When and Why to use a Classifier?

Alan Rector

with acknowledgement to
Jeremy Rogers, Pieter Zanstra, & the GALEN Consortium
Nick Drummond, Matthew Horridge, Hai Wang in CO-ODE/HyOntUSE
Information Management Group Dept of Computer Science, U Manchester

Holger Knublauch, Ray Fergerson, … and the Protégé-Owl Team

rector@cs.man.ac.uk
co-ode-admin@cs.man.ac.uk
www.co-ode.org
protege.stanford.edu
www.opengalen.org
Reasons to classify (1)

• Managing Compositional ontologies / Terminologies
  – “Conceptual Lego”
    • Managing combinatorial explosions - the exploding bicycle
  – Empowering users
    • “Just in time” ontologies
      – Give the users the Lego set with limited connectors
  – Organising polyhierarchies / Modularizing ontologies
    • “Normalising ontologies”
    • Multiaxial indexing of resources
  – Providing multiple views -
    • Reorganising the ontology by new abstractions
  – Constraining ontologies & schemas
    • Enforcing constraints
    • Imposing policies
      – For clinical Statements with SNOMED entries
        in request mode context is request
Reasons to classify (2)

- ‘Matching’ instances against classes
  - Resource/service discovery
  - Self-describing storage
    - ‘Archetypes’ & templates

- Providing a skeleton for default reasoning & Prototypes (but not to do the reasoning itself)
  - Molluscs typically have shells
    - Cephalopods are kinds of Molluscs but typically do not have shells
      - Nautiloids are kinds of Cephalopods but typically do have shells
        » Nautilus ancestor are kinds of Nautiloids but do (did) not have shells
  - Biology is full of exceptions
Classification is about Classes

• Classification works for
  – Organising & constraining classes / schemas
  – Identifying the classes to which an instance definitely belongs
    • Or those to which it cannot belong

• Classification is open world
  – Negation as unsatisfiability
    • ‘not’ == ‘impossible’ (“unsatisfiable”)
  – Databases, logic programming, PAL, queries etc are closed world
    • Negation as failure
      – ‘not’ == cannot be found
Reasons not to Classify

• To query large number of instances
  – Open world ("A-Box") reasoning does not work over large numbers of instances

• If the question is closed world
  • E.g. "Drugs licensed for treatment of asthma"

• If the query requires non-DL reasoning
  • E.g. numerical, optimisation, probabilistic, …
    – Would like to have a more powerful hybrid reasoner

• For Metadata and Higher Order Information
  – Classifiers are strictly first order
    • A few things can be ‘kluged’

• If there are complex defaults and exceptions
  – “Prototypical Knowledge”
    • E.g. “Molluscs typically have shells”
      – NB Simple exceptions can be handled, but requires care
Use instead

• To query large numbers of instances OR
  If the query is closed world
  – Queries / constraints over databases
  – Instance stores / triple stores / …
  – Rules
    • DL-programming
    • JESS, Algernon, Prolog, …
  – Belief revision / non-monotonic reasoning

• If query requires Non DL Reasoning
  – Hybrid reasoners or ?SWRL?
    • No good examples at the moment

• For defaults and Exceptions & Prototypical Knowledge
  – Traditional frame systems
    • More expressive default structure than Protégé
      – Exceptions for classes as well as instances
        » Over-riding rather than narrowing
Classification to build Ontologies: Conceptual Lego
Logic-based Ontologies: Conceptual Lego

“SNPolymorphism of CFTRGene causing Defect in MembraneTransport of ChlorideIon causing Increase in Viscosity of Mucus in CysticFibrosis…”

“Hand which is anatomically normal”
Linking taxonomies: Conceptual Lego Normalisation

Species

Genes

Protein

Function

Disease

CFTRGene in humans

Protein coded by (CFTRgene & in humans)

Membrane transport mediated by (Protein coded by (CFTRgene in humans))

Disease caused by (abnormality in (Membrane transport mediated by (Protein coded by (CFTR gene & in humans)))))
Conceptual Lego and Normalisation
Practical Example
Take a Few Simple Concepts & Properties
Sickle cell disease is a disease caused by some sickling haemoglobin.
Cystic fibrosis is caused by some non-normal ion transport that is the function of a protein coded for by a CFTR gene.
"Diseases linked to CFTR Genes"
We have built a simple tree

easy to maintain
Let the classifier organise it
If you want more abstractions, just add new definitions (re-use existing data)

