Structure-Based Partitioning of Large Concept Hierarchies

Heiner Stuckenschmidt, Michel Klein
Vrije Universiteit Amsterdam

1

Outline

- Motivation: The Case for Ontology Partitioning
- A Partitioning Method ← Lots of Pictures
  - Create a dependency graph
  - Determine the strength of dependencies
  - Compute Partitioning
  - Improve Partitioning
- Experiments
- Discussion

2

Ontologies and the Semantic Web

- Ontologies are the backbone of semantic web applications
  - Content-Based Retrieval
  - Information Integration
  - Web Service Discovery
- More and More Large Ontologies become available
  - General Purpose: Open Directory, Yahoo!, …
  - Medicine: GALEN, UMLS, FMA, …
  - Business: UNSPSC, e-class, …
- Maintenance and handling is becoming a problem.

The Case for Partitioning

- Distributed Development and Maintenance
  - Experts can update their portion independently of other parts
- Selective Publication and Use of Terminologies
  - Stable subsets can be published in the development phase
  - Users can chose relevant subset of an ontology
- Manual Inspection and Validation
  - Small, coherent modules are easier to understand
- Editing, Visualization and Reasoning
  - Available tools do not scale to very large ontologies

An Abstract View of the Problem

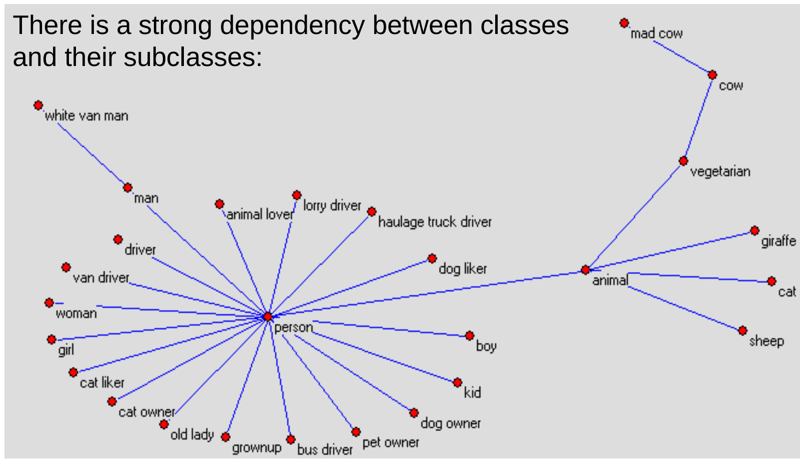
- Despite the standardization of Languages there is no agreement on the way ontologies are represented.
  - All ontologies contain classes
  - Most organize them in a hierarchy
  - Many define relations between classes
  - Some provide formal definitions of classes
- We concentrate on partitioning ontologies into disjoint sets of concepts. Class hierarchy, relations and definitions provide input for the partitioning algorithm.

Overview of the Process

1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions
4. Improve Partitioning

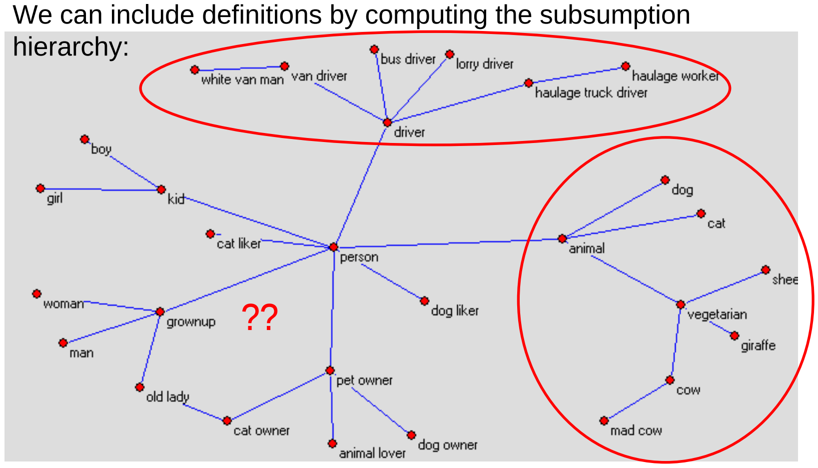
Dependencies I: Hierarchy

- There is a strong dependency between classes and their subclasses:

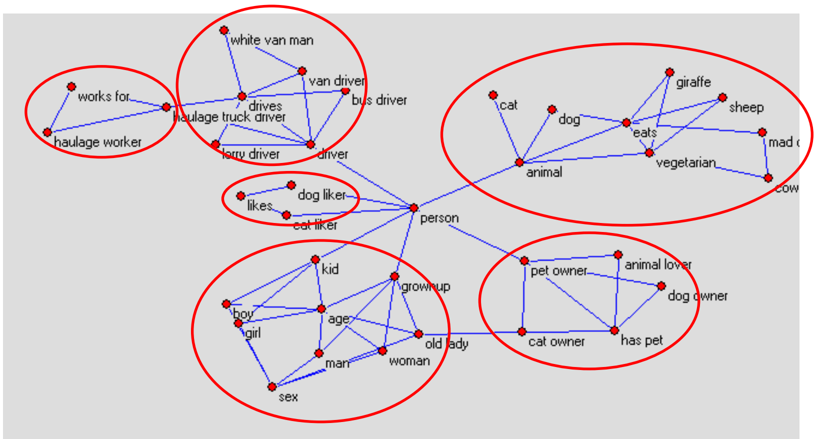


Dependencies I: Hierarchy

- We can include definitions by computing the subsumption hierarchy:



Dependencies II: Shared Relations



Overview of the Process

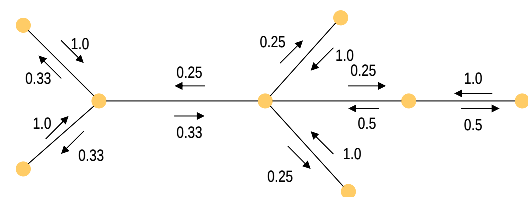
1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions
4. Improve Partitioning

Relative Strength Networks

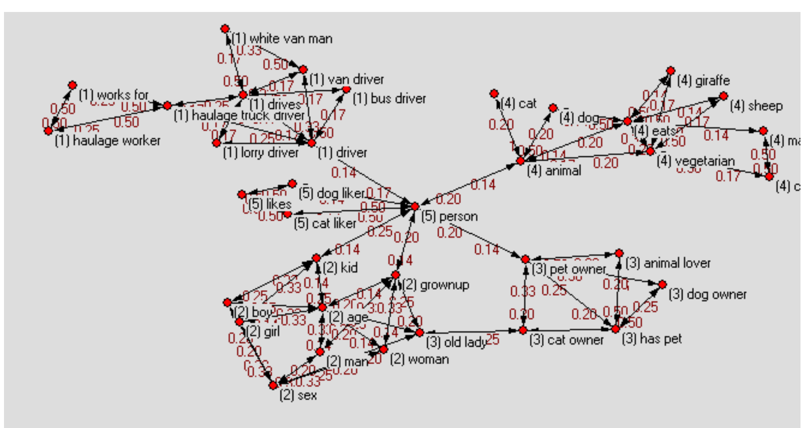
- Compute relative strength [Burt '92] of dependencies

$$w(c_i, c_j) = \frac{a_{ij} + a_{ji}}{\sum_k a_{ik} + a_{ki}}$$

- Example:



Result for the Example



Overview of the Process

1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions
4. Improve Partitioning

