UNIVERSITY OF JYVÄSKYLÄ

Lecture 10: Ontology Alignment

TIES4520 Semantic Technologies for Developers
Autumn 2023

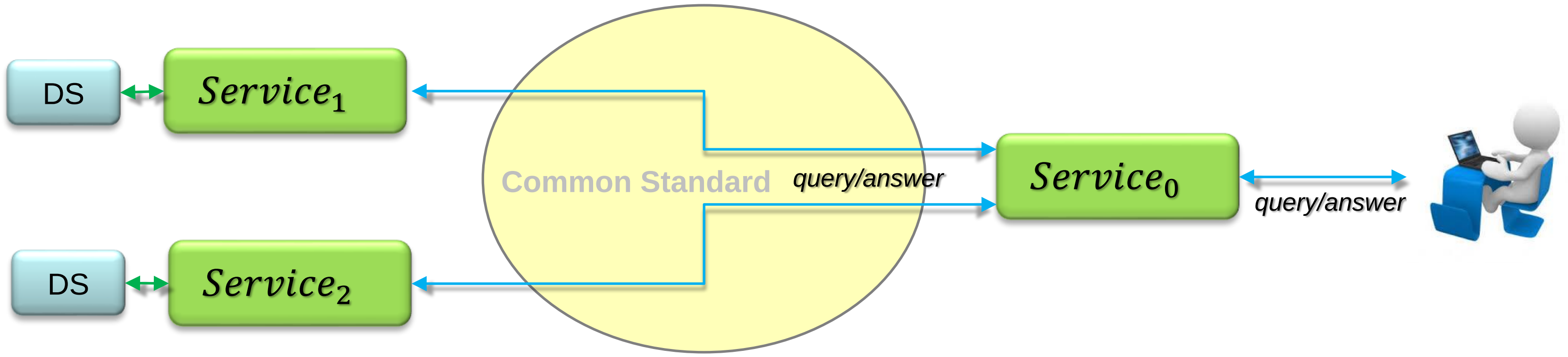
University of Jyväskylä

Khriyenko Oleksiy

Part 1

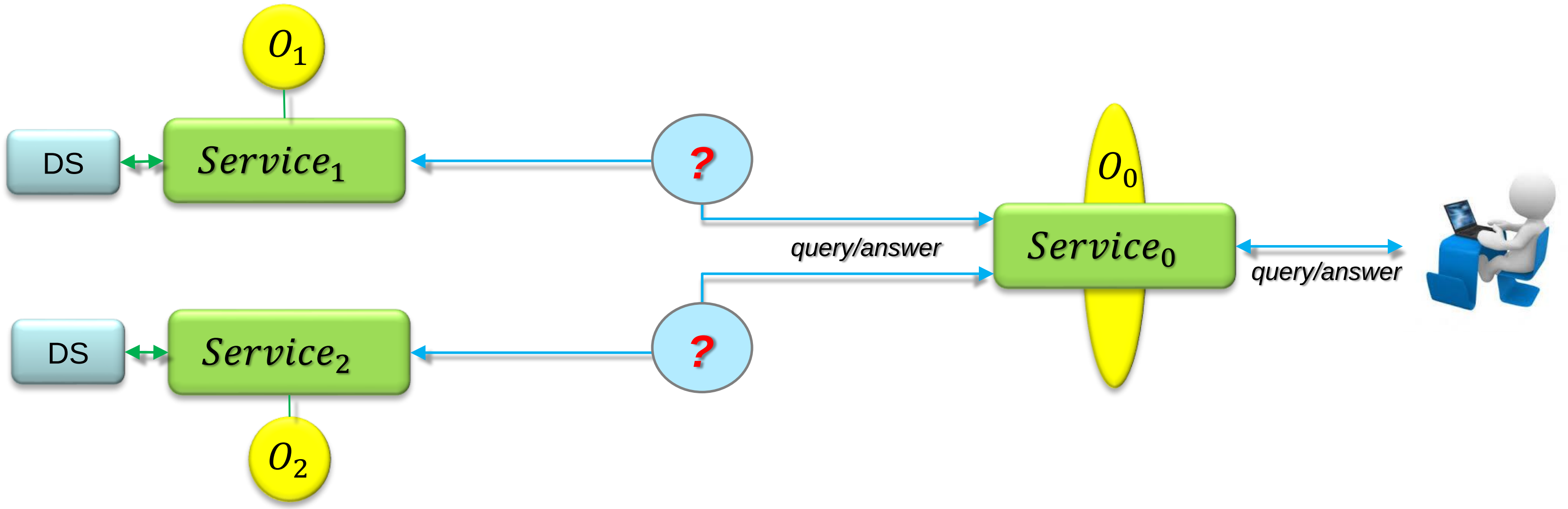
Ontology Alignment

- **Need for ontology alignment:**
  - *information integration (including schema integration, catalogue integration, data warehouses and data integration);*
  - *peer-to-peer information sharing;*
  - *web service composition;*
  - *autonomous communication systems (including agents and mobile devices communication) ;*
  - *navigation and query answering on the web.*