“Diseases linked to abnormal proteins”
And let the classifier work again.
And again –
For a view based on species

“Diseases linked genes described in the mouse”
And let classifier check consistency
(My first try wasn’t)
Normalising (untangling) Ontologies

Structure

Structure
Part-whole
Function

Function

Part-whole
Untangling and Enrichment

Using a classifier to make life easier

Substance
- Protein
  - ProteinHormone
  - Insulin
  - Enzyme
  - ATPase
- Steroid
  - SteroidHormone
  - Cortisol
- Hormone
  - ProteinHormone
  - Insulin
  - SteroidHormone
  - Cortisol
  - Catalyst
- Enzyme
  - ATPase
Normalisation & Quality Assurance

• Humans recognise errors of commission easily
  – Miss errors of omission

• Classifiers convert errors of omission to errors of commission
  – Inadequate definitions create “orphans”
    • BodyPart
      Parts of Heart
      Ventricle
      ...
      CardiacSeptum

• Classifiers flag errors of commission
  – Over definition leads to inconsistency (unsatisfiability)
    • “Pneumonia located in the brain”
Enforcing constraints & policies

- A class with both necessary & sufficient and additional necessary conditions acts as a rule
- The Unit testing Framework supports checking that rules are enforced
A Probe class to check a constraint

All Tests Passed
Use of beta blocker in asthma

Experience: Normalised ontologists lead to clean default inheritance
When to Classify

• … but isn’t having a classifier an intolerable overhead for the applications?
  – It depends on the life cycle you choose

• Life cycles
  – Pre-coordination
  – Just in time coordination
  – Post Coordination
Pre-coordination
If the terms can be enumerated in advance

Asserted form “Sources” → Classifier “Compiler” → Classified form “binary”

(“high level language”
“intermediate representations”)

Overwhelming the dominant pattern today.
“binary” can be
OWL Light RDF(S),
XML Schema, OBO, …
Commit Results to a Pre-Coordinated Ontology

Assert ("Commit") changes inferred by classifier
Post coordination
When there are combinatorially many potential terms

- “Lazy classification” on demand

Big on the outside: small kernel on the inside
Avoid the exploding bicycle
Post Coordination

Authors → Asserted form “Sources” → Asserted Ontology Store → Classifier “Compiler”

Terminology Services

Application

But requires a classifier available at application time

(& intermediate representations)
A Compromise
Just in Time Coordination

Authors

Asserted form “Sources”

(& intermediate representations)

Asserted Ontology Store

Terminology Services

Classifier “Compiler”

Pre-coordinated Cache

Application

Only the occasional new notion Requires classification - need not be real time

Users
Summary

- **Why Classify**
  - **Managing Compositional ontologies / Terminologies**
    - “Conceptual Lego”
    - Empowering users - just in time classification
    - Providing views & deferring decisions on abstractions
    - Quality assurance
  - **Constraining concepts & schemas**
  - **Providing a skeleton for default reasoning & Prototypes**

- **When to classify**
  - **Pre-coordination**
  - **Just-in-time coordination**
  - **Post-coordination**
Summary: When to Classify?

*Applications do not need a classifier to benefit from classification*

- **Pre-coordination**
  - If concepts/terms can be predicted
  - When classifier is not available at run time
  - When we must fit with legacy applications

- **Post-coordination**
  - When a few concepts are needed from a large potential set
  - When a classifier is available
    - and time cost is acceptable
  - When applications can be built or adapted to take advantage
Idiopathic Hypertension

In our company’s studies

In Phase 2 studies

Idiopathic Hypertension

In our co’s Phase 2 study
A problem in the digital anatomist

- To an anatomist, the Pericardium and the Heart are separate organs
- To a clinician they are part of the same organ
  - A disease of the pericardium counts as a kind of heart disease
Represent context and views by variant properties

Pericardium is_part_of Organ is_part_of OrganPart

is_clinically_part_of

Heart is_part_of CardiacValve

isstructurally_part_of

Disease of (Heart or part-of-heart)

Disease of Pericardium
Protégé-OWL alternative views

Disorder of “Clinical heart”
“Disorder of heart of any part of the heart”
(including clinical and functional parts)

Disorder of “FMA heart”
“Disorder of heart or any structural part of the heart”