INFO216:
Advanced Modelling

Theme, spring 2017:
Modelling and Programming the Web of Data

Andreas L. Opdahl
<Andreas.Opdahl@uib.no>

Session S14: Development and quality

- Themes:
  - ontology (and vocabulary) development
    - why a method?
    - Ontology Development 101
    - good practices
  - ontology (and vocabulary) quality
    - what is quality?
      - data and information
      - model and language quality
    - ontology (and vocabulary) quality
      - metrics, OntoClean

Readings

- Forum links (cursory):
  - Noy & McGuinness: Ontology Development 101: A Guide to Creating Your First Ontology
- Also:
  - Allemang & Hendler's chapter 14: Good and Bad Modelling Practices

Ontology Development Method

Why development methods?

- Reality (from general IS development):
  - methods are seldom followed completely
    - ...and often not at all
  - real projects are messy, opportunistic...
- Methods remain useful:
  - when you do not know how to proceed
  - for organising larger projects
  - for facilitating group sessions
  - for documenting development processes
  - as repositories of practical advice
  - for organising those repositories

Types of ontology development methods

- Several methods have been proposed
  - no convergence, consensus...
- Different types:
  - language-specific, tool-specific, generic
- *Ontology development 101*:
  - older, but tried and tested
  - well established, much cited
  - quite generic
- Part of a growing body of methods for:
  - ontology evaluation, ontology selection, ontology alignment, ontology integration, ontology extraction, ontology-based information extraction...

Ontology development 101

- 7 main steps:
  1. determine domain and scope
  2. consider reuse
  3. enumerate important terms
  4. define classes and hierarchy
  5. define properties
  6. define property restrictions
  7. create individuals
- Covers both:
  - developing TBoxes: vocabulary development
  - developing ABoxes: ontology population

Points to remember

1) “There is no one correct way to model a domain— there are always viable alternatives. The best solution almost always depends on the application that you have in mind and the extensions that you anticipate.”
2) “Ontology development is necessarily an iterative process.”
3) “Concepts in the ontology should be close to objects (physical or logical) and relationships in your domain of interest. These are most likely to be nouns (objects) or verbs (relationships) in sentences that describe your domain.”

Ontology development 101

- The process is *iterative*:
  - use in applications or for problem solving
  - evaluate and debug, discuss with experts
  - revise
- Rather old (2001):
  - pre-OWL
  - largely pre-Web of Data, pre-LOD
  - ***implications?***

Determine domain and scope

- Step 1. Determine the domain and scope of the ontology
- What is the domain that the ontology will cover?
- For what we are going to use the ontology?
- For what types of questions the information in the ontology should provide answers?
  - sketch a list of competency questions that a knowledge base based on the ontology should be able to answer
  - will serve as completeness tests later:
    - *Does the ontology contain enough information to answer these types of questions?*
    - *Do the answers require a particular level of detail or representation of a particular area?*
  - just a sketch and do not need to be exhaustive
- Who will use and maintain the ontology?

Example (from S13)

- *Competency questions* for
  - a *security ontology* that describes an organisation and its computer systems as concepts, roles and individuals, e.g.:
    - are all the *security levels* subclasses of one another?
      - what is the highest security level of a *temporary*?
      - what is the necessary security level of a *component*?
      - which employees have access to *critical data*?
      - for which *security roles* is an employee qualified?
      - which individuals are *suspicious persons*?
    - *The questions guide identification and description of classes, properties and individuals in clear, direct and compact way!*

Consider reuse

- Step 2. Consider reusing existing ontologies
- OD101 is pre-LOD:
  - bias towards reusing whole ontologies and larger chunks of ontologies
- In a LOD-context:
  - *reuse* becomes more fine-grained
    - we may reuse single (or small groups of) terms
    - ...or we may link to these terms
  - *development for reuse* becomes more important
    - anticipate future uses of our ontology
  - but the basic principles still remain

Important terms

- Step 3. Enumerate important terms in the ontology
- Write down a list of all terms we would like either to make statements about or to explain to a user:
  - What are the terms we would like to talk about?
  - What properties do those terms have?
  - What would we like to say about those terms?
- Do not worry about, e.g.:
  - overlap, relations, classes, properties, restrictions...
- *The following two steps — developing the class hierarchy and defining properties of concepts (slots) — are closely intertwined*