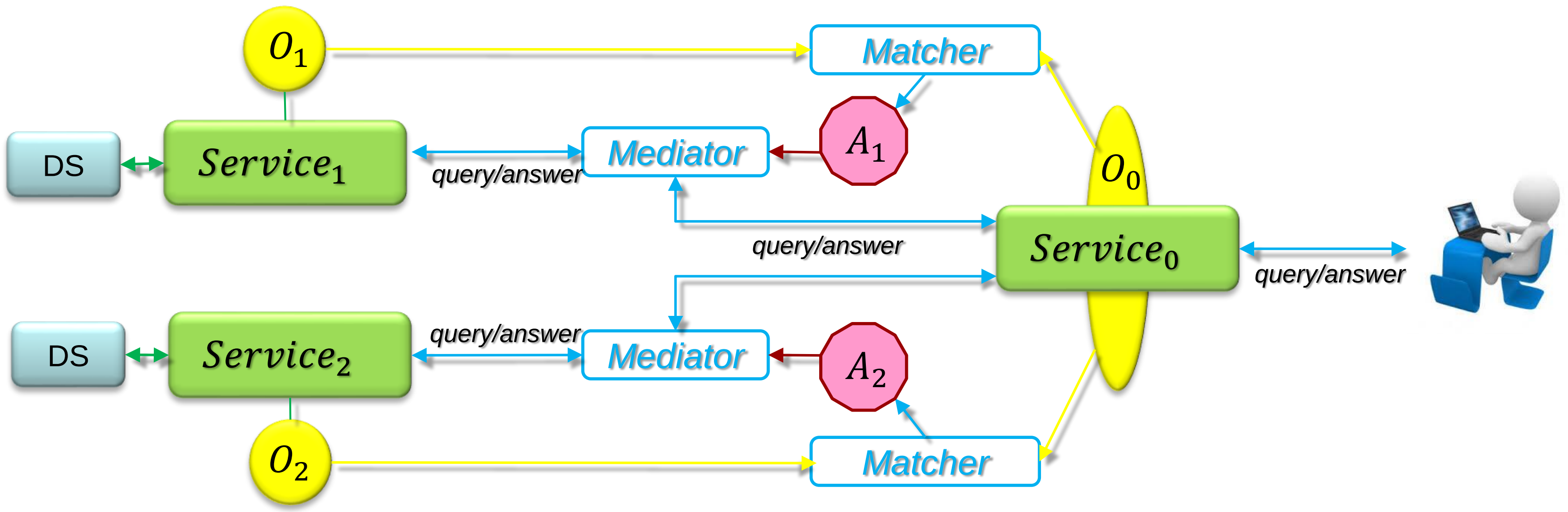
Ontology Alignment

- **Need for ontology alignment:**
  - *information integration (including schema integration, catalogue integration, data warehouses and data integration);*
  - *peer-to-peer information sharing;*
  - *web service composition;*
  - *autonomous communication systems (including agents and mobile devices communication) ;*
  - *navigation and query answering on the web.*



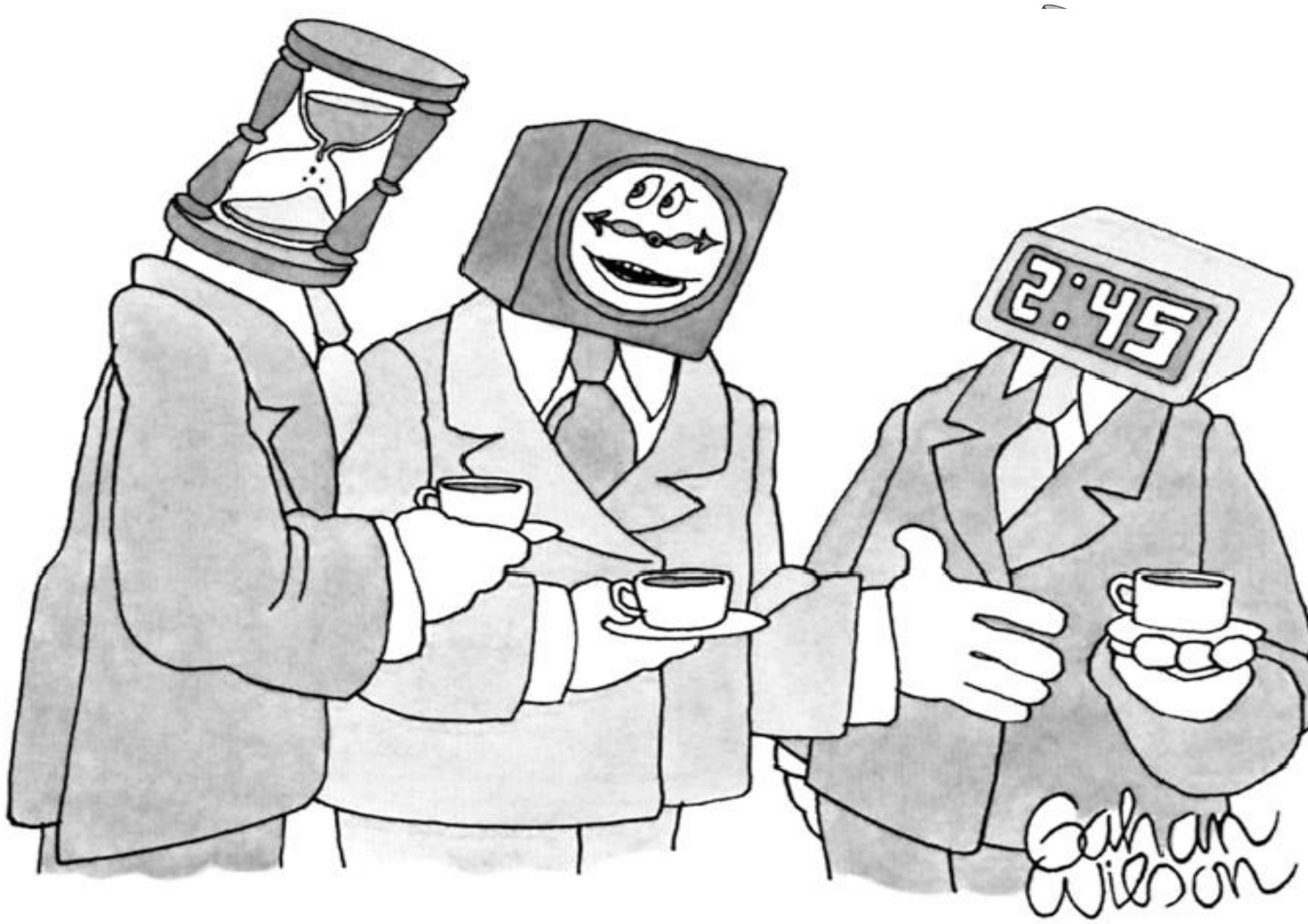
Ontology Alignment

- **Need for ontology alignment:**
  - *information integration (including schema integration, catalogue integration, data warehouses and data integration);*
  - *peer-to-peer information sharing;*
  - *web service composition;*
  - *autonomous communication systems (including agents and mobile devices communication) ;*
  - *navigation and query answering on the web.*



Ontology Alignment

- **Why ontologies are different?**
  - Ontology *is not a reality*, it is *a context-dependent projection* of it, its *subjective representation* of different designers;
  - Different tasks and requirements for applications;
  - Different conventions;
  - etc.
- **How ontologies are different?**
  - The same term is describing different concepts (Publication limited to peer-reviewed vs. Publication without any specific conditions);
  - Different terms are describing the same concept (Master-Student vs. MS.Student);
  - Different modeling conventions and paradigms (e.g., *intervals* vs. *points* to describe temporal aspects; Journal *as a class* vs. journal *as a property*; address *property broken up into several properties* vs. address *as a single string property*);
  - Different levels of granularity (e.g., *professor* vs. *adjunct*, *affiliated*, *associated* and *principal professors*);
  - Different coverage and points of view;
  - etc.



