(Conceptual) Clustering methods for the Semantic Web: issues and applications

Nicola Fanizzi and Claudia d’Amato

Computer Science Department • University of Bari, Italy

Poznań, June 21th, 2011
Knowledge Representation and Learning Issues
Nicola Fanizzi
The **Semantic Web** is a (new) vision of the Web

[T. Berners-Lee at al. © Scientific American 2001](#)

Making the Web machine-interoperable (readable, understandable, …)

**How:**

- adding *meta-data* describing the content of Web resources
- share precise semantics for the meta-data using *ontologies*
Web Ontologies

- An **ontology** is a formal conceptualization of a domain that is *shared* and *reused* across domains, tasks and groups of people [A. Gomez Perez et al. 1999]

- **OWL**: standard representation language for web ontologies
  - supported by *Description Logics* (DLs)
  - endowed with by *well-founded semantics*
  - implemented through *reasoning services* (reasoners)
DLs: The Reference Representation

Basics vocabulary: \( \langle N_C, N_R, N_I \rangle \)
- **Primitive concepts** \( N_C = \{ C, D, \ldots \} \): subsets of a domain
- **Primitive roles** \( N_R = \{ R, S, \ldots \} \): binary rels on the domain
- **Individual names** \( N_I = \{ a, b, \ldots \} \) domain objects

*Interpretation* \( \mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I}) \) where:
- \( \Delta^\mathcal{I} \): *domain* of the interpretation and
- \( \cdot^\mathcal{I} \): *interpretation function* assigning *extensions*:
  - each concept \( C \) with \( C^\mathcal{I} \subseteq \Delta^\mathcal{I} \) and
  - each role \( R \) with \( R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I} \)

- The **Open World Assumption** made \( \Rightarrow \)
different conclusion w.r.t. DB closed-world semantics
### DLs: a family of languages

Principal DL concept/role construction operators (a language for each subset)

<table>
<thead>
<tr>
<th>Name</th>
<th>Syntax</th>
<th>Semantics</th>
</tr>
</thead>
<tbody>
<tr>
<td>atomic negation</td>
<td>$\neg A$</td>
<td>$A^T \subseteq \Delta^T$ ($A \in N_C$)</td>
</tr>
<tr>
<td>full negation</td>
<td>$\neg C$</td>
<td>$C^T \subseteq \Delta^T$</td>
</tr>
<tr>
<td>concept conj.</td>
<td>$C \cap D$</td>
<td>$C^T \cap D^T$</td>
</tr>
<tr>
<td>concept disj.</td>
<td>$C \cup D$</td>
<td>$C^T \cup D^T$</td>
</tr>
<tr>
<td>full exist. restr.</td>
<td>$\exists R.C$</td>
<td>${a \in \Delta^T \mid \exists b\ (a, b) \in R^T \wedge b \in C^T}$</td>
</tr>
<tr>
<td>universal restr.</td>
<td>$\forall R.C$</td>
<td>${a \in \Delta^T \mid \forall b\ (a, b) \in R^T \rightarrow b \in C^T}$</td>
</tr>
<tr>
<td>at most restr.</td>
<td>$\leq nR$</td>
<td>${a \in \Delta^T \mid {b \in \Delta^T \mid (a, b) \in R^T} \leq n$</td>
</tr>
<tr>
<td>at least restr.</td>
<td>$\geq nR$</td>
<td>${a \in \Delta^T \mid {b \in \Delta^T \mid (a, b) \in R^T} \geq n$</td>
</tr>
<tr>
<td>qual. at most restr.</td>
<td>$\leq nR.C$</td>
<td>${a \in \Delta^T \mid {b \in \Delta^T \mid (a, b) \in R^T \wedge b \in C^T} \leq n$</td>
</tr>
<tr>
<td>qual. at least restr.</td>
<td>$\geq nR.C$</td>
<td>${a \in \Delta^T \mid {b \in \Delta^T \mid (a, b) \in R^T \wedge b \in C^T} \geq n$</td>
</tr>
<tr>
<td>one of</td>
<td>${a_1, a_2, \ldots a_n}$</td>
<td>${a \in \Delta^T \mid a = a_i, 1 \leq i \leq n}$</td>
</tr>
<tr>
<td>has value</td>
<td>$\exists R{a}$</td>
<td>${b \in \Delta^T \mid (b, a^T) \in R^T}$</td>
</tr>
<tr>
<td>inverse of</td>
<td>$R^{-}$</td>
<td>${(a, b) \in \Delta^T \times \Delta^T \mid (b, a) \in R^T}$</td>
</tr>
</tbody>
</table>
Terminologies as Hierarchies – Subsumption

