Toward Easier RDF

David Booth, PhD

US Semantic Technology Symposium

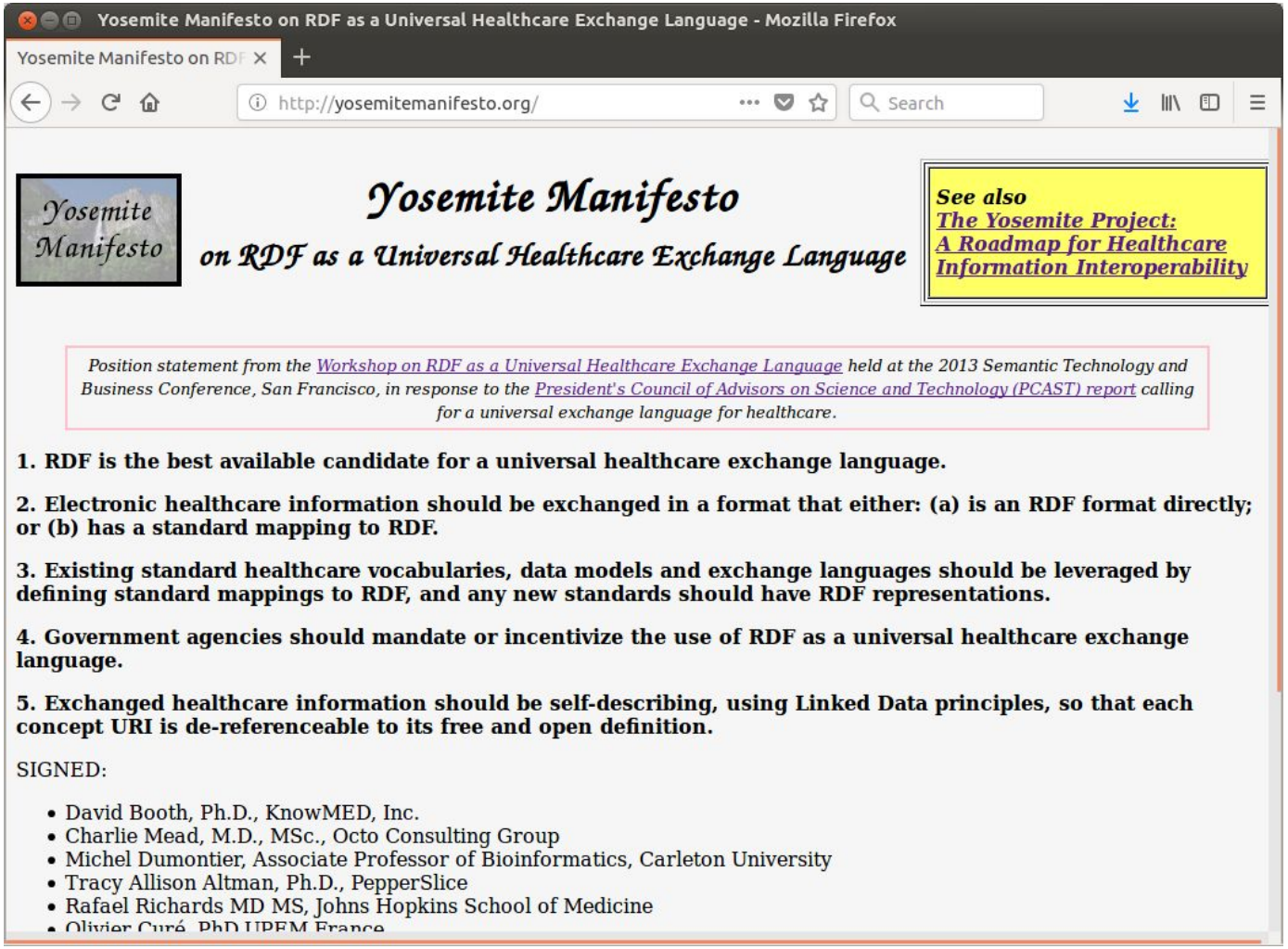
March 2018

Latest version of these slides: https://goo.gl/H2vBYi

http://YosemiteProject.org/

About the speaker: David Booth

- Co-founder of Yosemite Project
- Senior Software Architect
- In Semantic Web technology since 2002
- W3C Fellow 2002-2005
- Focus on healthcare data since 2009
- PhD in Computer Science from UCLA


Yosemite Manifesto on RDF as a Universal Healthcare Exchange Language - Mozilla Firefox
Yosemite Manifesto on RDF
http://yosemitemanifesto.org/
Search
Yosemite Manifesto
Yosemite Manifesto
on RDF as a Universal Healthcare Exchange Language
See also
The Yosemite Project: A Roadmap for Healthcare Information Interoperability
Position statement from the Workshop on RDF as a Universal Healthcare Exchange Language held at the 2013 Semantic Technology and Business Conference, San Francisco, in response to the President's Council of Advisors on Science and Technology (PCAST) report calling for a universal exchange language for healthcare.
1. RDF is the best available candidate for a universal healthcare exchange language.
2. Electronic healthcare information should be exchanged in a format that either: (a) is an RDF format directly; or (b) has a standard mapping to RDF.
3. Existing standard healthcare vocabularies, data models and exchange languages should be leveraged by defining standard mappings to RDF, and any new standards should have RDF representations.
4. Government agencies should mandate or incentivize the use of RDF as a universal healthcare exchange language.
5. Exchanged healthcare information should be self-describing, using Linked Data principles, so that each concept URI is de-referenceable to its free and open definition.
SIGNED:
David Booth, Ph.D., KnowMED, Inc.
Charlie Mead, M.D., MSc., Octo Consulting Group
Michel Dumontier, Associate Professor of Bioinformatics, Carleton University
Tracy Allison Altman, Ph.D., PepperSlice
Rafael Richards MD MS, Johns Hopkins School of Medicine
Olivier Curé, PhD UPEM France

Yosemite Project | Mission: Semantic interoperability of all structured healthcare information - Mozilla Firefox