"*Basically, we're all trying to say the same thing.*"

Ontology Alignment

- **Ontology alignment** or **ontology matching**

  It is the process of determining *correspondences* between ontological concepts.

$$\langle id, e_1, e_2, r, n \rangle$$

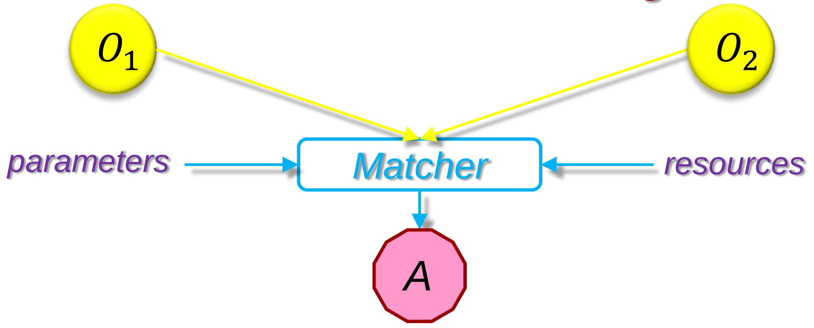
id – an identifier of the given correspondence;

e_1 *and* e_2 - entities, e.g., classes and properties of the correspondent ontologies;

r – a relation e.g., equal ($=$), equivalence ($\equiv$), more general ($\sqsupseteq$, $\geq$), less general ($\leq$, $\sqsubseteq$), disjointness ($\perp$), part-of or any user-specified relationship holding between entities;

n – a confidence measure (typically in the $[0,1]$ range) holding for the correspondence between e_1 and e_2.

A set of correspondences is also called an *alignment*.



$parameters$ – the matching parameters (weights or thresholds);

$resources$ – external resources such as domain specific thesauri, common knowledge, set of morphological rules, etc. (e.g. WordNet);

Ontology Alignment

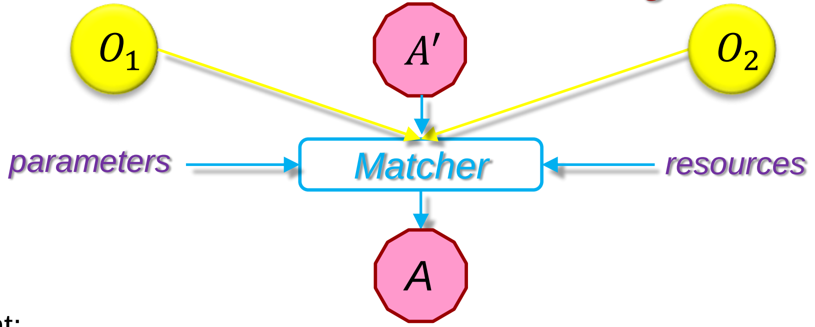
- **Ontology alignment** or **ontology matching**

  It is the process of determining *correspondences* between ontological concepts.

$$\langle id, e_1, e_2, r, n \rangle$$

id – an identifier of the given correspondence;
e_1 *and* e_2 - entities, e.g., classes and properties of the correspondent ontologies;
r – a relation e.g., equal ($=$), equivalence ($\equiv$), more general ($\sqsupseteq$, $\geq$), less general ($\leq$, $\sqsubseteq$), disjointness ($\perp$), part-of or any user-specified relationship holding between entities;
n – a confidence measure (typically in the $[0,1]$ range) holding for the correspondence between e_1 and e_2.

A set of correspondences is also called an *alignment*.



A' – an input alignment;
$parameters$ – the matching parameters (weights or thresholds);
$resources$ – external resources such as domain specific thesauri, common knowledge, set of morphological rules, etc. (e.g. WordNet);

Ontology Alignment

Different complexity level of correspondence

Simple:

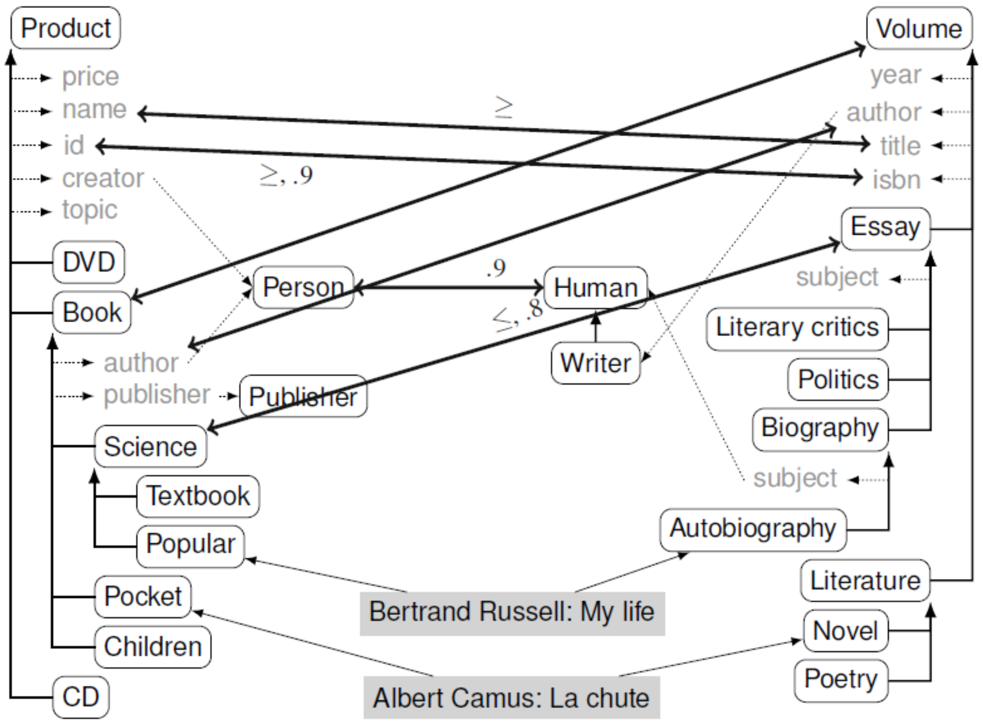
1. *www.ontologyEx.com/ex1.owl#Book* $=$ *www.ontologyEx.com/ex2.owl#Volume*
2. *speed* $\equiv$ *velocity*
3. *id* $\geq_{0.9}$ *isbn*