Concept Subsumption

Given two concept descriptions $C$ and $D$,

$$D \subseteq C$$

to be read $C$ subsumes $D$ (or $D$ is subsumed by $C$) iff for every interpretation $I$:

$$D^I \subseteq C^I$$

Equivalence: $C \equiv D$ iff $C \subseteq D$ and $D \subseteq C$

- It forms *hierarchies* of concepts
- It can be extended to roles
$$\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$$

- **TBox** $\mathcal{T}$ is a set of axioms
  
  $$C \equiv D \quad \text{(or} \quad C \subseteq D \text{)}$$

  where $C$ is a concept name and $D$ is a description

- **ABox** $\mathcal{A}$ contains extensional assertions on concepts or roles
  
  e.g. $C(a)$ and $R(a, b)$

  meaning, resp., that $a^\mathcal{I} \in C^\mathcal{I}$ and $(a^\mathcal{I}, b^\mathcal{I}) \in R^\mathcal{I}$

Interest in the **models** of $\mathcal{K}$:

interpretations $\mathcal{I}$ that satisfy all axioms/assertions in $\mathcal{K}$
TBox: Example

Primitive concepts:
\[ N_C = \{ \text{Female, Male, Human} \} \]

Primitive roles:
\[ N_R = \{ \text{hasChild, hasParent, hasGrandParent, hasUncle} \} \]

\[ T = \{ \text{Woman} \equiv \text{Human} \sqcap \text{Female}, \]
\[ \text{Man} \equiv \text{Human} \sqcap \text{Male}, \]
\[ \text{Parent} \equiv \text{Human} \sqcap \exists \text{hasChild}.\text{Human}, \]
\[ \text{Mother} \equiv \text{Woman} \sqcap \text{Parent}, \]
\[ \text{Father} \equiv \text{Man} \sqcap \text{Parent}, \]
\[ \text{Child} \equiv \text{Human} \sqcap \exists \text{hasParent}.\text{Parent}, \]
\[ \text{Grandparent} \equiv \text{Parent} \sqcap \exists \text{hasChild}.(\exists \text{hasChild}.\text{Human}), \]
\[ \text{Sibling} \equiv \text{Child} \sqcap \exists \text{hasParent}.(\exists \text{hasChild} \geq 2), \]
\[ \text{Niece} \equiv \text{Human} \sqcap \exists \text{hasGrandParent}.\text{Parent} \sqcup \exists \text{hasUncle}.\text{Uncle}, \]
\[ \text{Cousin} \equiv \text{Niece} \sqcap \exists \text{hasUncle}.(\exists \text{hasChild}.\text{Human}) \} \]
ABox: Example