Yosemite Project | Mission:

http://yosemiteproject.org/

Search

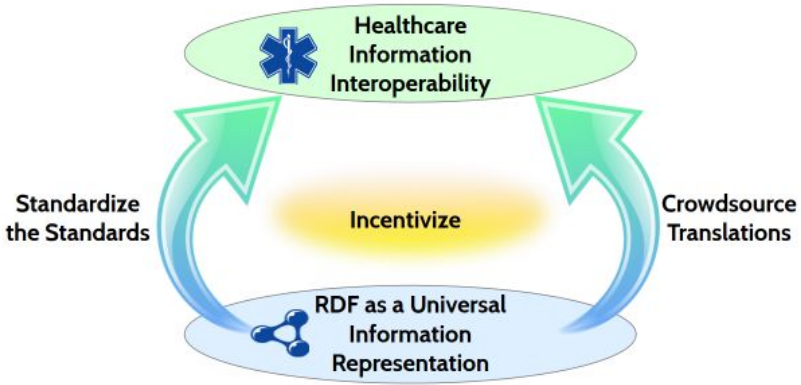
Yosemite Project

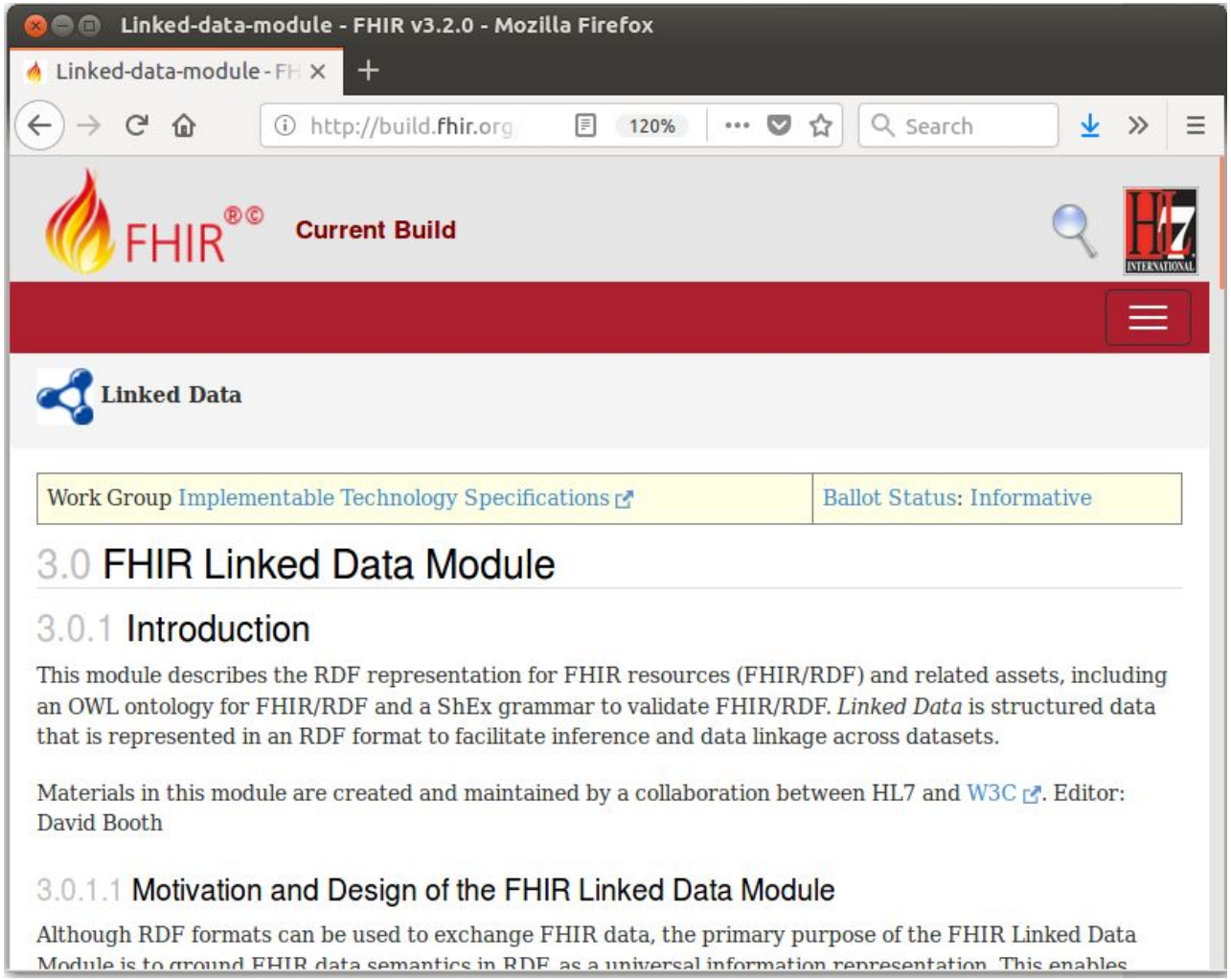
HOME ABOUT THE YOSEMITE PROJECT INTEROPERABILITY ROADMAP

PARTICIPATE RECORDED WEBINARS STEERING COMMITTEE

UPCOMING WEBINARS ACKNOWLEDGEMENTS

Mission: Semantic interoperability of all structured healthcare information




Linked-data-module - FHIR v3.2.0 - Mozilla Firefox
Linked-data-module - FH
http://build.fhir.org
120%
Search
FHIR
Current Build
Linked Data
Work Group Implementable Technology Specifications
Ballot Status: Informative
3.0 FHIR Linked Data Module
3.0.1 Introduction
This module describes the RDF representation for FHIR resources (FHIR/RDF) and related assets, including an OWL ontology for FHIR/RDF and a ShEx grammar to validate FHIR/RDF. Linked Data is structured data that is represented in an RDF format to facilitate inference and data linkage across datasets.
Materials in this module are created and maintained by a collaboration between HL7 and W3C. Editor: David Booth
3.0.1.1 Motivation and Design of the FHIR Linked Data Module
Although RDF formats can be used to exchange FHIR data, the primary purpose of the FHIR Linked Data

Yosemite Project | FHIR RDF as a Bridge to the Semantic Web in Healthcare - Mozilla Firefox

Yosemite Project | FHIR RDF

http://yosemiteproject.org/fhir-rdf

Search

Yosemite Project

MENU

FHIR RDF as a Bridge to the Semantic Web in Healthcare

Harold R. Solbrig
Mayo Clinic

Recorded: Thursday October 12, 2017

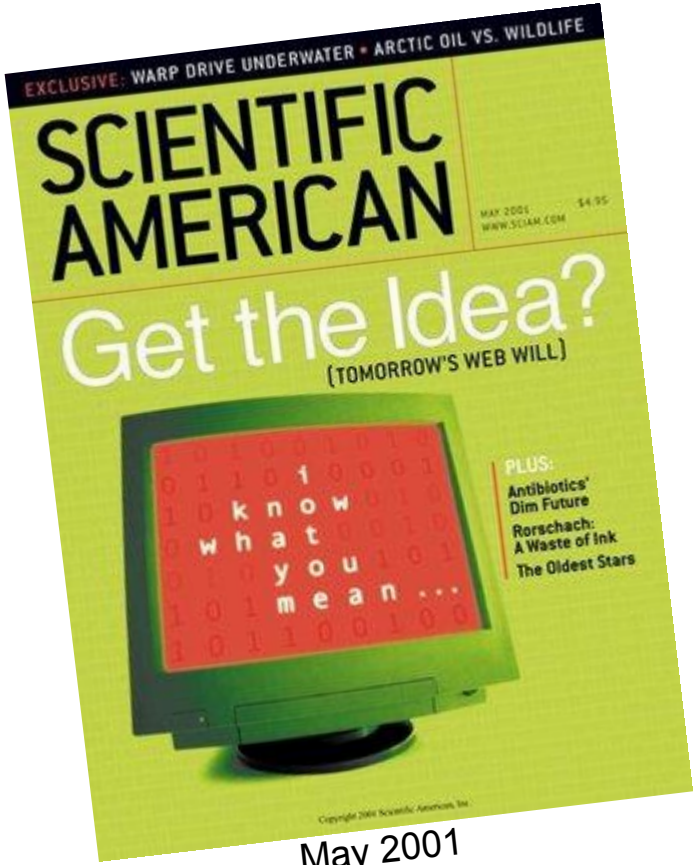
Watch now: ***https://youtu.be/21-dkynEYWk***