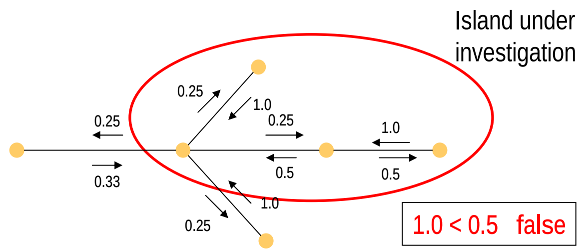
Computing Islands

- We use maximal line islands [Batagejl 2000] to compute partitions in the dependency graph.
  - *A set of vertices is a line island in network if and only if it induces a connected subgraph and the lines inside the island are stronger related among them than with the neighboring vertices. In particular there is a maximal spanning tree T over nodes in the island such that:*

$$\max_{\{(v,v')\in D \mid (v\in I \wedge v'\notin I)\vee(v'\in I\wedge v\notin I)\}} w(v,v') < \min_{(u,u')\in E_T} w(u,u')$$

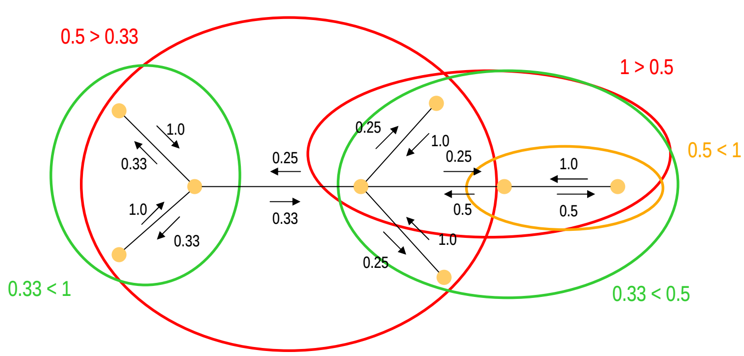
- The minimal weight in the spanning tree is called the height of the island

Computing Islands: an Example



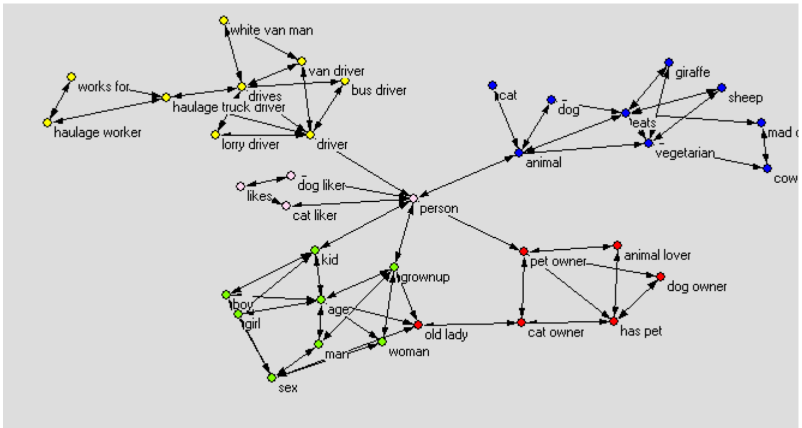
1. Determine maximal spanning tree T
2. Determine minimal weight in T = Height
3. Determine maximal weight for edges from and to island
4. Island if maximal weight < Height

Understanding Islands



$$\max_{\{(v,v')\in D \mid (v\in I \wedge v'\notin I)\vee(v'\in I \wedge v\notin I)\}} w(v,v') < \min_{(u,u')\in E_T} w(u,u')$$

Result for the Example



17

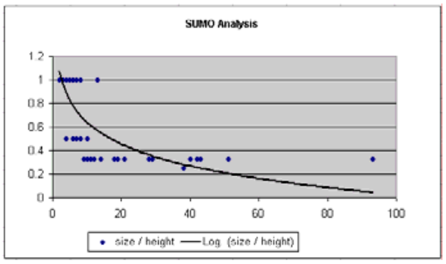
Overview of the Process

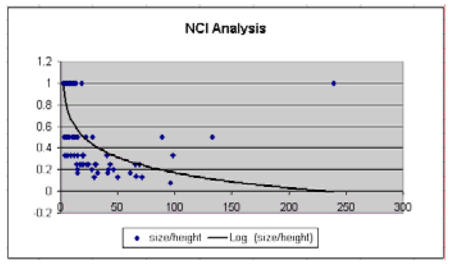
1. Create Dependency Graph
2. Determine Strength of Dependencies
3. Compute Partitions
4. Improve Partitions