\[ A = \{ \text{Woman(Claudia), Woman(Tiziana), Father(Leonardo), Father(Antonio),}
\]
\[ \text{Father(AntonioB), Mother(Maria), Mother(Giovanna), Child(Valentina),}
\]
\[ \text{Sibling(Martina), Sibling(Vito), hasParent(Claudia,Giovanna),}
\]
\[ \text{hasParent(Leonardo,AntonioB), hasParent(Martina,Maria),}
\]
\[ \text{hasParent(Giovanna,Antonio), hasParent(Vito,AntonioB),}
\]
\[ \text{hasParent(Tiziana,Giovanna), hasParent(Tiziana,Leonardo),}
\]
\[ \text{hasParent(Valentina,Maria), hasParent(Maria,ANTonio), hasSibling(Leonardo,Vito),}
\]
\[ \text{hasSibling(Martina,Valentina), hasSibling(Giovanna,Maria),}
\]
\[ \text{hasSibling(Vito,Leonardo), hasSibling(Tiziana,Claudia), hasSibling(Valentina,Martina),}
\]
\[ \text{hasChild(Leonardo,Tiziana), hasChild(Antonio,Giovanna), hasChild(Antonio,Maria),}
\]
\[ \text{hasChild(Giovanna,Tiziana), hasChild(Giovanna,Claudia),}
\]
\[ \text{hasChild(AntonioB,Leonardo), hasChild(Maria,Valentina),}
\]
\[ \text{hasUncle(Martina,Giovanna), hasUncle(Valentina,Giovanna) } \} \]
Inference Services

Besides standard inferences (satisfiability, inconsistency, subsumption checks):

- **instance checking** decide whether an individual is an instance of a concept \((\mathcal{K} \models C(a))\)

- **retrieval** find all individuals belonging to a given concept

- **least common subsumer** find the most specific concept that subsumes two (or more) given concepts

- **realization** find the concepts which an individual belongs to, esp. the most specific one: the most specific concept of a w.r.t. \(\mathcal{A}\) is \(C = \text{MSC}_\mathcal{A}(a)\), such that:
  1. \(\mathcal{K} \models C(a)\) and
  2. \(C \sqsubseteq D\), \(\forall D \mathcal{K} \models D(a)\).
**Clustering** discover groupings of domain objects

Many methods in the literature, e.g. optimize both
- intra-cluster *similarity* *(maximize)*
- inter-cluster *similarity* *(minimize)*

Many forms: hierarchical, probabilistic, fuzzy, etc. . . .

Different strategies: partitional, agglomerative
Issues with Multi-Relational Settings

In classical clustering settings:

- Data represented as feature vectors in an n-dimensional space
- Similarity can be defined algebraically (geometrically)
- The notion of centroid as a cluster representative often used

Issues with clustering individuals in knowledge bases:

- Individuals within KBs to be logically manipulated
- Similarity measure for DLs required
- An alternative cluster representative may be necessary, or, even better: a generalization procedure for producing intensional cluster descriptions (concepts/predicates)
DL Dissimilarity Measures

- Measures for comparing concepts
  - simple DL, allowing only disjunction [Borgida et al., 05]
  - *structural/semantic measures for ALC*
    - [d’Amato et al., 05] [d’Amato et al., 06]
  - *structural/semantic measures for ALCN'R* and *ALCHQ*
    - [Janowicz, 06] [Janowicz et al., 07]
  - semantic measure for *ALE(T)* [d’Amato et al., 07]

  All these *hardly scale* to more expressive DLs

- In Ontology Mining need for metrics for individuals
  - measures resort to the MSC approximations (not always available) for lifting individuals to the concept level
  - need for a *language-independent* measure
A Family of Semi-Distance Measures

- **IDEA**: *on a semantic level, similar individuals should behave similarly w.r.t. the same concepts*

- Inspired by [Sebag 1997]: individuals compared on the grounds of their behavior w.r.t. a set of *discriminating features*

\[ F = \{ F_1, F_2, \ldots, F_m \} \]

i.e. a collection of (primitive or defined) concept descriptions

- it may be found using stochastic search (GP)