Download slides | Associated files

Abstract

The FHIR specification now includes a representation of FHIR resources in RDF — FHIR RDF. FHIR RDF represents a significant milestone towards the realization of the Semantic Web for

History: The Semantic Web vision

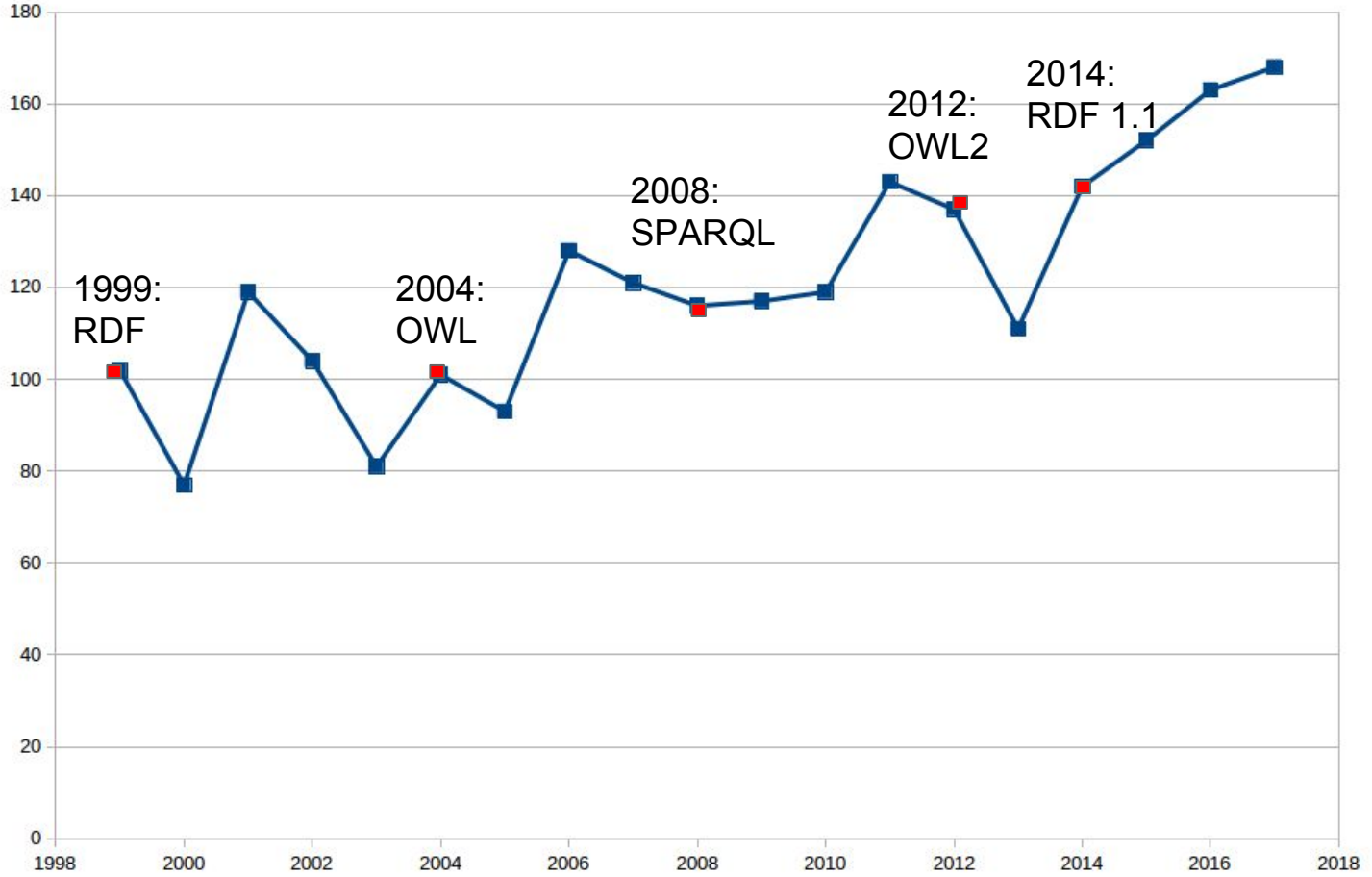


May 2001

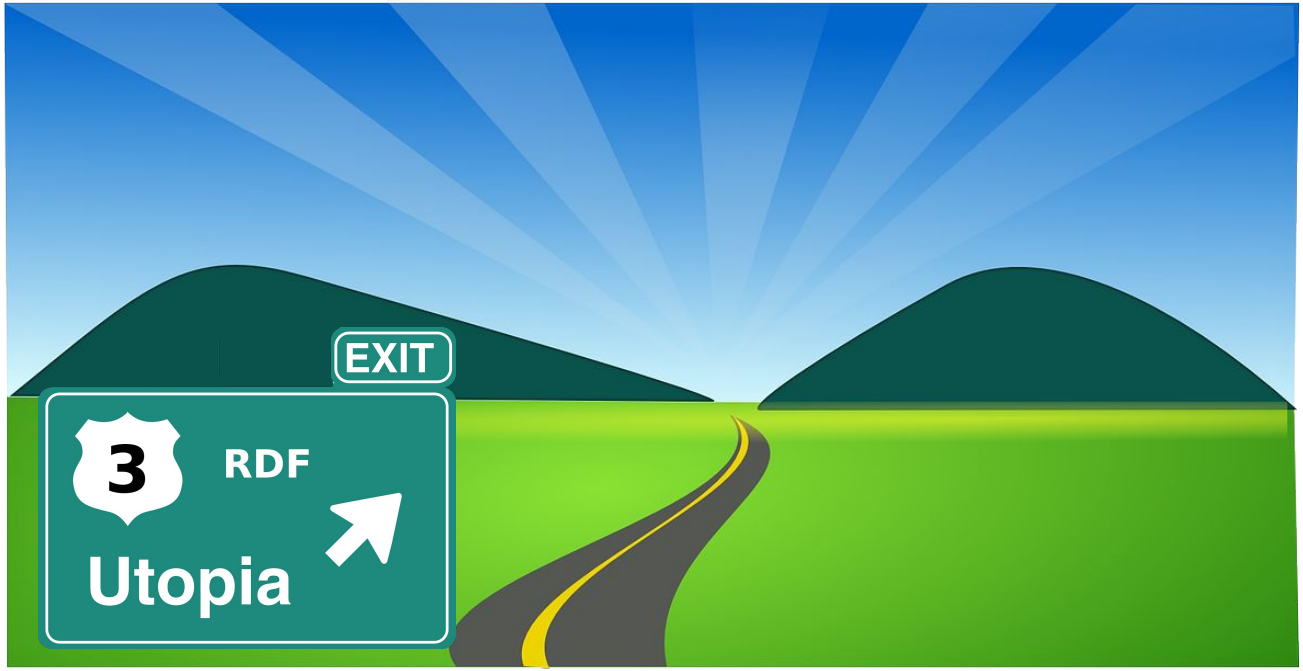
"The Semantic Web is . . . an extension of the current one, in which information is given well-defined meaning."

