Context Dependent Reasoning for Semantic Documents in Sindice

Renaud Delbru and Axel Polleres and Giovanni Tummarello and Stefan Decker
Motivations

- Sindice Semantic Web Index
  - + 30 million of documents
- Reasoning to find documents
  - Materialise implicit knowledge: IFPs, membership (sc, sp)
    - find someone called "Giovanni Tummarello" ignoring the wording:
      
      ```
      (* <http://xmlns.com/foaf/0.1/givenname> "Giovanni" AND
      * <http://xmlns.com/foaf/0.1/family_name> "Tummarello")
      OR * <http://xmlns.com/foaf/0.1/name> "Giovanni Tummarello"
      ```
  - Goal: Increase Precision/Recall (also find implicit information)
- But
  - Deal with real-world web data (heterogeneous, messy)
  - Computationally expensive (slow down indexing process)

→ Efficient&effective reasoning methodology required
Caching Ontologies

- **Naive approach:**
  - Cache all fetched ontologies + RDF data in one triple store
  - Compute and cache deductive closure

- **Problem:**
  - Leads to inappropriate deductive closure (too much)
  - Ontology is meant to be shared and reused
  - Diverging reuse reflects diverging points of view
    \[\rightarrow\] divergent semantics

- **Example:**
  - MY ontology can redefine `foaf:name`, e.g. as IFP
    - May lead to owl:sameAs inferences
    - Valid in the **context** of MY RDF graphs, but not for everybody
Context-Dependent Reasoning:
- Ensure context is preserved when aggregating documents
- “Quarantined Reasoning” approach:
  - Confine inference results to their context
  - Inferred axioms are invalid outside their context

Partition the Web of Data into smaller contexts (on a “per document” basis) ...
... and aggregate contexts based on dependencies

Prevents undesirable results ...
... while preserving intended meaning of the document
Reasoning over Linked Data
Reasoning over Linked Data
Reasoning over Linked Data

[Diagram showing relationships between ontologies and classes]
Reasoning over Linked Data

- Document taken alone: no semantics
- Recursive fetching of ontologies is mandatory
- Make use of
  - Explicit owl:imports
  - Implicit imports “by namespace” – make use of W3C best practices where possible.
- Intensive data processing
  - Data fetching, pre-processing
  - Deductive closure computing
Context on the Semantic Web

- Based on Guha's ideas on a context mechanism
- Context = Scope of validity of a statement

Aggregate context
- Composed by the content lifted from other contexts
- Contains specification of what it imports
- RDF document = aggregate context (as we will see later)

Lifting rules
- Expressive formulas
- Enable to lift axioms from one context to another
- At the moment, we only use the simplest lifting rule (simple import):

\[ \text{ist}(c_2, p) \land \text{ist}(c_1, \text{importsFrom}(c_1, c_2)) \rightarrow \text{ist}(c_1, p) \]
Import closure of Documents

- Explicit import
  - owl:imports primitive
  - Transitive: if $O_A$ imports $O_B$ and $O_B$ imports $O_C$, then $O_A$ imports $O_C$
  - When reasoning on an ontology $O$, one should consider the entire import closure of $O$.

- But, it is not a common practice
  - Only 5.56 thousand over 30 million of documents use owl:imports
Import closure of Documents

- **Implicit import**
  - Based on W3C best practices – Linked Data Principles
  - By dereferencing class or property URI

```
:me rdf:type foaf:Person .
:me foaf:name "Renaud Delbru" .


http://www.w3.org/1999/02/22-rdf-syntax-ns

```
owl:imports primitive and implicit imports
- mapped to Guha's importsFrom lifting rule
- See Definition 1

Cyclic import relations may occur:
- if O_A imports O_B and O_B imports O_A, then O_A ⇔ O_B
- Extend Guha's definition to allow cycles
- See Definition 2
Deductive closure of Documents

Reminder: aggregate context =
- Document content
- + ontology import closure (explicit and implicit imports)

Deductive closure of an aggregate context
- Computes full materialisation of aggregate context
- Original content + inferred statements

Inference based on a finite entailment regime
- Rule-based inference engine
- ter Horst’s pD* fragment (RDFS + subset of OWL)
Deductive closure of Documents

- Deductive closure of aggregate context
  - Lead to inferred statements that are not true in any of the source contexts alone
  - See *Definition 3*

Context C1:
```
:me rdf:type foaf:Person .
```

Context C2:
```
foaf:Person rdfs:subClassOf yago:Human .
```

\[ \Delta_{C1, C2} = \]
```
:me rdf:type yago:Human .
```

\[ \wedge \]
Ontology Base: Conceptual Model

- **Ontology Base**
  - Persistent TBox
  - Materialise import relations between ontology
  - Store inference results that has been performed