- dependence only on *semantic* aspects related to the individuals

N. Fanizzi, C. d’Amato  
Clustering methods for the Semantic Web
A Family of Semi-Distance Measures – Definition

[Fanizzi et al. @ DL 2007] Given $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, let $\text{Ind}(\mathcal{A})$ be the set of the individuals in $\mathcal{A}$, $F = \{F_1, F_2, \ldots, F_m\}$, $p > 0$, and a weight vector $\vec{w}$, the family of semi-distance functions $d_p^F : \text{Ind}(\mathcal{A}) \times \text{Ind}(\mathcal{A}) \mapsto [0, 1]$ is defined:

$$d_p^F(a, b) := \frac{1}{m} \left[ \sum_{i=1}^{m} w_i \cdot | \pi_i(a) - \pi_i(b) |^p \right]^{1/p}$$

where $\forall i \in \{1, \ldots, m\}$ the projection function $\pi_i$ are defined:

$$\forall a \in \text{Ind}(\mathcal{A}) \quad \pi_i(a) = \begin{cases} 
1 & \mathcal{K} \models F_i(a) \quad (F_i(a) \in \mathcal{A}) \\
0 & \mathcal{K} \models \neg F_i(a) \quad (\neg F_i(a) \in \mathcal{A}) \\
pr & \text{otherwise}
\end{cases}$$

N. Fanizzi, C. d’Amato
Clustering methods for the Semantic Web
Conceptual Clustering

Performed during a *supervised learning phase* using the results of the *unsupervised clustering phase*:

**Problem Definition:**

- **Given**
  - individuals in a cluster $C$ regarded as *positive* examples of the concept to learn;
  - individuals in the others regarded as *negative* examples
  - $\mathcal{K}$ as background knowledge
- **Learn**
  - a definition $D$ in the *DL language of choice* so that
  - the individuals in the target cluster are instances of $D$ while those in the other clusters are not
Few algorithms for Conceptual Clustering (CC) with multi-relational representations [Stepp & Michalski, 86]
Fewer dealing with the SemWeb standard representations
- Kluster [Kietz & Morik, 94]
- CSKA [Fanizzi et al., 04]
  - Produce a flat output
  - Suffer from noise in the data

Idea: adopting a CC algorithm that combines
- a similarity-based clustering method ⇒ noise tolerant
- a DL concept learning method
  (YinYang, DL-Learner, DLFoil)
Clustering Methods and Applications
Claudia d’Amato
The notion of *medoid* (drawn from the PAM algorithm) rather than the notion of *centroid* (that is a weighted average of points in a cluster) is introduced.

A *medoid* is the central element in a group of individuals:

$$m = \text{medoid}(C) = \arg\min_{a \in C} \sum_{j=1}^{n} d_p(a, a_j) \text{ where } a \neq a_j$$
ECM: Evolutionary Clustering around Medoid

- Implements a genetic programming learning schema
- Search space made by
  - Genomes = strings (list) of medoids of variable length
    - Each gene stands as a prototypical for a cluster
- Performs a search in the space of possible clusterings of the individuals, by optimizing a fitness measure (for a Genome $G$)

$$\text{FITNESS}(G) = \left( \sqrt{k + 1} \sum_{i=1}^{k} \sum_{x \in C_i} d_p(x, m_i) \right)^{-1}$$

- On each generation those strings that are best w.r.t. the fitness function are selected for passing to the next generation.
ECM Algorithm: Main Idea

\[ \text{Ind}(\mathcal{A}) = \{a_1, a_2, \ldots, a_n\} \]

**Inizialize()**
- Random PopLength = k

**CurrentPopulation**
- \([G_1 = \{m_1, \ldots, m_{l_1}\}], \ [G_2 = \{m_1, \ldots, m_{l_2}\}], \ldots, \ [G_k = \{m_1, \ldots, m_{l_k}\}]\) \]

**generateOffSprings**(currentPopulation)

**offsprings**
- \([G'_1, G'_2, \ldots, G'_{l_1}], \ [G'_{21}, G'_{22}, \ldots, G'_{l_2}], \ldots, \ [G'_{k1}, G'_{k2}, \ldots, G'_{l_k}]\) \]

**computeFitness**(offsprings)

**fitness Vector**
- \([f(G'_{11}), f(G'_{12}), \ldots, f(G'_{1l_1}), f(G'_{21}), f(G'_{22}), \ldots, f(G'_{2l_2}), \ldots, f(G'_{l_1}), f(G'_{l_2}), \ldots, f(G'_{kl_k})]\) \]