Define classes

- Step 4. Define the classes and the class hierarchy
- Strategy:
  - top-down, bottom-up, inside-out
- Classes represent constructs or concepts in the domain and not the words that denote these concepts
- Synonyms for the same concept do not represent different classes
- Know when to stop:
  - the ontology should not contain all the possible information about the domain
  - do not specialize/generalise more than you need for your application (at most one extra level each way)

...and class hierarchy (1)

- Step 4. Define the classes and the class hierarchy
- Avoid class cycles
- All the siblings in the hierarchy (except for the ones at the root) must be at the same level of generality
- Subclasses of a class usually

  1) have additional properties to the superclass,

  2) have different restrictions from the superclass, or

  3) participate in different relationships than the superclasses

  – ...but additional properties are not always necessary

...and class hierarchy (2)

- If a class has only one direct subclass there may be a modelling problem or the ontology is not complete
- If there are more than a dozen subclasses for a given class then additional intermediate categories may be necessary
- Use disjoint subclasses
- → *OntoClean offers more specific criteria*

Define properties

- Step 5. Define the properties of classes (“slots”)
  - this includes relations (object properties)
- (Perhaps use prototype individuals)
- Know when to stop:
  - the ontology should not contain all the possible properties of and distinctions among classes in the hierarchy
- Subclasses or property values?
  - will it be used in property restrictions of other classes?
  - is the property conceptually important?

Define property restrictions

- Step 6. Define the “facets” of (restrictions on) the “slots”
- Examples of restrictions:
  - cardinality
  - domain
  - “value type” (range: RDFS type or OWL class)
  - inverse properties
  - etc.
- ...same as in OWL

Create individuals

- Step 7. Create instances
- Adding individuals is also a validation technique
- Represent as instances or classes?
  - *this can be difficult!*
  - individual instances are the most specific concepts represented in a knowledge base
  - only classes can be arranged in a subclass hierarchy
    - if there is a natural hierarchy among terms, we should define these terms as classes

Noy & McGuiness' terms

- The OD101 paper does not use OWL
  - ...but the terms should be easy to understand:
    - classes, concepts
    - properties, roles, slots
    - property/role restrictions, facets
- *DAML* is a central precursor to OWL
- *Frames and slots* is an older way to represent knowledge, corresponding to – but not quite like – classes and properties
  - some languages and tools still use these concepts
- *Default slot values* (sec 5.2) do not exist in OWL
- *Value types* are rdfs:ranges

Good practices

- Record rationale for design decisions
- Know when to stop:
  - the competency questions are central
  - specific advice for classes and properties
- Define naming conventions (and stick to them!)
  - use of lower- and upper-case letters
  - singular or plural
  - directed naming (prefixes and suffixes)
  - do not add “-Class”, “-Property”, “-Individual”
  - consistent subclass names

Ontology Quality

Why important to us?

- Quality is the ultimate goal of method
  - a good method is a method that delivers quality
- Quality is also the ultimate goal of development
  - quality of *product*:
    - the product-as-artefact
    - the product-in-use
  - quality of *process*:
    - individual and collective learning
    - fostering agreement
    - revealing disagreement

What is quality?

- Important in many ICT fields
- The products can be, e.g.,
  - data, information, models, programs, languages, knowledge bases, vocabularies, ontologies
- Widely different views:
  - certification
  - adherence to standards
  - compliance to specification
  - fit for purpose
  - fit for use by consumers
- Ultimately related to power:
  - quality for whom?

Data quality

- Several definitions, e.g.:
  - complete, standards based, consistent, accurate and time stamped
  - correctly represent intended real-world phenomena
  - data that are fit for their intended uses
- *Intrinsic qualities* of the data per se
- *Extrinsic qualities* of the data in their use context
  - "information quality"

Data quality dimensions

- Many classifications exist, e.g.:
  - *intrinsic DQ* - "inherent data quality":
    - accuracy, objectivity, believability, reputation
  - *extrinsic DQ* - "data quality in context":
    - *accessibility DQ*: accessibility, access security
    - *contextual DQ*: relevancy, value-addition, timeliness, completeness, amount of data
    - *representational DQ*: interpretability, ease of understanding, concise representation, consistent representation

Wang (1997)

Information quality dimensions

- Examples of information quality dimensions:
  - authority/verifiability
  - scope of coverage
  - composition and organization
  - objectivity
  - integrity
  - comprehensiveness
  - validity
  - uniqueness
  - timeliness
  - reproducibility (of instructive information)