More complex (*transformation*):

1. *speed* $=$ *velocity* $*$ *2,237* - *(meters per second vs. miles per hour)*
2. $$autor(x, concat(w.firstname, w.lastname)) \underset{0.85}{\Longleftarrow} \begin{array}{l} Book(x) \\ \wedge\, writtenBy(x, w) \\ \wedge\, Writer(w) \end{array}$$

Ontology Alignment

- **Example of an alignment between two ontologies**
  - *Book* $=_{1.0}$ *Volume*
  - *id* $\geq_{0.9}$ *isbn*
  - *Person* $=_{0.9}$ *Human*
  - *name* $\geq_{1.0}$ *title*
  - *author* $=_{1.0}$ *author*
  - *Science* $\leq_{0.8}$ *Essay*



Picture is taken from [6]

Ontology Matching techniques

- **Element-level techniques** consider ontology entities or their instances in isolation from their relations with other entities or their instances.
  - *String-based techniques* are often used in order to match names and name descriptions of ontology entities. The more similar the strings, the more likely they are to denote the same concepts.
  - *Language-based techniques* consider names as words in some natural language, e.g., English. They are based on natural language processing techniques exploiting morphological properties of the input words. Usually, they are applied to names of entities before running *string-based* or *lexicon-based techniques* in order to improve their results.
  - *Constraint-based techniques* are algorithms which deal with the internal constraints being applied to the definitions of entities, such as types, cardinality (or multiplicity) of attributes, and keys.
  - *Linguistic resources* such as lexicons or domain specific thesauri are used in order to match words (in this case names of ontology entities are considered as words of a natural language) based on linguistic relations between them, e.g., synonyms, hyponyms.
  - *Alignment reuse techniques* represent an alternative way of exploiting external resources, which record alignments of previously matched ontologies.

Ontology Matching *(Element-level techniques)*

String-based techniques

Before comparing actual strings, which have a meaning in natural language, there are ***normalization procedures*** that can help improve the results of subsequent comparisons.

- *Case normalization* (example: *CD* becomes *cd*, *ISBN* becomes *isbn*).
- *Diacritics suppression* (for example: replacing *Montréal* with *Montreal*).
- *Blank normalization* consists of normalizing all blank characters, such as *blank*, *tabulation*, *carriage return*, or *sequences of these*, into a *single blank character*.
- *Link stripping* consists of normalizing some links between words, such as replacing *apostrophes* and *blank underline* into *dashes* or *blanks*.
- *Digit suppression* consists of suppressing digits (for example: *book24545-18* becomes *book*).
- *Punctuation elimination* suppresses punctuation signs (for example: *C.D.* becomes *CD*).

Warning!!!

- do not guarantee to bring the same results when applied in any order;
- can result in loosing some meaningful information (for example: *carbon-14* becomes *carbon*);
- may reduce variations, but increase synonyms (for example: in French *livre* and *livré* are different words respectively meaning *book* and *shipped*)

Ontology Matching *(Element-level techniques)*

String-based techniques

- *Hamming distance* (*dissimilarity function*):

$$\delta(s,t) = \frac{\left(\sum_{i=1}^{\min(|s|,|t|)} s[i] \neq t[i]\right) + \big||s| - |t|\big|}{\max(|s|,|t|)}.$$ *(Prefix)*

article and aricle: $\frac{(4)+|7-6|}{7} = \frac{5}{7} = 0,714$

$$\delta(s,t) = \frac{\left(\sum_{i=|s|,j=|t|}^{|s|-\min(|s|,|t|),|t|-\min(|s|,|t|)} s[i] \neq t[j]\right) + \big||s| - |t|\big|}{\max(|s|,|t|)}.$$ *(Suffix)*

article and aricle: $\frac{(2)+|7-6|}{7} = \frac{3}{7} = 0,429$

- *Substring similarity* (*similarity function*):

$$\sigma(s,t) = \frac{2 * |k|}{|s| + |t|}$$, where k is the longest common substring of s and t.

article and aricle: $\frac{2*4}{13} = \frac{8}{13} = 0,615$

particle and article: $\frac{2*7}{15} = \frac{14}{15} = 0,933$

- *N-gram similarity* (*similarity function*):

$$\sigma(s,t) = \frac{|ngram(s,n) \cap ngram(t,n)|}{max(|s|+|t|) - n + 1}$$, where $ngram(s,n)$ is the set of substrings of s of length n.

or

$$\sigma(s,t) = \frac{2 * |ngram(s,n) \cap ngram(t,n)|}{|ngram(s,n)| + |ngram(t,n)|}$$

ngram(article,3) = art, rti, tic, icl, cle.

ngram(aricle,3) = ari, ric, icl, cle.

ngram(particle,3) = par, art, rti, tic, icl, cle.

article and aricle: $\frac{2*2}{9} = \frac{4}{9} = 0,444$

particle and article: $\frac{2*5}{11} = \frac{10}{11} = 0,909$

Ontology Matching *(Element-level techniques)*

- **String-based techniques**
  - *Edit distance* (*dissimilarity function*) were designed for measuring similarity between strings that may contain spelling mistakes.

$$\delta(s,t) = \frac{\min_{(op_i)_I;\, op_n(\dots op_1(s))=t} \left(\sum_{i\in I} w_{op_i}\right)}{\max(|s|,|t|)}$$

, where w_{op_i} is a cost(weight) of transformation operation op_i (insertion of a character *ins*(*c*,*i*), replacement of a character by another *sub(c,c´,i)* and deletion of a character *del*(*c*,*i*));