**orderedFitnessVector**
- \([f(G'_{22}), f(G'_{11}), \ldots, \ldots]\) \]

**select**(offsprings, orderedFitnessVector)

**CurrentPopulation'** \(\rightarrow\) (ordered w.r.t. orderedFitnessVector)

**UNTIL**
- MaxGenerations
- EarlyStop(fitnessVector)

**RETURN** (currentPopulation'[0])
Running the ECM Algorithm...

```plaintext
medoidVector : ECM(maxGenerations, nGenOffsprings, nSelOffsprings)
output: medoidVector: list of medoids
static: offsprings: vector of generated offsprings, fitnessVector: ordered vector of fitness values, generationNo: generation number

currentPopulation = INITIALIZE()  generationNo = 0
repeat
  offsprings = GENERATEOFFSPRINGS(currentPopulation, nGenOffsprings)
  fitnessVector = COMPUTEFITNESS(offsprings)
  currentPopulation = SELECT(offsprings, fitnessVector, nSelOffsprings)
  ++generationNo
until (generationNo = maxGenerations OR EARLYSTOP(fitnessVector))
return SELECT(currentPopulation, fitnessVector, 1)  // fittest genome
```
Evolutionary Operators

offsprings: `GENERATEOFFSPRINGS(currentPopulation)`

**DELETION**($G$) drop a randomly selected medoid: $G := G \setminus \{m\}, m \in G$

**INSERTION**($G$) select $m \in \text{Ind}(\mathcal{A}) \setminus G$ that is added to $G$: $G := G \cup \{m\}$

**REPLACEMENTWITHNEIGHBOR**($G$) randomly select $m \in G$ and replace it with $m' \in \text{Ind}(\mathcal{A}) \setminus G$ s.t.

$$\forall m'' \in \text{Ind}(\mathcal{A}) \setminus G \quad d(m, m') \leq d(m, m'')$$

$G' := (G \setminus \{m\}) \cup \{m'\}$

**CROSSOVER**($G_A, G_B$) select subsets $S_A \subset G_A$ and $S_B \subset G_B$ and exchange them between the genomes:

$G_A := (G_A \setminus S_A) \cup S_B$ and $G_B := (G_B \setminus S_B) \cup S_A$
ECM Algorithm: Discussion

- **The ECM algorithm** *the optimal number of cluster* reflecting the **data distribution**
  - the algorithm can be easily modified if the number of clusters is known thus reducing the search space

- **The ECM algorithm is** *grounded on the notion of medoid*

- **Medoids are more robust** in presence of outliers w.r.t. centroids that are weighted average of points in a cluster
  - The medoid is dictated by the location of predominant fraction of points inside a cluster

- An alternative partitional clustering method for DLs inspired to the k-Means algorithm [Fanizzi et al. @ ESWC 2008]
The DL-Link Algorithm

- Modified average-link algorithm
- *Clusters are always made by a single concept description given by the GCS of the child nodes* (Instead of Euclidean average)
- **Output:** *DL-Tree* where actual *elements to cluster are in the leaf nodes*, *inner nodes are intentional descriptions of the child nodes*

![DL-Link Algorithm Diagram]

N. Fanizzi, C. d’Amato
Clustering methods for the Semantic Web
Running DL-Link

[d’Amato et al. @ IJSC 2010]

\textbf{DL-Link(S)}

\textbf{input} $S = \{R_1, \ldots, R_n\}$ the set of available concept descriptions;

\textbf{output} $DL-Tree$: dendrogram of the clustering process

Let $C = \{C_1, \ldots, C_n\}$ be the set of initial clusters obtained by considering each $R_i$ in a single cluster $C_i$;

$DL-Tree = \{C_1, \ldots, C_n\}; \quad n := |C|$

\textbf{while} $n \neq 1$ \textbf{do}

\textbf{for} $i, j := 1$ \textbf{to} $n$

\hspace{1cm} Compute the similarity values $s_{ij}(C_i, C_j)$;