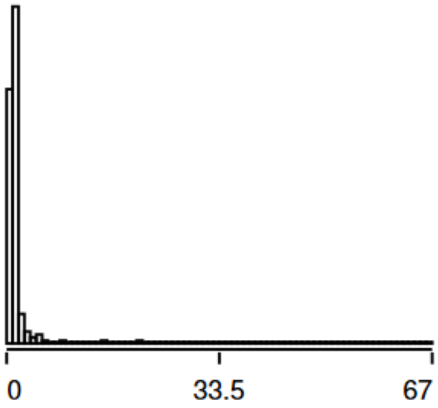
Ontology (vocabulary) quality

- A much newer area
- Many frameworks and methods proposed
  - but no convergence, consensus yet
- Model and language quality frameworks can be used to evaluate ontologies and vocabularies too:
  - as general *models* of a domain
  - as *languages* for describing domains
- Example: OntoClean

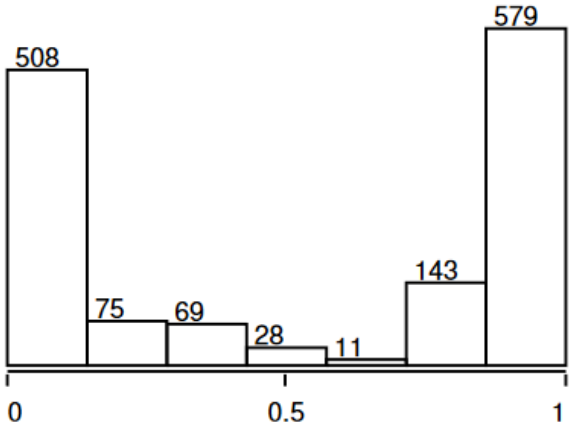
Ontology measures

- Simple metrics for ontologies:
  - Number of Classes *(noc)*
  - Number of Instances *(noi)*
  - Number of Properties *(nop)*
  - Number of Root Classes *(norc)*
  - Number of Leaf Classes *(nolc)*
  - Average Population *(ap)*
  - Class Richness *(cr)*
  - Explicit Depth of Subsumption Hierarchy *(dosh)*
  - Inheritance Richness *(ir)*
  - Relationship Richness Metric *(rr)*
  - Ontorank *(or)*