- **Concepts**
  - **Ontology entity:**
    rdfs:Property or rdfs:Class identified by a resolvable URI
  - **Ontology context:**
    Named graph composed by ontology statements
  - **Ontology network:**
    directed graph of ontology contexts where edges are import relations (see Definition 4)
1. Import closure of Doc1 is materialised
1. Import closure of Doc1 is materialised
2. Compute deductive closure of aggregate context $O_A, O_B, O_C$
Ontology Base: Update Strategy

1. Import closure of Doc1 is materialised
2. Compute deductive closure of aggregate context $O_A$, $O_B$, $O_C$
3. Store $\Delta_{A,B,C}$ in a separate named graph
A new document is coming, importing only $O_A$ and $O_C$:
1. Compute deductive closure of $O_A$ and $O_C$
A new document is coming, importing only $O_A$ and $O_C$:
1. Compute deductive closure of $O_A$ and $O_C$
2. Store $\Delta_{A,C}$ in a separate named graph
A new document is coming, importing only $O_A$ and $O_C$:
1. Compute deductive closure of $O_A$ and $O_C$
2. Store $\Delta_{A,C}$ in a separate named graph
3. Update deductive closure of $O_A$, $O_B$, $O_C$ so that the inferred triples are never duplicated
   a) Subtract $\Delta_{A,C}$ from $\Delta_{A,B,C}$
   b) add inclusion relation
      i.e., $\Delta_{A,B,C} := \Delta_{A,B,C} - \Delta_{A,C} + \Delta_{A,C}\text{owl:imports}\Delta_{A,B,C}$
Ontology Base: Querying Strategy

1. A document imports $O_A$ and $O_B$
Ontology Base: Querying Strategy

1. A document imports $O_A$ and $O_B$
2. Import closure is derived, and corresponding ontology network activated
1. A document imports $O_A$ and $O_B$
2. Import closure is derived, and corresponding ontology network activated
3. The related $\Delta_{A,B,C}$ is derived and activated
Ontology Base: Querying Strategy

1. A document imports $O_A$ and $O_B$
2. Import closure is derived, and corresponding ontology network activated
3. The related $\Delta_{A,B,C}$ is derived and activated
4. It is then found that $\Delta_{A,B,C}$ includes $\Delta_{A,C}$ which is also activated

→ Our Observation: “caching” Tbox inferences makes indexing (mostly ABox) much faster
Prototype and Preliminary Results

- ** Prototype implementation**
  - Distributed architecture based on Apache Hadoop
    - Hadoop “worker” (map-job):
      - reasoning agent (processing one document at a time)
  - Single ontology base shared among “workers”
    - Ontology base: context aware reasoning SAIL (Aduna Sesame)
      - Receives sets of URIs = aggregate contexts as “queries”

- ** Experimental setup**
  - Cluster of 3 nodes (á 4 cores 2.33GHz, 8GB)
  - 4 Hadoop workers / node
  - No syncing yet done between nodes

- ** Preliminary Results**
  - 40 documents / second on average;
  - up to 80 documents / second for simple datasets (Geonames)
  - Original size: 18GB - 46GB after inference (ratio of 2.5)
Discussions

- **Known problems**
  - Changing ontologies
  - Possibility to hijack our system:
    - Let \( d_1 \) and \( d_2 \) be ABox documents,
    - Observe: if \( d_1 \) refers to \( d_2 \) as an ontology entity, e.g.
      \[
      \text{rdfs:subClassOf} \quad \text{d1} \quad \text{d2}.
      \]
    - An attacker, could query indexed documents and then create a “fake” document making all indexed documents “look like” ontologies.

- **Solutions:**
  - Add Metadata on the ontology level (last update, etc.)
  - Fine-grained context (on a per-entity basis)
  - By analysing the content of \( d_2 \), we can detect that it does not contain any ontological statements about an entity \( d_2 \).
    \[ \rightarrow \text{The entity context d2 will not be added to the ontology base} \]
Conclusions

- We introduce a context-dependent inference methodology
  - Materialise implicit knowledge “per document”
  - Keep track of provenance of the inferred assertions
  - Inference based on Ter-Horst fragment
    (but other entailment regime possible)
- Context-dependent Inference Enables Sindice to
  - Be more effective in term of Precision/Recall
  - Avoid the deduction of undesirable assertions
  - Distribute & cache reasoning tasks on a per-document basis
- Future Work:
  - Analyse precise and average time and space complexity
  - Investigate lifting rules on ABox level (owl:sameAs)
  - Investigate fine-grained context (on a per-entity basis)