Aggregating and Reasoning

Introduction

- First step, Convert from Traditional format to RDF format
- Allows ontology store and manipulate with ontology-based tools
- Process need to assign identifiers to resources
- Re-represent data in terms of a shared ontology (Ex. FOAF)
- While conversion preserve their original schema
- Apply *ontology mapping to unify data on the schema level*
- *Done by mapping classes (types)* and properties from different schemas to a shared ontology (Ex. FOAF)

Contd...

- Aggregation task need to find identical resources across data sets

Two step process:

- Requires domain-specific knowledge to consider two instances are same

- RDF or OWL has limited knowledge to capture instance equality

- Determine equality applying threshold based similarity measure need procedural representation of knowledge

- Determine rules or procedures in the domain to carry out the actual instance unification or "*smushing*"

- Perform smushing as a reasoning task iteratively executing rules until no more equivalent instances is found

Representing Identity

- Main advantages of RDF over other representation to uniquely identify resources

- Primary mechanism is assignment of URIs to resources

- Every resource, except blank nodes is identified by a URI

- Many candidates for identifier exists (Ex. ISBN, ISSN etc.)

- All are unique but not single Ex. same publication is assigned two or more identifiers (DOIs)

Contd...

- Multiple identifiers represented in RDF in two ways:
- First, introduce separate resource and use the identifiers as URIs
- Alternative way, to chose one of the identifiers and use it as a URI
- Equality of these resources can be expressed using the *owl:sameAs property*
- Good practice for Resource identifiers need to conform to the URI specification

Contd...

- Good URIs are unique and stable
- Good URIs should be unambiguous
- If URI changes, there is no way to rename resources
- Only solution is to introduce a new resource and assert its equivalence with the old resource
- Difficult to notify changes to remote systems
- Abstract concepts one should not choose URIs that are recognized by a server

On the notion of equality

- RDF/OWL represent (in)equality using the *owl:sameAs and owl:differentFrom properties*

- *E*quality meaning depends on domain knowledge and level of modelling

- Requires characteristic to consider, level of modelling Ex. Individual

or group (role equivalence)

Leibniz-law

- Identity of Indiscernibles or Leibniz-law: For any *x* and *y*,
if *x* and *y* have all the same properties, then *x* is identical to *y*.

- $\forall P : P(x) \leftrightarrow P(y) \rightarrow x = y$

- converse of the Leibniz-law is called Indiscernibility of Identicals: For any *x* and *y*, if *x* is identical to *y*, then *x* and *y* have all the same properties

- $\forall P : P(x) \leftrightarrow P(y) \leftarrow x = y$

Contd...

- Leibniz-law different interpretations in open and closed worlds
- open world no. of properties unknown Leibniz-law is not useful
- closed world iterate over all properties to assume two resources equal
- closed world assumption undesirable in situation due to lack of information

Ex. same gender, but not want to assume they are identical

- Leibniz-law stronger than our natural notion of equality Ex. 2 perfect spheres at distance d to each other
- Solution - to introduce weaker notions of equality, exclude impure, extrinsic properties, Ex. *foaf :based near, foaf :gender*

Contd...

- OWL built on open world assumption, which means Leibniz-law cannot be used to infer identity

- *But owl:sameAs conforms to Formula of* Indiscernibility of Identicals (self-identical objects)

- two symbols denote the same object and thus they must be indiscernible (interchangeable in statements)

- $(s1, owl{:}sameAs, s2) \land (s1, p, o) \rightarrow (s2, p, o)$
- $(p1, owl{:}sameAs, p2) \land (s, p1, o) \rightarrow (s, p2, o)$
- $(o1, owl{:}sameAs, o2) \land (s, p, o1) \rightarrow (s, p, o2)$

Contd...