Example: Let's make an assumption that w for *ins*(*c*,*i*) = *sub(c,c´,i)* = *del*(*c*,*i*) = 1

article and aricle: $\frac{w_{del("t")}}{7} = \frac{1}{7} = 0,143$

article and particle: $\frac{w_{ins("p")}}{8} = \frac{1}{8} = 0,125$

Ontology Matching *(Element-level techniques)*

String-based techniques

- *Jaro measure* (*similarity function*):

$$\boldsymbol{\sigma}(s,t) = \frac{1}{3} * \left(\frac{|com(s,t)|}{|s|} + \frac{|com(t,s)|}{|t|} + \frac{|com(s,t)| - |transp(s,t)|}{|com(s,t)|}\right)$$

, where $com(s,t)$ is a number of common characters between two strings, and $transp(s,t)$ is a number of elements of $com(s,t)$ which occur in a different order in s and t.

article and *aricle*: $\frac{1}{3} * (\frac{6}{7} + \frac{6}{6} + \frac{6}{6}) = 0,952$

particle and *article*: $\frac{1}{3} * (\frac{7}{8} + \frac{7}{7} + \frac{7}{7}) = 0,958$

- *Jaro–Winkler measure* (*similarity function*): is improved by favoring matches between strings with longer common prefixes.

$\boldsymbol{\sigma}(s,t) = \boldsymbol{\sigma}_{Jaro}(s,t) + P * Q * (1 - \boldsymbol{\sigma}_{Jaro}(s,t))$, where P is the length of the common prefix and Q is a constant (weight to the prefix).

In case $Q = 0,4$:

article and *aricle*: $0,952 + 2 * 0,4 * 0,048 = 0,990$

particle and *article*: $0,958 + 0 = 0,958$

Ontology Matching *(Element-level techniques)*

String-based techniques

Token-based distances (*similarity function*): usually work well on long texts (comprising many words). For that reason, it is helpful to take advantage of other strings that are attached to ontology entities.

- By aggregating different sources of strings: identifiers, labels, comments, documentation, etc.
- By splitting strings into independent tokens. For example, *InProceedings* becomes *In* and *Proceedings*, *peer-reviewed article* becomes *peer*, *reviewed* and *article*.

Apply:

- *Lemmatization* – leave the basic forms (for example: *kids* become *kid*)
- *Elimination* – discard “empty” tokens that are articles, prepositions, conjunctions… (e.g. *a*, *the*, *by*, *type of*, …)

Ontology Matching *(Element-level techniques)*

String-based techniques

- *Overlap metrics (Jaccard):*

$$\text{Jaccard}(S,T) = \frac{|S \cap T|}{|S \cup T|}$$

3-grams: article and aricle: {art,rti,tic,icl,cle} and {ari,ric,icl,cle} $\text{Jaccard} = \frac{2}{7} = 0{,}286$

3-grams: particle and article: {par,art,rti,tic,icl,cle} and {art,rti,tic,icl,cle} $\text{Jaccard} = \frac{5}{6} = 0{,}833$

2-grams: article and aricle: {ar,rt,ti,ic,cl,le} and {ar,ri,ic,cl,le} $\text{Jaccard} = \frac{4}{7} = 0{,}571$

2-grams: particle and article: {pa,ar,rt,ti,ic,cl,le} and {ar,rt,ti,ic,cl,le} $\text{Jaccard} = \frac{6}{7} = 0{,}857$

Problem:

- Str1 - “Apple Corporation, CA”
- Str2 - “IBM Corporation, CA”
- Str3 - “Apple Corp.”

Edit distance and *Jaccard* measure would match Str1 and Str2. To improve the measure, we need to recognize distinguishing term (e.g. “Apple”, not “Corporation” or “CA”).

Ontology Matching *(Element-level techniques)*

String-based techniques

- *Term frequency - Inverse document frequency* is usually not a measure of similarity: it assesses the relevance of a term (token).

All the strings should be converted into a bag of terms (document $\boldsymbol{d}$).

Example:

s1= "a b a c"	*d1={a,b,a,c}*
s2= "a c"	*d2={a,c}*
s3= " b a d"	*d3={b,a,d}*

- *Term frequency* $\mathbf{tf}(\boldsymbol{t},\boldsymbol{d})$: number of times t occurs in document $\boldsymbol{d}$ *(example:* $\boldsymbol{tf}(\text{"}\boldsymbol{a}\text{"},\boldsymbol{d1}) = \mathbf{2}$ *)*
- *Inverse document frequency* $\mathbf{idf}(\boldsymbol{t}) = \frac{|D|}{|\{d\in D; t\in d\}|}$: total number of documents in the collection D (set of documents) divided by the number of documents that contain t. *(example:*$\boldsymbol{idf}(\text{"}\boldsymbol{a}\text{"}) = \frac{\mathbf{3}}{\mathbf{3}} = \mathbf{1}$*)*

Each document is represented by feature vector $\overrightarrow{\mathbf{V_d}}$. Vector has feature $V_d(t)$ for each term t and the value is a function of $\mathbf{tf}$ and $\mathbf{idf}$ ($V_d(t) = tf(t,d) * idf(t)$). Amount of features are equal to T - the number of terms in the collection D.

	"a"	"b"	"c"	"d"
$\overrightarrow{\mathbf{V_{d1}}}$	**2** : 2*1	**1,5** : 1*3/2	**1,5** : 1*3/2	**0** : 0*3
$\overrightarrow{\mathbf{V_{d2}}}$	**1** : 1*1	**0** : 0*3/2	**1,5** : 1*3/2	**0** : 0*3
$\overrightarrow{\mathbf{V_{d3}}}$	**1** : 1*1	**1,5** : 1*3/2	**0** : 0*3/2	**3** : 1*3

Ontology Matching *(Element-level techniques)*

String-based techniques

- *Term frequency - Inverse document frequency*

An alternative score computation for dampening the **tf** and **idf** components by a log factor is:

$$V_d(t) = \log(tf(t,d) + 1) * \log(idf(t))$$

	"a"	"b"	"c"	"d"
$\overrightarrow{V_{d1}}$	0	0,053	0,053	0
$\overrightarrow{V_{d2}}$	0	0	0,053	0
$\overrightarrow{V_{d3}}$	0	0,053	0	0,143

,with normalization of $V_d(t)$: $V_d(t) = \frac{V_d(t)}{\sqrt{\sum_{t \in T} V_d(t)^2}}$