*"Meaning is expressed by **RDF**."*

RDF: Slow but steady adoption



Google: "RDF" pages by specific years. Date range limits results to a small fraction of the total hits.



- RDF value is well proven, but . . .

- RDF value is well proven, but . . .
- Too hard for **average** development teams

Why is RDF* hard to use?

*RDF ecosystem (includes OWL, tools, etc.)

“Any darn fool can make something complex; it takes a genius to make something simple.”

— Pete Seeger

“Complexity is often caused not by one big flaw, but by an accumulation of small flaws whose effects multiply.”

— My opinion

Why is RDF hard to use?
How can we make it easier?

- Education?
- Tools?
- Standards?

Problems

Tools are scattered

- How to find them?
- Which to use?
- Every team goes through a similar research and selection process

URI allocation

- URIs must be allocated for almost everything in RDF:
  - Things, concepts, properties, etc.
  - Both TBox (ontology) and ABox (instance data)
- Easy in theory but hard in practice!
  - "Cool URIs" are dereferenceable http URIs
  - Domain registration costs money and is not permanent
  - Many possible solutions, no standard best practice

Blank nodes

- Cannot be used in follow-up SPARQL queries
- Subtle, confusing semantics
  - "Name that is not a name"
- Prevent standard RDF canonicalization

RDF canonicalization

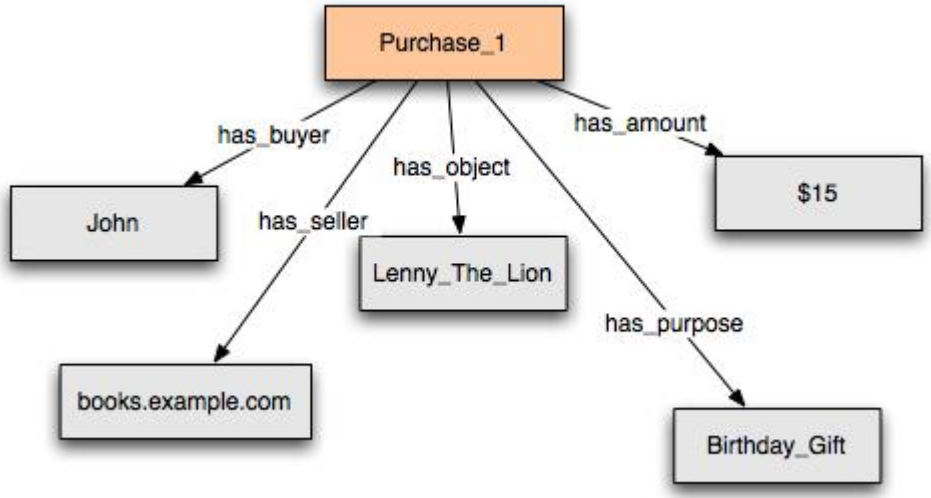
- Canonicalization = standard, predictable serialization
- Essential for diff, digital signatures, etc.
- Other data formats have it. Why not RDF?
  - Answer: Blank nodes
  - Unrestricted blank nodes cause RDF canonicalization to be a "hard problem", equivalent to graph isomorphism problem.

SPARQL-friendly lists

- Very hard to query RDF lists and retain item ordering:
  :derek :children (:alice :bob :carol) .
- Apache Jena offers list:index property
  https://jena.apache.org/documentation/query/rdf_lists.html

Standard n-ary relations

- RDF triples are binary relations
- Workarounds described in 2006: http://www.w3.org/TR/swbp-n-aryRelations
- No standard RDF representation!
- Tools cannot recognize them



Literals as subjects

- RDF should allow "anyone to say anything about anything"
- But RDF does not currently allow literals as subjects

Lack of standard rules language

- W3C RIF is not a rules language
  - RIF = Rules Interchange Language
  - Any rules language can be exchanged in RIF
- Inference is fundamental to RDF value proposition
- App-specific rules are often needed
- But still no standard rules language

Namespace proliferation

- Complexity of the namespace environment (FoF, SKOS, DC and then all the hundreds of specialized namespaces) within a real triple store.
  a. Hard to manage all the namespaces
- Related issue: RDF model does not retain namespaces info!

URI synonyms or renaming

- Different developers should be able to use their own names for things already named by others
  - They do this routinely in other languages
- owl:sameAs is not great for this:
  - Too heavyweight for simple synonyms
  - Only for OWL individuals -- not classes
  - No way to indicate which URI is locally preferred
- Need simple standard ways to rename URIs or define synonyms

Overview of an RDF dataset

- Need tooling for understanding the contents of a triple store
  a. What kinds of relationships are present
  b. What do they mean?
  c. What are the namespaces used, and their purposes?
- In a RDBMS this is fairly simple given an ER diagram
- Need ability to visually zoom in or out, like with google maps

Hard to build mappings between ontologies

- Many mappings are not simple OWL relationships
- Standard rules language could help

Hard to debug SPARQL queries

-

Need robust methods to go from domain experts to ontologies

-

Need higher-level RDF

- RDF++?
- Use coarser-grained atoms?
  - Tree? Concise bounded description?
  - Structure? List of structures?

Need a 4GL for RDF

- Higher-level language for using RDF
- Example: Semantic MediaWiki
  - Extension of MediaWiki (engine for Wikipedia)
  - Allows form-based data entry
  - Generates RDF (without hand-coding Turtle)
- Other examples?