\hspace{1cm} Compute $(C_h, C_k) = \arg\max_{i,j} s_{ij}$

\hspace{1cm} Create $C_m = GCS(C_h, C_k)$ the intensional descr. of the new cluster;

\hspace{1cm} Set $C_m$ as parent node of $C_h$ and $C_k$ in $DL-Tree$;

\hspace{1cm} Insert $C_m$ in $C$ and remove $C_h$ and $C_k$ from $C$;

\hspace{1cm} $n := |C|$;

\textbf{return} $DL-Tree$;
DL-Link: Discussion

- The GCS is an approximation of the LCS of ALE(T) concept descriptions \[\text{Baader et al. 2004}\]
- Because of the use of the GCS, DL-Link clusters ALE(T) concept descriptions referring to an ALC TBox.
- Individuals can be clustered by preliminarily computing the MSC for each of them
- Alternative hierarchical clustering methods for DL representations \[\text{Fanizzi et al. @ IJSWIS 2008; Fanizzi et al. @ Information Systems Journal 2009}\]
How to learn concept definitions?

- For DLs that allow for (approximations of) the msc and lcs, (e.g. $\mathcal{ALC}$ or $\mathcal{ALE}$):
  - given a cluster $C_j$,
    - $\forall a_i \in C_j$ compute $M_i := msc(a_i)$ w.r.t. the ABox $\mathcal{A}$
    - let $MSCs_j := \{ M_i | a_i \in \text{node}_j \}$
  - $C_j$ intensional description $lcs(MSCs_j)$

- Alternatively
  - other algorithms for learning concept descriptions expressed in DLs may be employed ([Fanizzi et al.'08] [Iannone et al.'07] [Lehmann and Hitzler’07] [Fanizzi et al.'10])
In the real life, knowledge is generally changing over the time

- New instances are asserted
- New concepts are defined

Clustering methods can be used for automatically:

- learning novel concept definitions which are emerging from assertional knowledge (**Novelty Detection**)
- for detecting concepts that are evolving, for instance because their intentional definitions do not entirely describe their extensions (**Concept drift**)

N. Fanizzi, C. d’Amato

Clustering methods for the Semantic Web
Automated Concept Drift and Novelty Detection

[Fanizzi et al. © Information Systems Journal 2009]

1. All individuals of the KB of reference are clustered
2. When *new annotated individuals are made available* they have to be integrated in the clustering model
3. **Adopted Approach:** The new instances are considered to be a *candidate* cluster
   - An *evaluation* of it is performed in order to assess its nature
Evaluating the Candidate Cluster: Main Idea 1/2
Evaluating the Candidate Cluster: Main Idea 2/2

[Diagram showing a candidate cluster with sub-clusters labeled 'Drift' and 'Novelty', connected to a global decision boundary.]
Evaluating the Candidate Cluster

- Given the initial clustering model, a *global boundary* is computed for it
  - $\forall C_i \in \text{Model}, \text{decision boundary cluster} = \max_{a_j \in C_i} d(a_j, m_i)$ (or the average)
  - The average of the decision boundary clusters w.r.t. all clusters represent the *decision boundary model or global boundary* $d_{overall}$

- The decision boundary for the candidate cluster $\text{CandCluster}$ is computed $d_{candidate}$
- if $d_{candidate} \leq d_{overall}$ then $\text{CandCluster}$ is a *normal* cluster
  - integrate :
    $\forall a_i \in \text{CandCluster} \ a_i \rightarrow C_j \ s.t. \ d(a_i, m_j) = \min_{m_j} d(a_i, m_j)$
- else $\text{CandCluster}$ is a *Valid Candidate* for *Concept Drift* or *Novelty Detection*
Evaluating Concept Drift and Novelty Detection

- The **Global Cluster Medoid** is computed
  \[
  \overline{m} := \text{medoid}\left( \{ m_j \mid C_j \in \text{Model} \} \right)
  \]

- \[d_{\text{max}} := \max_{m_j \in \text{Model}} d(\overline{m}, m_j)\]