	"a"	"b"	"c"	"d"
$\overrightarrow{V_{d1}}$	0	0,706	0,706	0
$\overrightarrow{V_{d2}}$	0	0	1	0
$\overrightarrow{V_{d3}}$	0	0,346	0	0,935

- *Cosine similarity*

$$\sigma_T(d_1, d_2) = \frac{\sum_{t \in |T|} V_{d_1}(t) * V_{d_2}(t)}{\sqrt{\sum_{t \in |T|} V_{d_1}(t)^2} * \sqrt{\sum_{t \in |T|} V_{d_2}(t)^2}}$$

$\sigma_T(d_1, d_2) =$	0,707
$\sigma_T(d_1, d_3) =$	0,244
$\sigma_T(d_2, d_3) =$	0

Example:

s1= "a b a c"	*d1={a,b,a,c}*
s2= "a c"	*d2={a,c}*
s3= " b a d"	*d3={b,a,d}*

Ontology Matching *(Element-level techniques)*

String-based techniques

- *Soundex* *(Phonetic similarity measure):* match strings based on their sound. Specially effective in matching names (e.g., "Meyer" and "Meier") which are often spelled in different way but sound the same. Method maps word into a four-character code that captures the sound of it.

1. Keep the first letter of the word as the first letter of the code;
2. Remove all "W" and "H";
3. Replace all other letters with digits as follows:
   - Replace "B", "F", "P", "V" with **1**;
   - Replace "C", "G", "J", "K", "Q", "S", "X", "Z" with **2**;
   - Replace "D", "T" with **3**;
   - Replace "L" with **4**;
   - Replace "M", "N" with **5**;
   - Replace "R" with **6**.
4. Do not replace the vowels "A", "E", "I", "O", "U", "Y";
5. Replace sequence of identical digits with the digit itself;
6. Drop all the non-digit letters (except the first one) and return the first four letters as the *soundex* code (add **0** if there are not enough digits).

Example:

x = " *Meyer* "	*x* = " *Meier* "	*x* = " *Murphy* "
1.code = *Meyer*	*Meier*	*Murphy*
2.code = *Meyer*	*Meier*	*Murpy*
3.code = *Meye6*	*Meie6*	*Mu61y*
6.code = *M600*	*M600*	*M610*

- *Works well* for names from different origins
- *Doesn't work* well for Asian names, because the power of these names is based on vowels that are ignored by the code

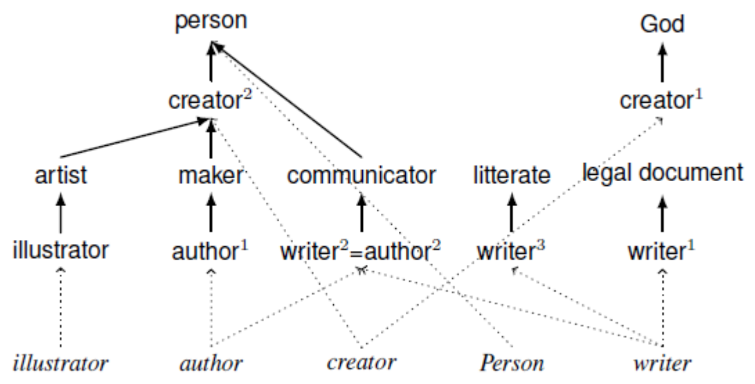
Ontology Matching *(Element-level techniques)*

Linguistic methods

Use of *WordNet* illustrates the use of external resources. ***WordNet*** is an electronic lexical database for English (see ***EuroWordNet*** for other languages). ***WordNet*** provides:

- *synsets* - sets of synonyms. $A = B$ if they are synonyms (ex: $Quantity = Amount$).
- *hypernym / holonym* (superconcept) structure. $A \sqsupseteq B$ if A is a hypernym of B (ex: $Europ \sqsupseteq Finland$).
- *hyponym / meronym* (part of) relations. $A \sqsubseteq B$ if A is a meronym of B (ex: $Brand \sqsubseteq Name$).
- *antonyms* or the siblings in the *part of* hierarchy. $A \perp B$ (ex: $Germany \perp Spain$).
- *textual descriptions* of the concepts (gloss) containing definitions and examples.

author^1 *noun*: Someone who originates or causes or initiates something; *Example* 'he was the generator of several complaints. ***Synonym*** generator, source. ***Hypernym*** maker. ***Hyponym*** coiner.
author^2 *noun*: Writes (books or stories or articles or the like) professionally (for pay). ***Synonym*** **writer**^{2}. ***Hypernym*** communicator. ***Hyponym*** abstractor, alliterator, authoress, biographer, coauthor, commentator, contributor, cyberpunk, drafter, dramatist, encyclopedist, essayist, folk writer, framer, gagman, ghostwriter, Gothic romancer, hack, journalist, libretist, lyricist, novelist, pamphleter, paragrapher, poet, polemist, rhymer, scriptwriter, space writer, speechwriter, tragedian, wordmonger, word-painter, wordsmith, Andersen, Assimov...
author^3 *verb.*: Be the author of; *Example* 'She authored this play'. ***Hypernym*** write. ***Hyponym*** co-author, ghost.



WordNet - http://wordnet.princeton.edu
EuroWordNet - http://www.illc.uva.nl/EuroWordNet/

Picture is taken from [6]

Ontology Matching *(Element-level techniques)*

Linguistic methods (*WordNet-based similarities*)

- *Synonymy similarity:*

$$\sigma(s,t) = \begin{cases} 1 & , if\ \Sigma(s) \cap \Sigma(t) \neq 0 \\ 0 & otherwise \end{cases}$$

, where $\Sigma(s)$ and $\Sigma(t)$ are sets of synsets associated with terms s and t.

- *Cosynonymy similarity (Jaccard metrics):* can show how close synonymous objects are.

$$\sigma(s,t) = \frac{|\Sigma(s) \cap \Sigma(t)|}{|\Sigma(s) \cup \Sigma(t)|}$$

, where $\Sigma(s)$ and $\Sigma(t)$ are sets of synsets associated with terms s and t.

- *Gloss overlap (Jaccard metrics):* used the following treatments before:
  - take gloss for all senses and add the term name;
  - suppress quotations (‘. . . ’);
  - suppress empty words (*or, and, the, a, an, for, of,* etc.);
  - suppress technical vocabulary, e.g., ‘term’;
  - suppress empty phrases, e.g., ‘usually including’;
  - etc.