18

Improving Partitions (work in progress)

- Islands are often very small (2 - 4 nodes) resulting in unwanted partitions of the ontology.
- Observation: small islands almost always have a large height value (1 or 0.5):



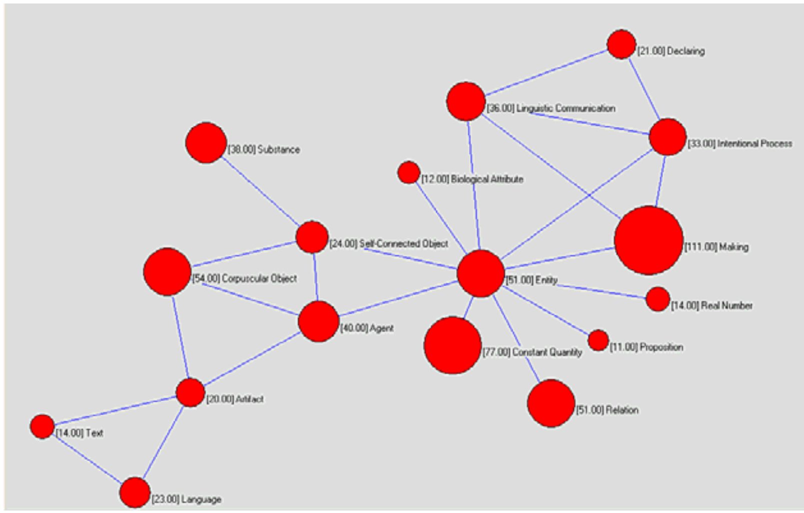


- We can use the height to determine unwanted partitions !

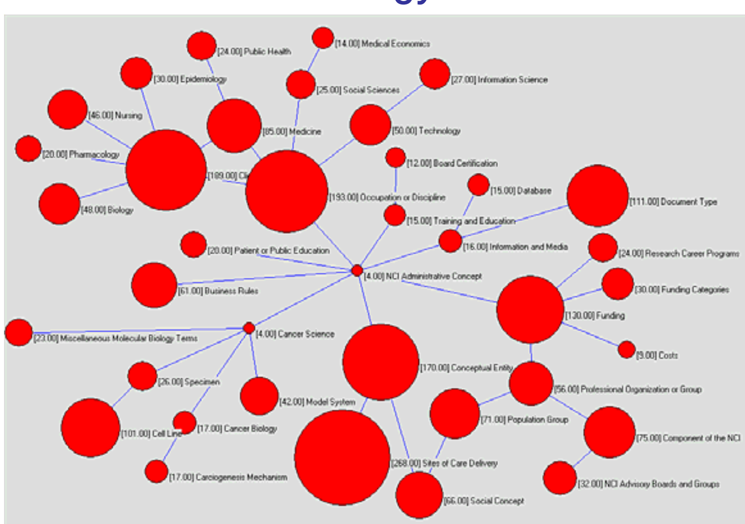
Experiments

- The ACM Computer Science Classification
  - 1000 concepts, fixed hierarchy
- The Standard Upper Model Ontology SUMO
  - 600 concepts, fixed hierarchy
- The NCI cancer ontology
  - Subset with 2400 concepts, fixed hierarchy
- The DICE ontology
  - About 2000 concepts, classified hierarchy
- Details about results can be found at: http://swserver.cs.vu.nl/partitioning/

Result for SUMO Ontology



Result for NCI Ontology



Discussion

- We developed a customizable method for partitioning ontologies
  - Different ways of creating the dependency graph
  - Use of different measures of dependency
- The method is well suited to the needs of the semantic web:
  - It is independent of the language and encoding
  - It scales up to very large ontologies (the current limit is our capability of assessing the result)
- Open Problems are:
  - The lack of a software environment to perform intensive testing
  - The lack of a golden standard to evaluate against

23