Need a backend for Protege

- Protege does not have a triplestore backend

Need programming language bindings

-

Need more RDF datasets available

-

Paradigm shift from current practice is too big

- Hard to see tangible benefit
- Easy to see abstract benefit
- Development needs to be incremental
  - Both semantics and data
  - Incremental cost and benefit

Open World Assumption (OWA) is a major barrier to understanding

-

Need it simpler like neo4j

-

Need GraphQL for RDF

- Alternative to SPARQL?
- App code uses json, and developers just want to ask for that json

No killer educational tool

- Too much to do it yourself
- Too much jargon to learn
- Need more tutorials and practical documentation -- cookbooks
- Tools are scattered

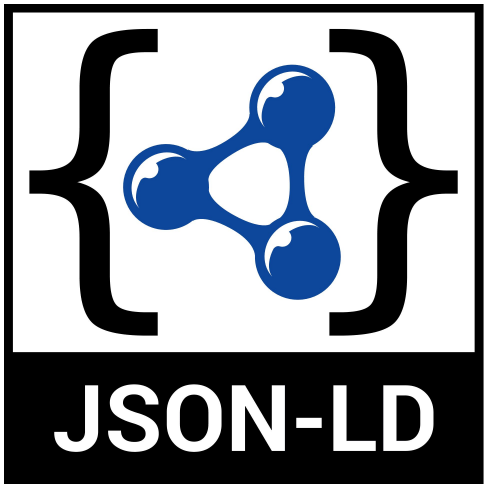
Better tools needed

- . . . in general

Some other ideas

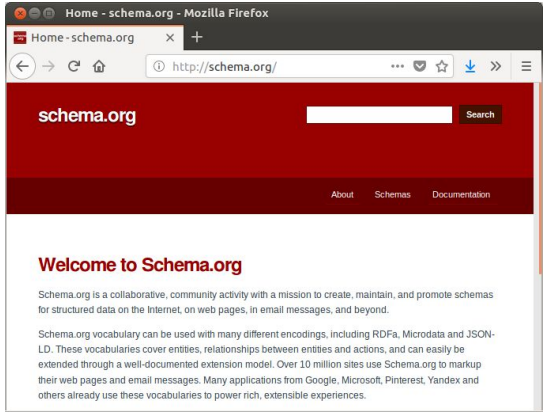
JSON-LD

- JSON-based format for RDF
  - Both JSON and RDF



schema.org

- Facilitates publication of RDF data



Bundled release of RDF tools -- think LAMP, Ubuntu or Red Hat

- RDF-related tools are scattered
- Pre-packaged bundle of commonly used RDF tools
- Analogous to Red Hat / Ubuntu / LAMP bundle

Eliminate explicit blank nodes

- Implicit blank nodes are very helpful:
  :x :colors (:red :blue :green) ;
  :shape [a :Rectangle ; :label "foo"] .
- Implicit blank nodes are fine.
- Explicit blank nodes cause trouble:
  _:b01 :foo :bar .
- **RDF canonicalization** becomes easily feasible if explicit blank nodes are avoided

Allow local IRIs

- Like a combination of:
  - Blank node
  - Relative IRI
  - Skolem IRI
- Syntactically an IRI
- Unique within an RDF dataset
- Intended to be automatically renamed when merging RDF

Local URIs - Straw man

- Syntax:
  `urn:local:`***`foo`***
- When merging datasets **x** and **y**, rename local URIs to be unique in the new dataset:
  - Local URIs from **x**:
    `urn:local:`***`foo`*** `--> urn:local:x/foo`
  - Local URIs from **y**:
    `urn:local:`***`foo`*** `--> urn:local:y/foo`

Local URIs - Why

Pros:

- Easy URI allocation
- SPARQL-friendly alternative to blank nodes
- Compatible with standard tools

Cons:

- Local URIs must be renamed before merging graphs
- New concept -- must be taught

Best practice: [] should declare IFPs

- [...] in Turtle creates an implicit blank node
- This can cause "duplicate" triples when the same RDF is loaded more than once
  - Blank nodes are not reused, hence not recognized as the same node
  - Causes a non-lean graph
  - Causes "wrong" SPARQL results (over counting)
- If inverse functional properties (IFPs) were declared for uses of [], then tools could:
  - Convert blank nodes <--> predictable IRIs
  - Eliminate those duplicate triples

Why is RDF hard to use?
How can we make it easier?

Breakout I: Broadening the base

Raw notes: https://docs.google.com/document/d/1SHZMpiDsrtBpEaXQOQrAuV11VgTbMK96w0Q1Wh28oqg/edit#

Questions:

- How can we lower the entrance hurdle?
- Can we improve tool support; which tools are missing?
- What are the lessons learned in designing our current technology stack that we can apply in the future?
- How do we improve support for scope (time, space,...) and probability/uncertainty?
- When does reasoning actually matter?

Breakout I: Broadening the base

Raw notes: https://docs.google.com/document/d/1SHZMpiDsrtBpEaXQOQrAuV11VgTbMK96w0Q1Wh28oqg/edit#

Ideas:

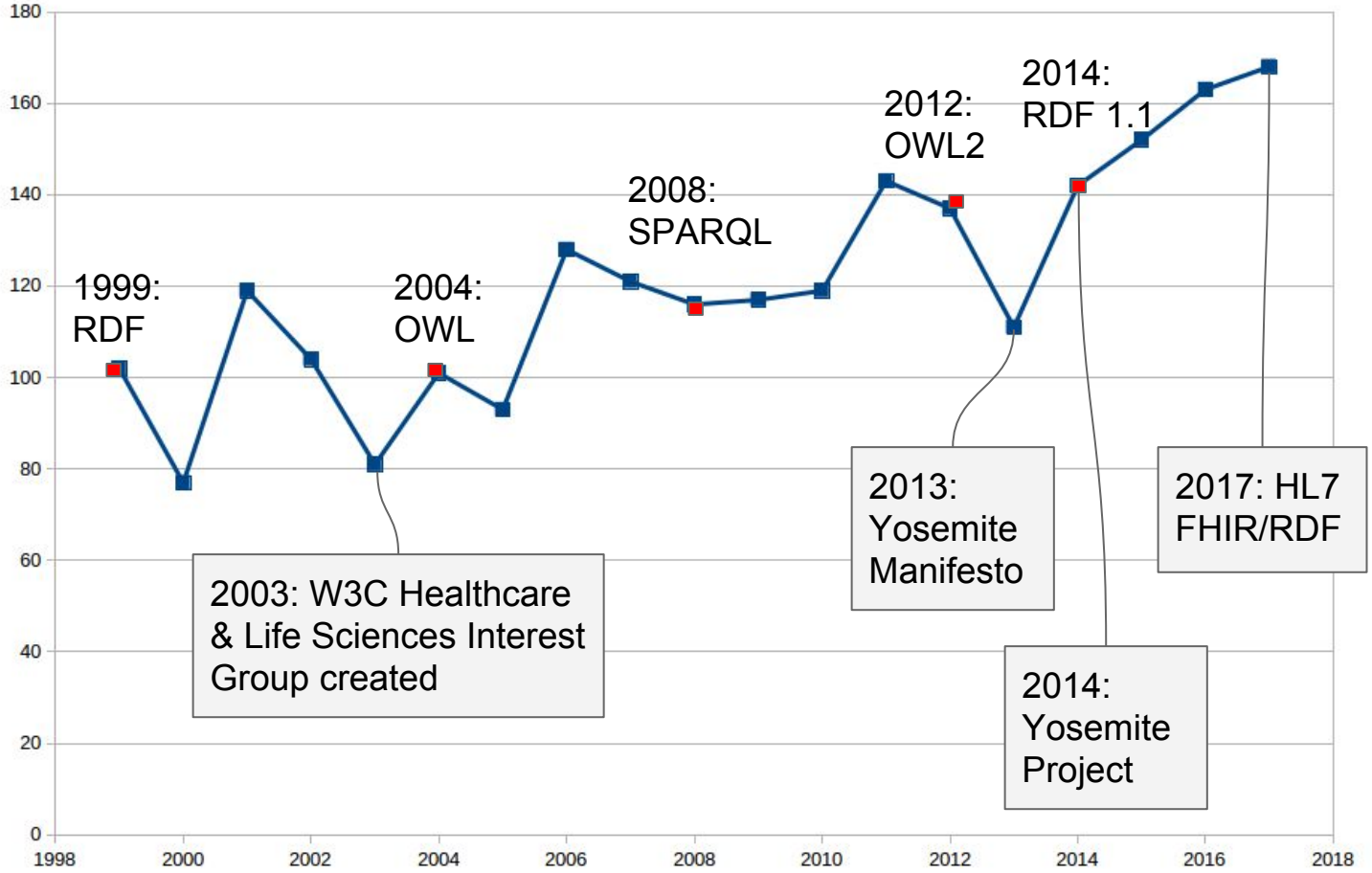
- Need robust methods to go from domain experts to ontologies
- The OWA is a major barrier to understanding
  - rdfs:domain and rdfs:range are not constraints!
  - But users expect them to be
- Best practices: what tools are available?
- Front-end visualization app for RDF data
  - Zoom-in/out like google maps
- Mimic how SQL was adopted
  - Cookbooks