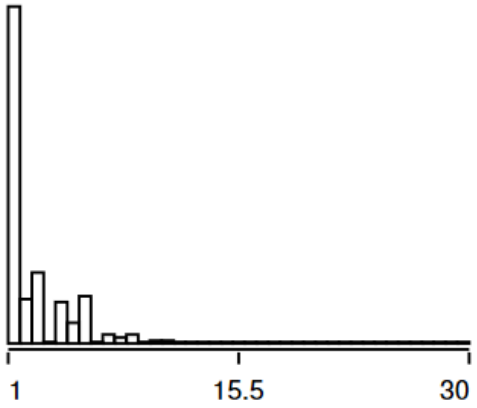
Ontology measures



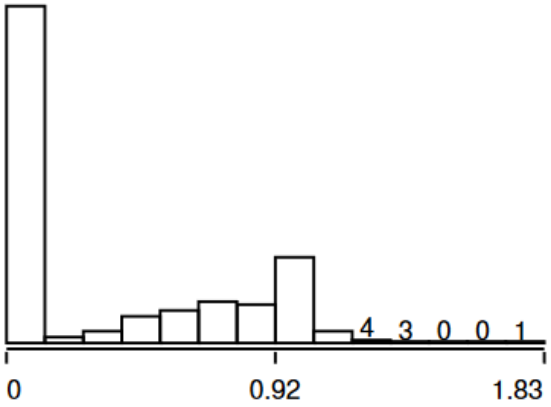
ap



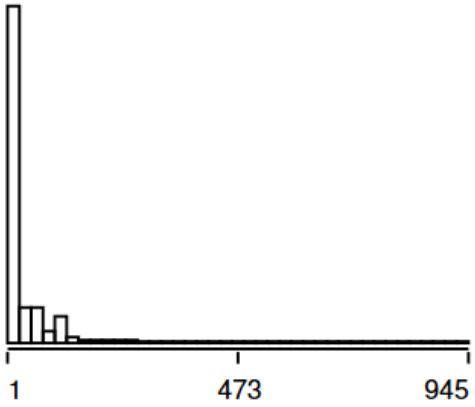
cr



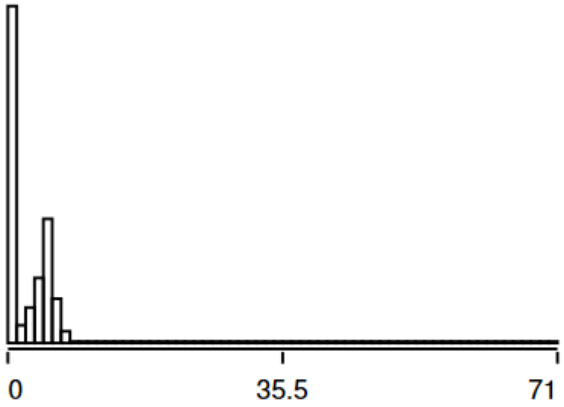
dosh



ir

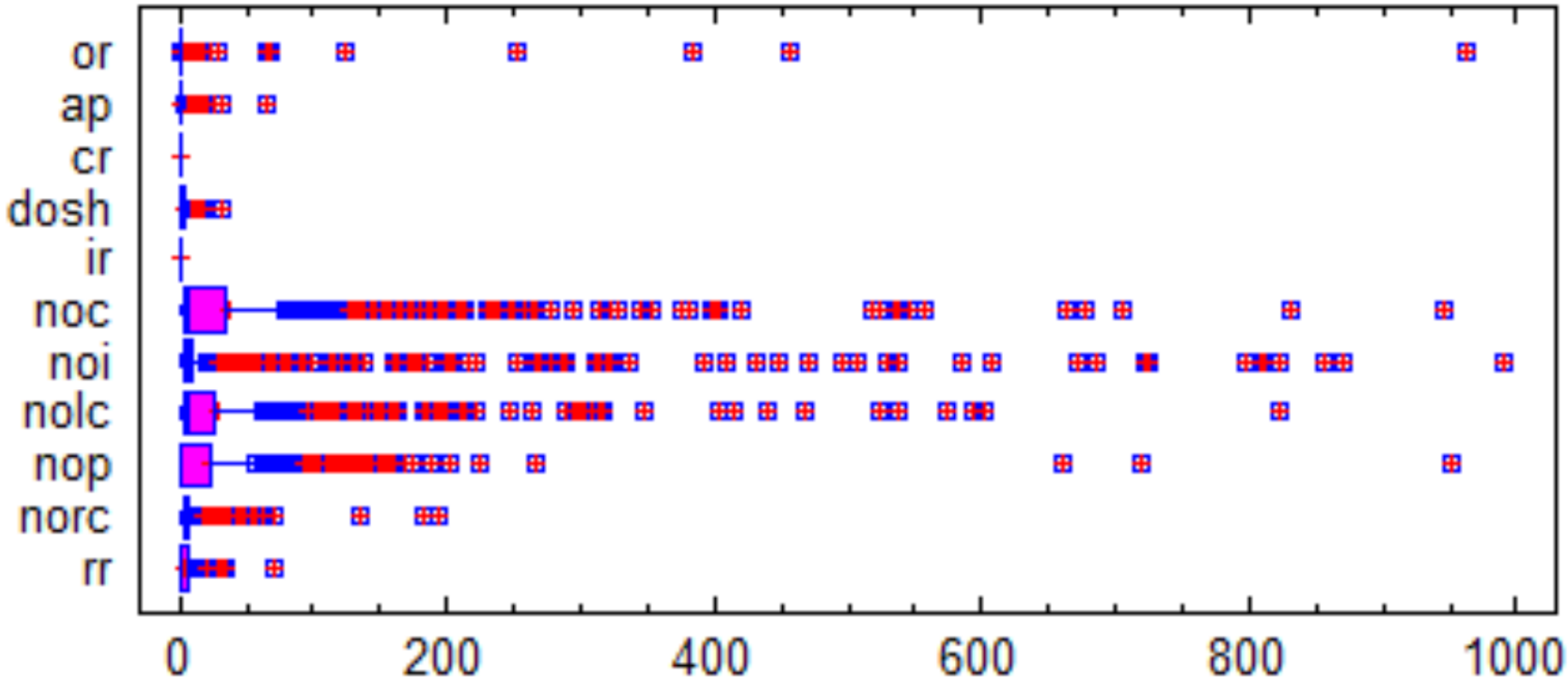


noc



rr

Ontology measures



Uses of ontology measures

- Why ontology measures?
  - they are shallow, say little about semantics
  - a starting point for research
  - no clear correlations with onto rank
  - classification of ontology types
  - fit for purpose, ease of use
    - given alternative ontologies of the same domain, which ontology measures impact dependent measures such as comprehension, recall, problem solving etc.?
  - towards smarter ontology development tools