending traditional CMS's with semantic services

Reusable components

- Via a RESTful web service API that returns JSON, RDF and supports JSON-LD

Ontology manipulation

Content enhancement

- Semantic annotation

Reasoning

Persistence

Stardog

RDF database, geared towards scalability

Reasoning

OWL 2

SWRL

Implemented in Java

Exposes APIs for Jena and Sesame

Offers bindings for its HTTP protocol in numerous languages

- Javascript, .Net, Ruby, Clojure, Python

Commercial and free community edition

Outline

Introduction

Modeling Data

Software Tools for Storing and Processing Linked Data

Tools for Linking and Aligning Linked Data

Software Libraries for working with RDF

The L in LOD (1)

Web of Documents

- HTML Links
  - Navigate among (HTML) pages

Web of (Linked) Data

- RDF links
  - Navigate among (RDF) data
  - Relationships between Web resources
  - Triples (resource, property, resource)
  - Main difference from simple hyperlinks: they possess some meaning

The L in LOD (2)

Links to external datasets of the LOD cloud

Integration of the new dataset in the Web of data

Without links, all published RDF datasets would essentially be isolated islands in the “ocean” of Linked Data

The L in LOD (3)

Establishing links can be done

- Manually
  - I.e. the knowledge engineer identifies the most appropriate datasets and external resources
  - More suitable for small and static datasets
- Semi-automatically
  - Harness existing open-source tools developed for such a purpose
  - More suitable for large datasets

Silk (1)

An open-source framework for the discovery of links among RDF resources of different datasets

Available

- As a command line tool
- With a graphical user interface
- Cluster edition, for the discovery of links among large datasets

Silk (2)

Link specification language

- Details and criteria of the matching process, including
  - The source and target RDF datasets
  - The conditions that resources should fulfill in order to be interlinked
  - The RDF predicate that will be used for the link
  - The pre-matching transformation functions
  - Similarity metrics to be applied

LIMES

A link discovery framework among RDF datasets

Extracts instances and properties from both source and target datasets, stores them in a cache storage or memory and computes the actual matches

- Based on restrictions specified in a configuration file

Offers a web interface for authoring the configuration file

Sindice

A service that can be used for the manual discovery of related identifiers

An index of RDF datasets that have been crawled and/or extracted from semantically marked up Web pages

Offers both free-text search and SPARQL query execution functionalities

Exposes several APIs that enable the development of Linked Data applications that can exploit Sindice's crawled content

DBpedia Spotlight

Designed for annotating mentions of DBpedia resources in text

Provides an approach for linking information from unstructured sources to the LOD cloud through DBpedia

Tool architecture comprises:

- A web application
- A web service
- An annotation and an indexing API in Java/Scala
- An evaluation module

Sameas.org

An online service

Retrieves related LOD entities from some of the most popular datasets

Serves more than 150 million URIs

Provides a REST interface that retrieves related URIs for a given input URI or label

Accepts URIs as inputs from the user and returns URIs that may well be co-referent

Outline

Introduction

Modeling Data

Tools for Linking and Aligning Linked Data

Software Libraries for working with RDF

Jena (1)

Open-source Java API

Allows building of Semantic Web and Linked Data applications

The most popular Java framework for ontology manipulation

First developed and brought to maturity by HP Labs

Now developed and maintained by the Apache Software Foundation

First version in 2000

Comprises a set of tools for programmatic access and management of Semantic Web resources

Jena (2)

ARQ, a SPARQL implementation

Fuseki

- SPARQL Server, part of the Jena framework, that offers access to the data over HTTP using RESTful services. Fuseki can be downloaded and extracted locally, and run as a server offering a SPARQL endpoint plus some REST commands to update the dataset.

OWL support

- Coverage of the OWL language

Inference API

- Using an internal rule engine, Jena can be used as a reasoner

Jena TDB (1)

High-performance triple store solution

Based on a custom implementation of threaded B+ Trees

- Only provides for fixed length key and fixed length value
- No use of the value part in triple indexes

Efficient storage and querying of large volumes of graphs

- Performs faster, scales much more than a relational database backend

Stores the dataset in directory

Jena TDB (2)

A TDB instance consists of

- A Node table that stores the representation of RDF terms
- Triple and Quad indexes
  - Triples are used for the default graph, quads for the named graphs
- The Prefixes table
  - Provides support for Jena's prefix mappings and serves mainly presentation and serialization of triples issues, and does not take part in query processing

Apache Any23 (1)

A programming library

A web service

A command line tool

Ability to extract structured data in RDF from Web documents

Input formats

- RDF (RDF/XML, Turtle, Notation 3)
- RDFa, microformats (hCalendar, hCard, hListing, etc.)
- HTML 5 Microdata (such as schema.org)
- JSON-LD (Linked Data in JSON format)

Apache Any23 (2)

Support for content extraction following several vocabularies

- CSV
- Dublin Core Terms
- Description of a Career
- Description Of A Project
- Friend Of A Friend
- GeoNames
- ICAL
- Open Graph Protocol
- schema.org
- VCard, etc.

Redland

Support for programmatic management and storage of RDF graphs

A set of libraries written in C, including:

- Raptor
  - Provides a set of parsers and serializers of RDF
  - Including RDF/XML, Turtle, RDFa, N-Quads, etc.
- Rasqal
  - Supports SPARQL query processing of RDF graphs
- Redland RDF Library
  - Handles RDF manipulation and storage

Also allows function invocation through Perl, PHP, Ruby, Python, etc.

EasyRDF

A PHP library for the consumption and production of RDF

Offers parsers and serializers for most RDF serializations

Querying using SPARQL

Type mapping from RDF resources to PHP objects

RDFLib

A Python library for working with RDF

Serialization formats

Microformats

RDFa

OWL 2 RL

Using relational databases as a backend

Wrappers for remote SPARQL endpoints

The Ruby RDF Project

RDF Processing using the Ruby language

Reading/writing in different RDF formats

Microdata support

Querying using SPARQL

Using relational databases as a storage backend

Storage adaptors

- Sesame
- MongoDB

dotNetRDF (1)

Open-source library for RDF

Written in C#.Net, also offering ASP.NET integration

Developer API

dotNetRDF (2)

Supports

- SPARQL
- Reasoning
- Integration with third party triple stores
  - Jena, Sesame, Stardog, Virtuoso, etc.

Suite of command line and graphical tools for

- Conversions between RDF formats
- Running a SPARQL server and submitting queries
- Managing any supported triple stores