BACKUP SLIDES

Votes - Cambridge Meetup - Feb 2018

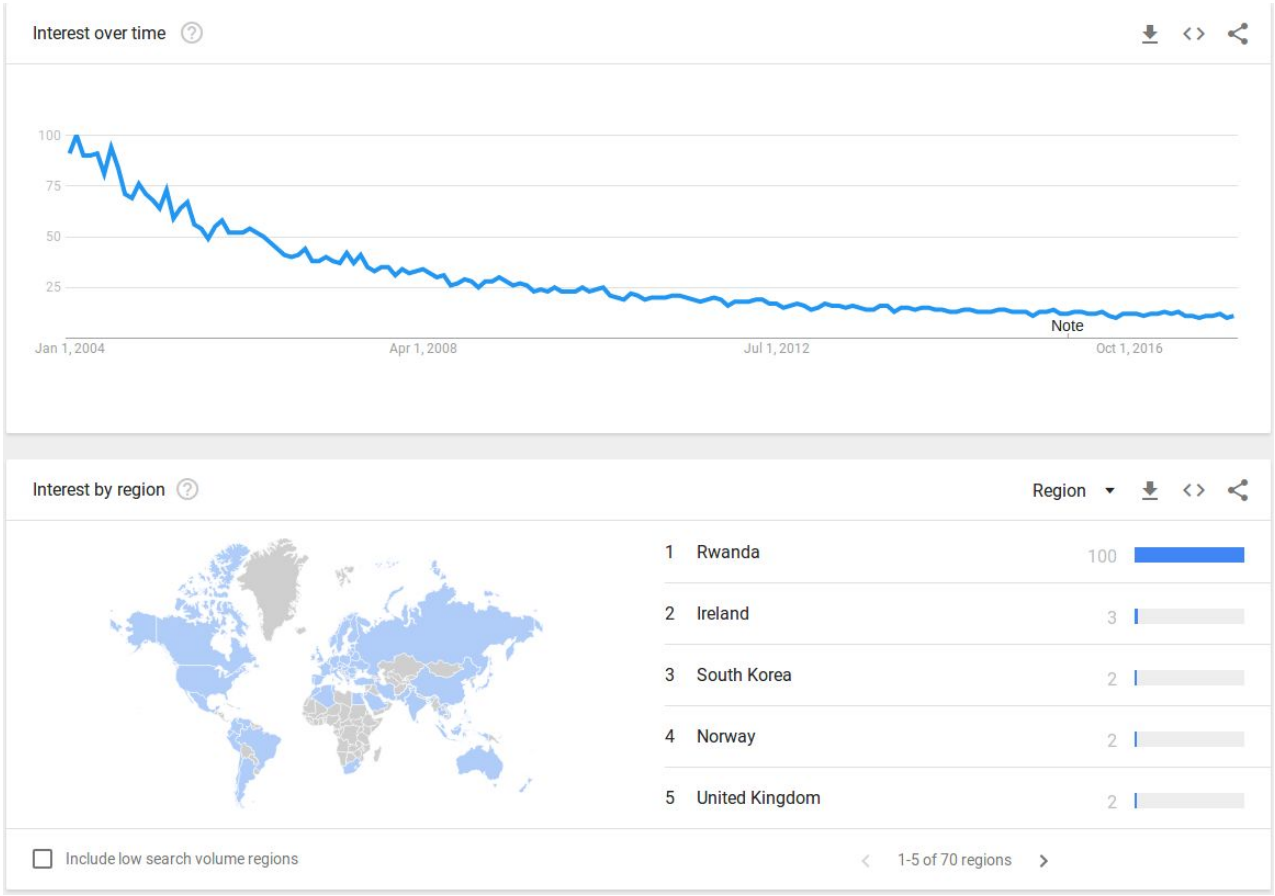
No killer educational tool	9
Overview of an RDF dataset	9
Hard to see tangible benefit of it	8
Lack of standard rules language	8
Standard LAMP stack for RDF	7
Better tools needed	6
Need programming language bindings	6
Literals as subjects	5
Make it simpler like neo4j	5
URI allocation	5
4GL for RDF	4
Namespace proliferation	4
Need RDF datasets available	4
SPARQL friendly lists	4
Standard n-ary relations	4
Very specific jargon to learn	4
Hard to debug queries	3
Need one gigantic ontology	3
RDF canonicalization	3
URI renaming	3
Local URIs	2
Need a backend for Protege	1
GraphQL	1