Maltese dog (breed of toy dogs having a long straight silky white coat)
Afghan hound (tall graceful breed of hound with a long silky coat; native to the Near East)

$$\sigma(s,t) = \frac{|\lambda(s) \cap \lambda(t)|}{|\lambda(s) \cup \lambda(t)|}$$

, where $\lambda(s)$ and $\lambda(t)$ are glosses of the terms s and t.

Ontology Matching *(Element-level techniques)*

Internal structure and constraint-based techniques *[1]*

These methods exploit other information directly associated to the concept and are based on the internal structure of entities. *Internal similarity for classes* use such criteria as:

- data types (integer, float, string , data, etc.) of their properties and data type similarities (e.g., float and double are both real number representations);
- range of their properties (attributes and relations);
- cardinality or multiplicity, and the transitivity or symmetry of their properties;

to calculate the similarity between them.

These kinds of methods are commonly ***used to create correspondence clusters*** rather than to discover accurate correspondences between entities. They are usually combined with other element-level techniques.

- *Internal similarity for properties* is calculated as a weighted sum of the domain and range similarities.
- *Internal similarity for individuals* is calculated by comparing the values of their properties (datatype and object properties).

Ontology Matching

Extensional based techniques [1]

These methods utilize individual representations (or instances) of the classes. When two *ontologies share the same set of individuals*, matching is highly facilitated. The easiest way to compare classes when they share instances is to test the intersection of their instance set A and B:

- equal $(A \cap B = A = B)$,
- contains $(A \cap B = A)$,
- contained-in $(A \cap B = B)$,
- disjoint $(A \cap B = 0)$,
- overlap.

Hamming distance: $$\boldsymbol{\delta}(\boldsymbol{A}, \boldsymbol{B}) = \frac{|A \cup B - A \cap B|}{|A \cup B|}$$

Jaccard similarity: $$\boldsymbol{\sigma}(\boldsymbol{A}, \boldsymbol{B}) = \frac{|A \cap B|}{|A \cup B|}$$

- *External similarity between classes.* The extensional similarity measure for two classes is calculated in the same way as the children hierarchical similarity.
- *External similarity between properties.* To determine extensional similarity between properties, all individuals that contain a value for a given property are analyzed to determine a list of possible matches.

Ontology Matching techniques

- **Structure-level techniques** consider the ontology entities or their instances to compare their relations with other entities or their instances.
    - *Graph-based techniques* are graph algorithms which consider the input ontologies as labelled graphs. Usually, the similarity comparison is based on the analysis of nodes' positions within the graphs. The intuition behind this is that, if two nodes from two ontologies are similar, their neighbours must also be somehow similar.
    - *Taxonomy-based techniques* are also graph algorithms which consider only the specialization relation. The intuition behind taxonomic techniques is that *is-a* links connect terms that are already similar (being interpreted as a subset or superset of each other), therefore their neighbours may be also somehow similar.
    - *Model-based techniques* are semantically grounded algorithms handle the input based on its semantic interpretation, e.g., model-theoretic semantics. The intuition is that if two entities are the same, then they share the same interpretations. Examples are propositional satisfiability and description logics reasoning techniques.
    - *Instance-based techniques (data analysis and statistics)* take advantage of a (hopefully large) representative sample of a population in order to find regularities and discrepancies. This helps in grouping together items or computing distances between them. Among data analysis techniques we discuss distance-based classification, formal concepts analysis and correspondence analysis; among statistical analysis methods we consider frequency distributions.

Ontology Matching *(Structure-level techniques)*

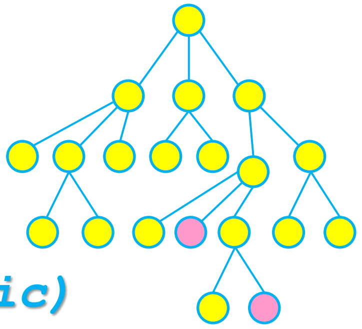
Graph-based techniques

- ***Shortest path*** (*Bouquet, Kuper, Scoz and Zanobini's metric* [2]) **:** is very simple metric that only considers the distance between the concepts in the ontology and does not take into account the level of generalization or specialization of both concepts.

  $\delta(c_1, c_2)$ - distance between simple concepts c_1 and c_2 defined as the shortest path between the concepts if the path exists, and 0 otherwise.

- ***Normalized distance:*** *Shortest path* normalized by the maximal length of a path between two classes in the taxonomy.

$$\overline{\delta}(c_1, c_2) = \frac{\delta(c_1, c_2)}{max_{c,c' \in O}\,\delta(c, c')}$$



- ***Upward cotopic similarity (Jaccard metric)***

$$\sigma(c_1, c_2) = \frac{|UC(c_1, H) \cap UC(c_2, H)|}{|UC(c_1, H) \cup UC(c_2, H)|}$$

, where $UC(c, H) = \{c' \in H; c \leq c'\}$ is the set of super-classes of c .

Ontology Matching *(Structure-level techniques)*

Graph-based techniques

- ***Scaled shortest path*** (*Leacock and Chodorow's metric [3]*) **:** is shortest path metric scaled by the depth of the ontology and with a logarithmic growth.
  Defined as:

$$\sigma(c_1, c_2) = max\left[-log\left(\frac{\delta(c_1, c_2)}{2D}\right)\right]$$

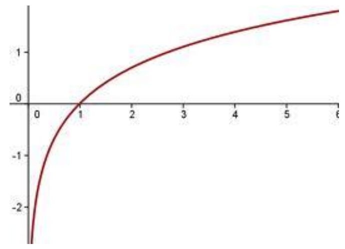
, where $\delta(c_1, c_2)$ is the shortest path length between concepts c_1 and c_2 and D is the maximum depth of the ontology that helps us to scale the similarity depending on ontology detailing.