- If \[d(\overline{m}, m_{\text{CC}}) \leq d_{\text{max}}\] the CandCluster is a **Concept Drift**
  - CandCluster is **Merged** with the most similar cluster \(C_j \in \text{Model}\)

- If \[d(\overline{m}, m_{\text{CC}}) \geq d_{\text{max}}\] the CandCluster is a **Novel Concept**
  - CandCluster is **added** to the model
    - in case of a hierarchical approach the cluster is added at the level \(j\) where the most similar cluster is found
Efficient Resource Retrieval: Motivation 1

- Resource Retrieval is performed:
  - by matching a request $R$ with each provided resource description, in order to detect relevant ones
  - **Example:** “finding the low cost companies that fly from Bari to Cologne?”
    - the query is expressed as a concept description

- **Problem:** inefficient approach with growing number of available resources
- **Solution:** similarly to databases, *exploiting a tree-based index* for *DL resource specifications* to improve the retrieval efficiency
Overall Idea

The Idea

Currently

Provided Services

Match Test

O(n)

Request

Match Test

O(log n)

Provided Services
Example: “finding the low cost companies that fly from Bari to Cologne?”

- the query is expressed as a concept description
- resources are retrieved by performing concept retrieval

- **Concept retrieval** is performed by executing instance checking for each individual in the ontology
- for DL with qualified existential restriction (as the one supporting OWL-DL), instance checking suffers from an additional source of complexity which do not show up other inference services such as concept subsumption.
Solution: *decrease the complexity of semantic retrieval by using concept subsumption rather than instance checking*

1. compute, for each resource, its **most specific concept** (MSC)
2. **semantic retrieval**: checking for each MSC, if subsumption between the query concept and the **MSC** holds

- *For a large number of resources*, the naive approach of matching the query w.r.t. each specification becomes *highly inefficient*.

- **Solution**: similarly to databases, *exploiting a tree-based index* for **DL resource specifications**
Tree-based index: desired characteristics

- Each leaf node contains a provided resource description
- Each inner node is a generalization of its children nodes
- Nodes at the same level have to be (possibly) disjoint

The DL-TREE obtained as output of the DL-LINK algorithm can be exploited

[d’Amato et al. @ IJSC 2010]
Service Retrieval Exploiting Clustered Service Descriptions

- Checks for subsumption of an available resource description w.r.t. the request

\[ R = \text{Flight from.Cologne to.Bari} \]

Match Test \((R,C)\)

\[ C = \text{Flight \& Hotel} \]

\[ C_{11} = \text{Flight} \]

\[ C_{111} = \text{Flight from.Paris to.Bari} \]

\[ C_{112} = \text{Flight from.Cologne to.Bari} \]

\[ C_{12} = \text{Hotel} \]

Available Resources

- Once the concepts representing the retrieved resource descriptions are found, their instances (namely the actual resources) are collected to assess, via *instance checking* which of them are also instances of the request.
Automatic Ontology Refinement: Motivations

[d’Amato et al. @ SWJ 2010]

- Manual ontology refinement is a complex task, particularly for large ontologies.
- Conceptual clustering methods could be adopted to (semi-)automatize this task

**Strategy:**

1. Given a KB, individuals are clustered
2. A Description for each cluster is learnt
3. The new concepts are merged with the existing ontology by exploiting the subsuption relation
4. In this way the ontology is refined/enriched introducing a fine granularity level in the concept descriptions
Conclusions

Presented:
- issues in applying conceptual clustering methods to the standard SW representation
- some proposals for solving these problems
- exploitation of clustering methods for:
  - Automatically detecting concept drift and new emerging concept in an ontology
  - Improve the efficiency of the resource retrieval task
  - Automatically enriching/refining existing (and potentially large) ontologies

Ongoing work with Dr. A. Ławrinowicz:
- Clustering query answers for reducing the information overload
The End

The end!

Questions?

Nicola Fanizzi
fanizzi@di.uniba.it
Claudia d’Amato
claudia.damato@di.uniba.it