RDF pages by year*



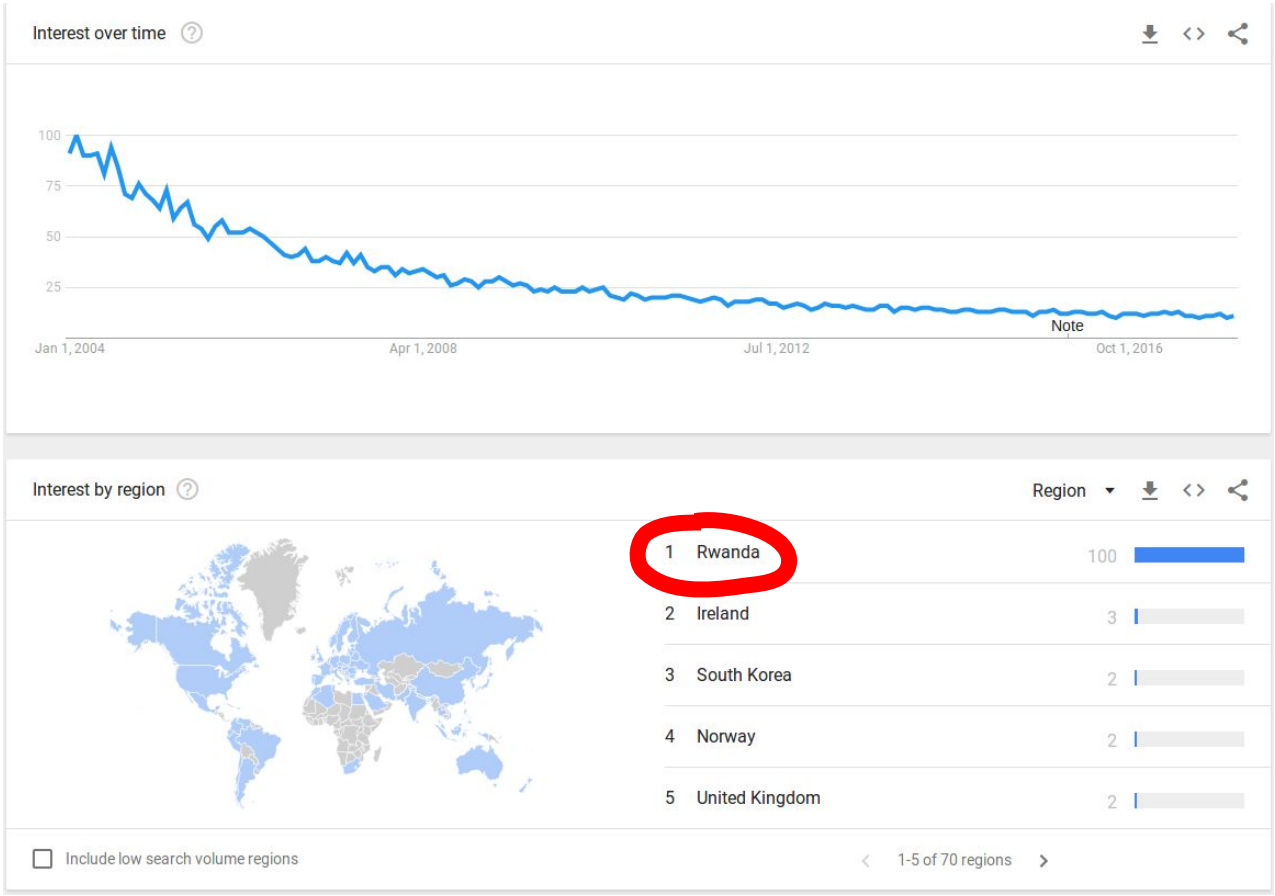
*Search restricted to yearly date ranges, which limits results to a small fraction of the total hits.

