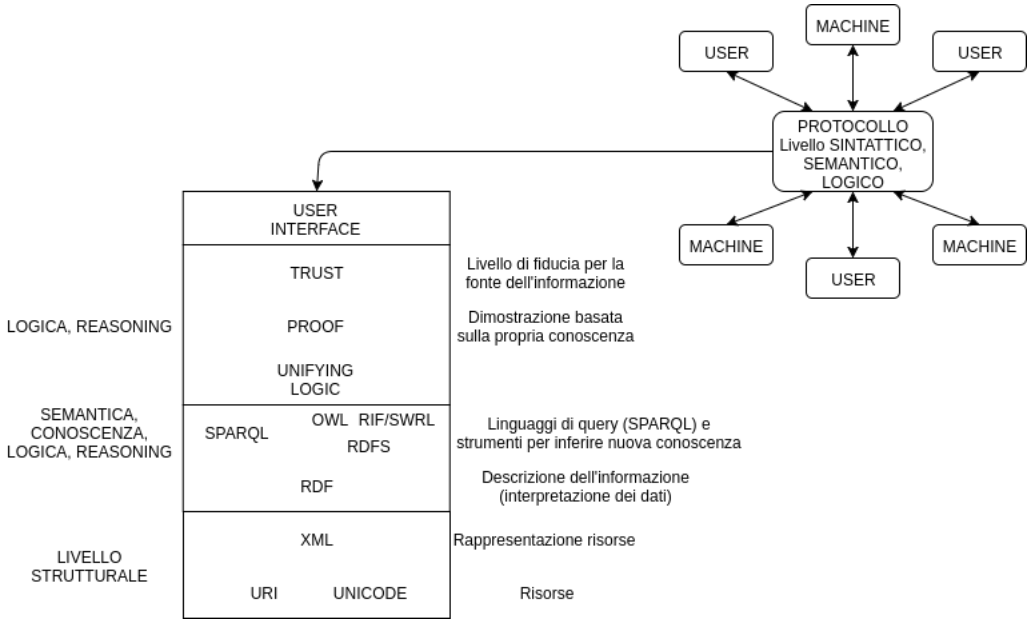
Semantic Web & Linked Open Data

1 Introduction

Il Web 2.0 con la sua struttura (risorse descritte tramite linguaggi, protocolli, ecc.) fornisce contenuto informativo accessibile solo ad esseri umani in quanto tali risorse sono espresse in linguaggio naturale. Introducendo il concetto di **interoperabilità** tra esseri umani e macchine nasce l'idea del **Semantic Web**, ovvero un'estensione dell'attuale Web per rendere l'informazione accessibile automaticamente anche alle macchine e garantire una mutua collaborazione tra queste ultime e gli esseri umani.
L'idea di base è quindi la condivisione della conoscenza che potrebbe avvenire usando un linguaggio che abbia una sintassi comune e quindi faciliti l'interoperabilità, che abbia una semantica condivisa, che permetta di esprimere le ontologie e infine che permetta elaborazioni efficienti, vista la dimensione del Web. Oltre al linguaggio si ha bisogno anche di un'infrastruttura che consenta di aggiungere metadati (annotazioni semantiche) e di referenziare in modo univoco le risorse presenti all'interno del Web.

1.1 Interoperabilità - Rappresentazione



Il protocollo di interoperabilità tra macchine e umani è strutturato su 3 livelli, dal basso verso l'alto nel diagramma:

- **Sintattico**: tutto ciò che comprende le risorse e la loro rappresentazione, per facilitarne l'accesso e lo scambio tra le entità (macchine e umani).
- **Semantico**: riguarda il mapping tra termini e concetti; il tutto deve avvenire in modo convenzionale e universale per facilitare l'interoperabilità. In questo livello ci sono linguaggi basati sulla logica del primo ordine, usati per definire le ontologie e i vocabolari RDF, oltre a quelli per interrogarli (SPARQL).

- **Logico**: in questo livello si posizionano gli strumenti (reasoner) per inferire nuova conoscenza dal livello semantico e operare su di essa.

2 RDF (Resource Description Framework)

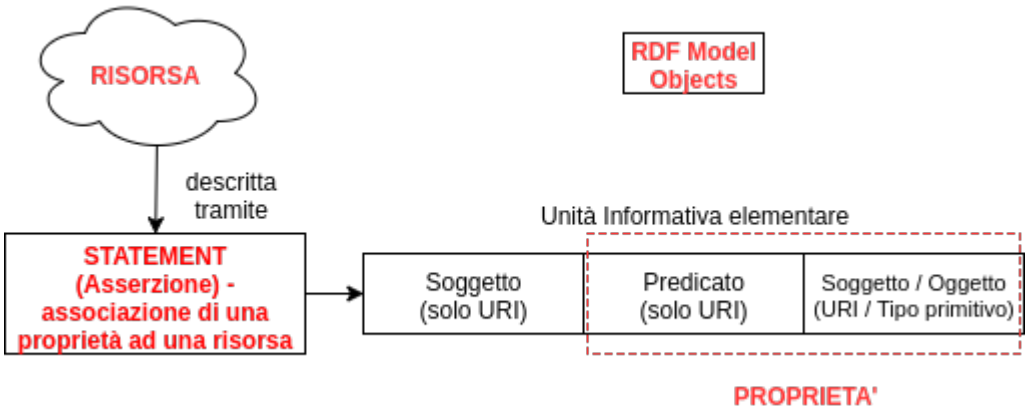
Key Idea: strumento per codifica, scambio e riutilizzo dei metadati sul web. E' stato introdotto per descrivere le **risorse** (metadati), dove per risorsa si intende tutto ciò che è identificabile tramite URI. Le componenti principali dell'RDF sono:

- RDF Data Model
- RDF Schema

2.1 RDF Data Model

Il Data Model RDF si basa su 3 principi fondamentali:

- qualunque cosa può essere identificata attraverso un URI;
- qualunque cosa deve essere descritta nel modo più semplice possibile;
- tutti possono dire tutto su tutti.



Per quanto riguarda la rappresentazione grafica degli oggetti del modello, indichiamo con un **ellisse** il soggetto e il predicato, e con un **rettangolo** il valore (literal).

2.1.1 RDF structures

Blank node: si usa per rappresentare una risorsa (soggetto o oggetto) non identificabile direttamente da un URI

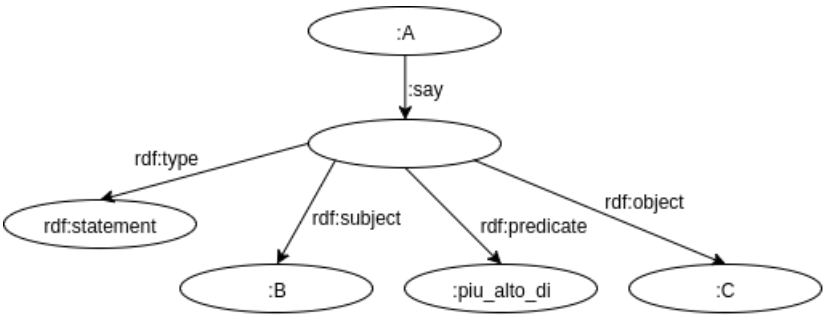
Contenitori: si utilizzano quando si vuole fare riferimento ad una serie di risorse che saranno il contenuto di una struttura oggetto dello statement. RDF prevede 3 tipi di strutture:

- Bag: insieme non ordinato con ripetizioni - si rappresenta collegando il Blank Node all'oggetto **rdf:bag** (o soggetto) tramite **rdf:type** e agli elementi presenti nella struttura ognuno labellato con rdf:1, rdf:2, ecc.
- Sequence: insieme ordinato con ripetizioni - si rappresenta in modo analogo alla bag ma tramite **rdf:seq**
- Alternative: insieme ordinato senza ripetizioni, nel quale si può scegliere solo un elemento - si rappresenta in modo analogo alla bag ma tramite **rdf:alt**

Collezioni: si usano quando si vuole indicare una lista strutturata contenente tutti e soli gli elementi della lista - una collezione si rappresenta con un soggetto iniziale collegato ad una lista di puntatori (blank node) uniti da **rdf:rest**; ogni puntatore tramite **rdf:first** è collegato all'effettiva URI di ogni oggetto della lista.

2.1.2 Reificazione

E' la procedura di riduzione ad oggetto dell'asserzione, usata nel caso in cui si vogliano rappresentare meta-informazioni (risorse) tramite altre meta-informazioni.
Esempio. "A dice che B è più alto di C" può essere vista come: "A (soggetto) dice che (proprietà) ..." e il resto della frase come uno statement a se stante collegato tramite un blank node alla proprietà. Il blank node identificherà l'asserzione tramite 4 elementi:



- tramite **rdf:type**, l'URI **rdf:statement** dice che si sta rappresentando un'asserzione;
- tramite **rdf:subject** si identifica il soggetto dello statement (solo URI);
- tramite **rdf:predicate** si specifica il predicato dell'asserzione (solo URI);
- tramite **rdf:object** si va a descrivere un oggetto, a sua volta soggetto di un altro statement (URI), oppure si assegna un valore "literal" (tipo primitivo).

2.2 RDF vocabulary and RDFS

Per la validazione e "realizzazione" di una semantica condivisa degli statements RDF, si necessita l'uso di vocabolari RDF. Un vocabolario RDF serve a definire, in un dominio applicativo, un insieme di proprietà che possiedono un **specifica semantica**. Utilizzando quindi un vocabolario standard, la semantica espressa dagli statements, potrà essere condivisa tra tutte le applicazioni che utilizzano quel vocabolario (es: se le applicazioni A e B utilizzano il vocabolario "Dublincore" per fare riferimento al creatore di un documento per una determinata risorsa, allora A e B "condividono" la semantica di "creatore del documento", ovvero "parlano della stessa cosa").
I vocabolari vengono definiti tramite il linguaggio RDFS, ovvero statments formati da classi e proprietà RDF che descrivono la struttura del vocabolario. RDFS rappresenta:

- Linguaggio ontologico light-weight e indipendente dal dominio.
- Schema interpretativo del documento e non strutturale.