- The reflexive, symmetric and transitive properties of sameAs:

- $\forall s : (s, owl{:}sameAs, s)$
- $(s1, owl{:}sameAs, s2) \rightarrow (s2, owl{:}sameAs, s1)$
- $(s1, owl{:}sameAs, s2) \wedge (s2, owl{:}sameAs, s3) \rightarrow (s1, owl{:}sameAs, s3)$

- Below is not inconsistent in open world assumption
- *(s1, owl:sameAs, s2)*
- *(s1, foaf :name, ”John”)*
- *(s2, foaf :name, ”Paul”) [But inconsistent in closed world]*
- *(because s1 has the foaf :name Paul and s2 has the foaf :name John exist somewhere, but missing assumption in closed world)*

Determining equality

- OWL has limited set of constructs for (in)equality statements

- Equality proved through - Functional and inverse functional properties (IFPs) and maximum cardinality restrictions

- inverse-functional – Ex. *foaf :mbox* , cardinality – Ex. *ex:hasParent*

- Ways to conclude 2 symbols do not denote the same, object*owl:differentFrom -* *Ex.* instances of disjoint classes

- All knowledge cannot be expressed in OWL, need more expressive rule languages Ex. Rule Language - Horn logic

Contd...

- Sometimes expressivity of rule languages is not sufficient
- Concatenation of literals is not part of either DL or rule languages – Ex. matching of person names
- programming language or transformation languages such as XSLT can perform using data manipulation functions
- Solution - combine declarative and procedural reasoning
- Reasoning combines - data manipulation services + execute regular Description Logic or rule-based reasoner

Reasoning with instance equality

- Reasoning - inference of new statements (facts) *follow from* set of known statements

- Every piece of additional knowledge excludes some unintended interpretation to the knowledge base

- infinite number of new statements could be inferred from any non-trivial knowledge base

- Not all, but knowledge base should contain all the important knowledge relevant to that task

Description Logic versus rule-based reasoners

- OWL DL was language is expressive, *decidable,* completeness to create *efficient reasoners for it*

- Decidability - guarantee to find an answer in finite time
- Completeness - guarantee to find all the complete answers

- Complexity of OWL DL are theoretical, concern worst-case complexity

- In practice, only average case matters as worst cases are rare

Contd...

- Description Logic reasoners primary tasks of classification, consistency checking of ontologies
- Other reasoning tasks are reformulated as consistency checking
- To check equality, check for inconsistency
- Inefficient to perform consistency check for every pair of instances in the ontology
- Alternative - Rule-based reasoning

Forward versus backward chaining

- Rules used either in forward-chaining or backward-chaining manner with different trade-offs

- Forward chaining - all consequences of the rules are computed to obtain a *complete materialization of the knowledge base*

- Done by repeatedly checking prerequisites of rules and adding their conclusions until no new statements can be inferred

- Advantage – Queries are faster, Disadvantage – takes huge space

Contd...

- Backward-chaining, rules executed on demand, i.e. when queries needs to be answered
- given a conclusion and check whether it is explicitly stated or whether it could inferred from some rule
- Drawback - longer query execution times
- Rule-based axiomatization advantage – reasoning fine-tuned by removing rules that only infer knowledge and irrelevant to reasoning task
- Applied in forward chain can save significant amounts of space

Contd...

- Ex. Flink System use built-in inference engine of the ontology store
+ Java based identity reasoner

- Reasoning is performed combining forward- and backward chaining to balance efficiency and scalability

Variations of Identity Reasoning

- 3 basic variations on what point the identity reasoning performed

- (Smushing is carried out while adding data into repository, if new instance already exists)

1. Descriptions merged: only one identifiers is kept, all information of resource consolidated under that identifier
    1. Disadvantage: impossible to unmerge descriptions using *owl:sameAs relationship* later
2. Reasoning performed after added to repository for duplicates using *owl:sameAs relationships*
    1. Disadvantage: removing statements is an expensive operation
3. Aggregating data dynamically done at query time such as Ajax

Evaluating smushing

- Smushing considered as retrieval problem or a clustering problem
- Retrieval - try to achieve a maximum precision and recall for correct set of mapping
- Conceptualize as clustering - single resource mapped to a number of other resources
- evaluate clusters against the ideal clustering
- smushing considered as optimization task - optimize an information retrieval or clustering-based measure

Contd...

- Instance unification (smushing) is “easier” than ontology mapping
- Ontology mapping chances for two instances ends up in a local minimum
- Some cases have both positive and negative rules – ends up inconsistent state
- Measuring the success of retrieval or clustering is application dependent

Advanced Representations

- Missing features in future versions – Ex. *owl:ReflexiveProperty*

- More expressive Rule Interchange Format (RIF) include FOL is under development

- Additional expressive power using temporal logic (statement true over certain time or interval)

- Though ontology should be stable, still dynamism desirable at some instances

- Extending logic with probabilities for accurate equivalence checking