RDF searches



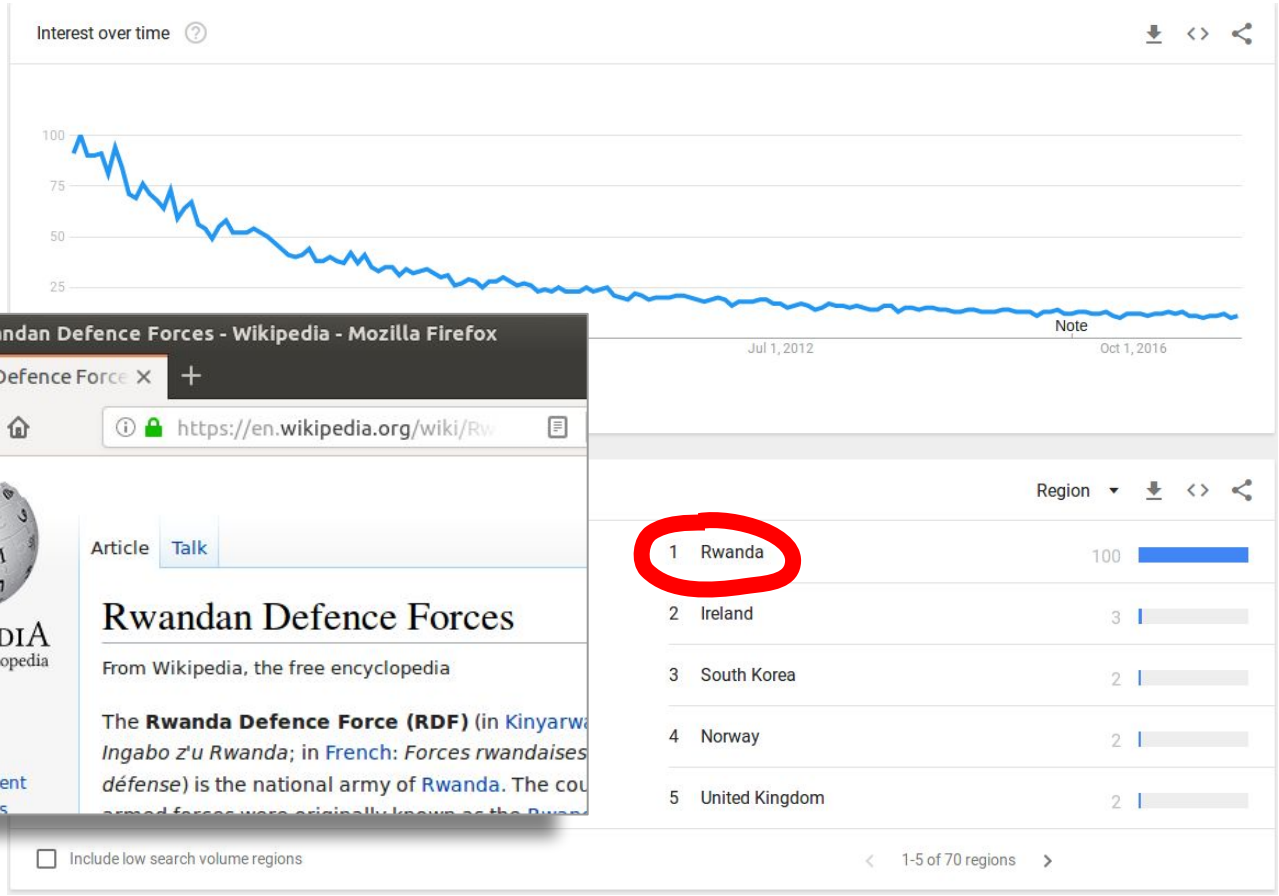
Google Trends

RDF searches



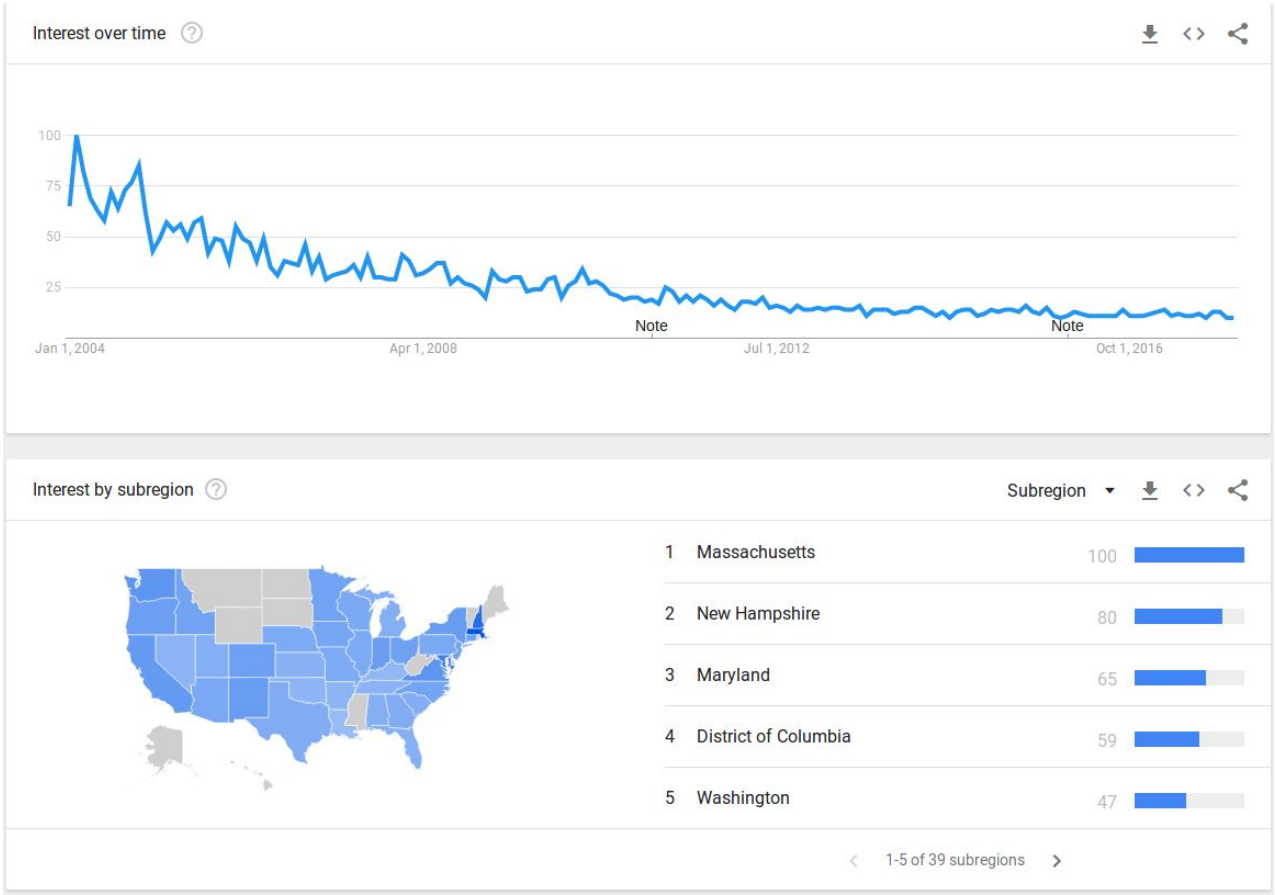
Google Trends

RDF searches



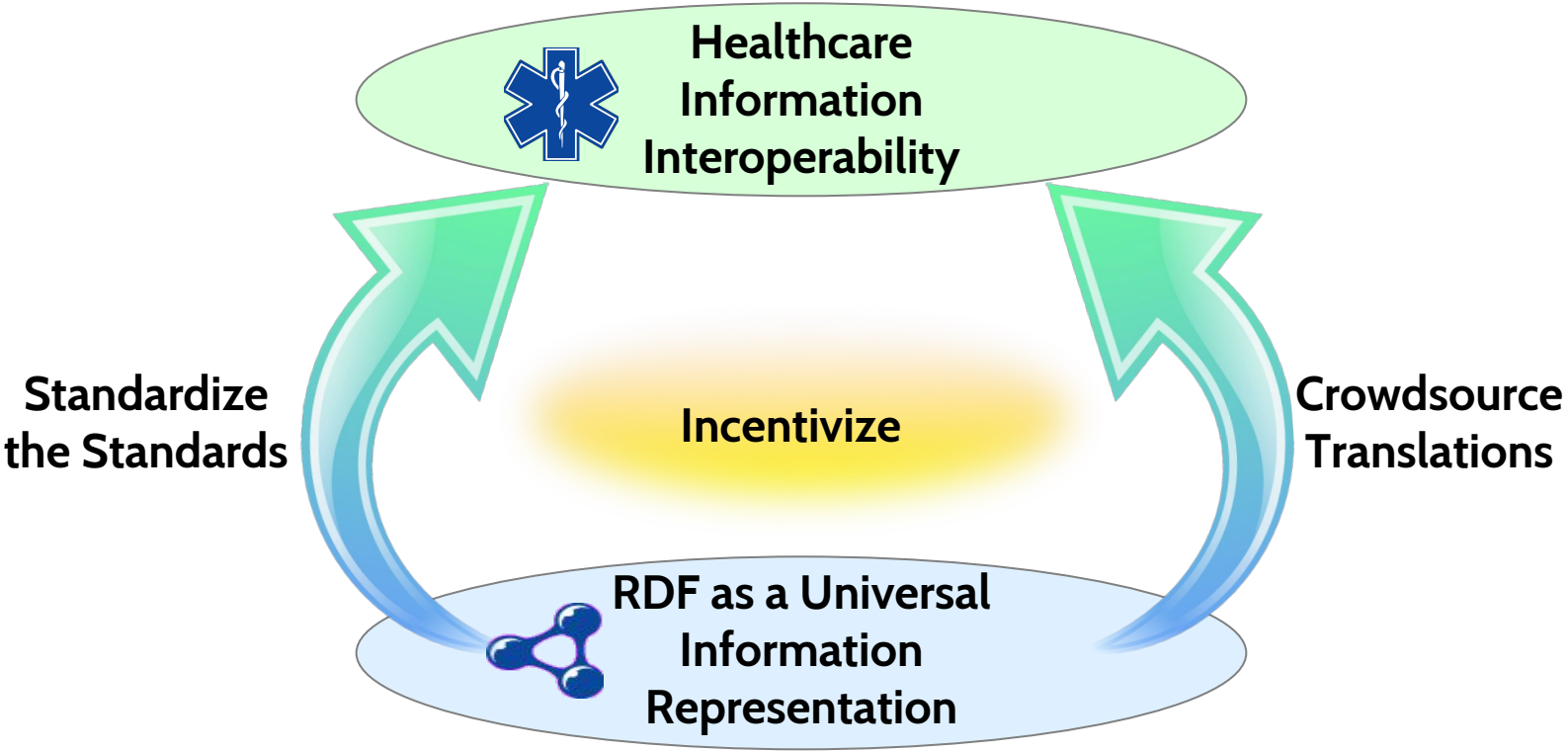
Google Trends

RDF searches - USA



Google Trends

Interoperability Roadmap



http://YosemiteProject.org/