2.2.1 RDF Schema Components

L'RDF Schema è formato principalmente da classi e proprietà, ed è così strutturato:

- L'entità principale è la risorsa, istanza della classe **rdfs:Resource**;
- **rdfs:Class** identifica che la risorsa è una classe, quindi un insieme di risorse, è definita ricorsivamente (es. uno statement potrebbe essere di tipo: rdfs:Class rdf:type rdfs:Class);

- **rdf:Property** è sottoclasse di rdfs:Resource e rappresenta la classe delle proprietà; un'istanza di questa classe lega le risorse soggetto alle risorse oggetto, in questo caso è detta predicato;
- **rdf:ConstraintProperty** è una "definizione/specializzazione" della classe rdf:Property e in particolare con **rdf:domain** si indica la classe del soggetto e con **rdf:range** la classe dell'oggetto legati da una determinata proprietà (es. rdf:impiegato rdfs:domain rdf:Person indica che la proprietà impiegato deve avere come soggetto una persona, e rdf:impiegato rdfs:range rdf:Organization indica che la stessa proprietà impiegato deve avere come oggetto un'azienda; quindi lo statement successivo sarà rdf:Mario rdfs:impiegato rdf:NomeAzienda);
- **rdfs:Literal** rappresenta la classe che contiene tutti i valori letterali, semplici o tipizzati.

come altre istanze della classe rdf:Property abbiamo:

- **rdf:type** usato per dire che una risorsa è istanza di una classe;
- **rdf:subClassOf** per indicare una sottoclasse;
- **rdf:subPropertyOf** per indicare una sottoproprietà;
- **rdf:seeAlso** indica che quella risorsa può fornire altre informazioni riguardanti quel soggetto;
- **rdf:label** fornisce una versione leggibile da utenti umani di una risorsa;
- **rdf:isDefinedBy** indica che una risorsa è soggetto di uno statement.

3 Linked Open Data for Content Based Recommender System

I CBRS presentano principalmente 2 grandi problemi/svantaggi:

- Dataset limitati per la rappresentazione del contenuto.
- Bassa rappresentazione semantica del contenuto (se basati su Bag of Word)

Per superare queste limitazioni, una delle possibili soluzioni consiste nell' utilizzo di **external knowledge sources**. Tra le varie tipologie: ontologie, folksonomie, semantica distribuzionale, conoscenza enciclopedica e LOD, quella da preferire risulta sicuramente la Linked Open Data cloud poichè rappresenta fonti di conoscenza disponibile su **larga scala** espresse/codificate tramite **semantica formale**; a differenza delle altre tipologie che presentano solo una di queste proprietà.

Linked Open Data A livello strutturale i LOD costituiscono un insieme di dataset interconnessi da relazioni semantiche. A livello operativo costituiscono una metodologia per la pubbicazione, condivisione, collegamento e accesso ai dati strutturati espressi tramite RDF/OWL presenti nel web. Sicuramente uno dei progetti più interessanti risulta DBpedia. DBpedia è un progetto crowd-sourced con lo scopo di estrarre in maniera automatica/semi-automatica informazione strutturataa da wikipedia e renderla disponibile nel web. Inoltre rappresenta uno dei nodi centrali della LOD essendo connessa ad altri Linked Dataset da circa 50mld di links.

Integrazione LOD in CBRS Facendo riferimento a DBpedia come knowledge dataset, 2 sono i principali approcci per l'integrazione dei LOD in un CBRS:

- **Direct Access**: ogni item è una risorsa di DBpedia.
- **Entity Linking**: estrazione automatica ed associazione di risorse DBpedia da testo non strutturato.

Ogni metodo presenta dei vantaggi/svantaggi:

Direct Access

- Pro's:
  - approccio lineare/chiaro (straightforward)
  - accesso tramite SPARQL endpoints
  - caratteristiche dei db relazionali grazie alla possibità di querying tramite SPARQL
- Con's:
  - Assunzione che l'item sia presente in DBpedia.
  - Le feature di un item sono definite manualmente, l'approccio risulta molto domain-dependent.
  - "Sfrutta" solo informazione strutturata.

Entity Linking

- Pro's:
  - sfrutta informazione non strutturata.
  - Eventuale introduzione di features inaspettate ma rilevanti (incremento novelty e serendipity)
  - caratteristiche dei db relazionali grazie alla possibità di querying tramite SPARQL
  - approccio generale e poco domain-dependent.
- Con's:
  - introduzione di "rumore" (risorse non rilevanti) dato dal processo di riconoscimento automatico delle entità e dai problemi di disambiguazione tipici del linguaggio naturale.