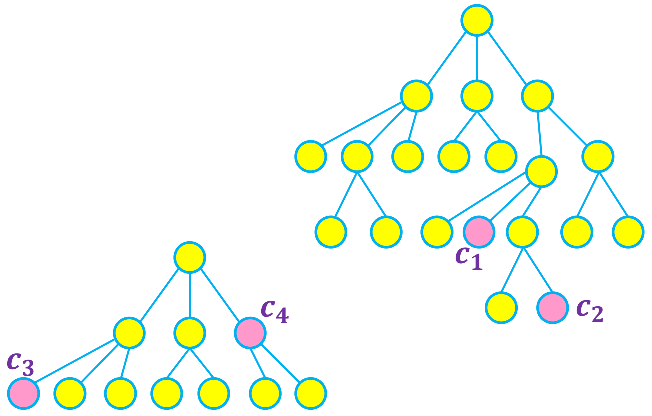
$\delta(c_1, c_2) = 3$
$\delta(c_3, c_4) = 3$

$$\sigma(c_1, c_2) = max\left[-log\left(\frac{3}{8}\right)\right]$$

$$\sigma(c_3, c_4) = max\left[-log\left(\frac{3}{4}\right)\right]$$

$$\sigma(c_1, c_2) > \sigma(c_3, c_4)$$





Ontology Matching *(Structure-level techniques)*

Graph-based techniques

- ***Depth of the subsumer and closeness to the concepts*** *(Wu and Palmer's metric [5] and Haase, Siebes, and van Harmelen's metric [6])* **:** is metric based on the depth of the concepts and depth of the lowest common subsumer of them.

$$\sigma(c_1,c_2)=\frac{2*depth(lcs(c_1,c_2))}{depth(c_1)+depth(c_2)}$$

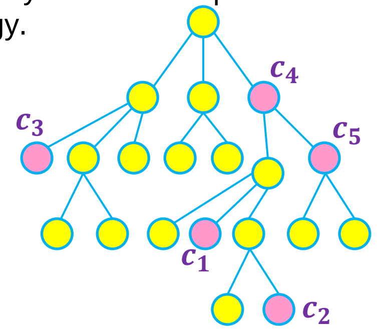
, where $lcs(c_1,c_2)$ is the *lowest common subsumer* between concepts c_1 and c_2. In other words, the first common concept in the paths from c_1 and c_2 to the root. $depth(c_n)$ is the distance from the concept c_n. So, the similarity of two concepts increases as their lcs is deeper in the ontology.

$$sim(c_1,c_2)=\frac{4}{7}=0,57$$

$$sim(c_2,c_5)=\frac{2}{6}=0,33$$

$$sim(c_3,c_4)=\frac{0}{3}=0$$

, in case lcs is also a the root node, the similarity between both concepts is 0. You may use ***Shortest path*** metric to compare the similarity of such pairs.



$$\sigma(c_1,c_2)=\begin{cases} e^{-\alpha l}\dfrac{e^{\beta h}-e^{-\beta h}}{e^{\beta h}+e^{-\beta h}}, & if\, c_1\neq c_2\\ 1 & otherwise\end{cases}$$

, where l is the length of the shortest path between concepts c_1 and c_2, h is depth of lcs in the tree, and $\alpha\geq 0$ and $\beta\geq 0$ are parameters that scale the contribution of l and h, respectively.

$$sim(c_1,c_2)=0,455$$

$$sim(c_2,c_5)=0,24$$

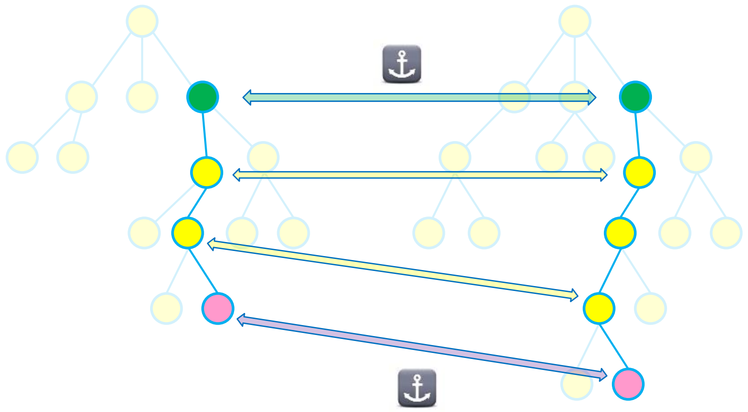
$$sim(c_3,c_4)=0$$

Than deeper lcs and shorter the distance between the concept, then similarity is bigger.

Ontology Matching *(Structure-level techniques)*

Taxonomy-based techniques

- *Bounded path matching:* take two paths with links between classes defined by the hierarchical relations, compare terms and their positions along these paths, and identify similar terms. *Anchor-Prompt* technique is primarily guided by *two anchors of paths*:



- *Super or subclass rules:* based on rules capturing the intuition that *classes are similar if their super or subclasses are similar*.

Ontology Matching *(Structure-level techniques)*

- **Model-based techniques**
  - *Description logic (LD) relation inference:*

Ontology 1: $Micro - company = Company \sqcap \leq_5 employee$ - *Micro-company* is a *Company* with at most 5 *employee*

Ontology 2: $SME = Firm \sqcap \leq_{10} associate$ - *SME* is a *Firm* with at most 10 *associates*.

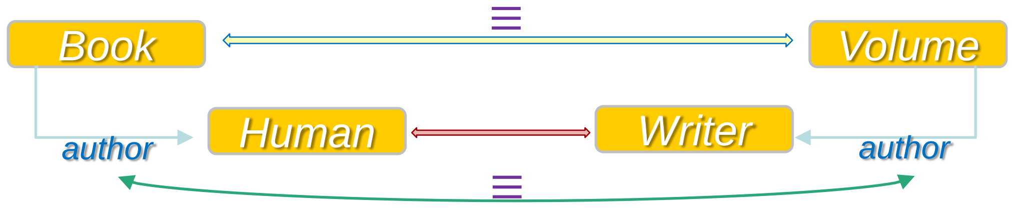
Initial alignment: $Company = Firm$
$associate \sqsubseteq employee$ - *Company* is equivalent to *Firm* and *associate* is a subclass of *employee*.

$Micro - company \sqsubseteq SME$ - *Micro-company* is a subclass of *SME*.

Ontology Matching *(Structure-level techniques)*

Relation-based techniques

- The similarity between nodes can also be based on their relations. If two classes from given ontologies are similar and are related to some other classes by similar relations (properties), then we can infer that those two target classes may be similar too.



- This can be applied to a set of classes and a set of relations. If we have a set of relations $r_1 \dots r_n$ in the first ontology which are similar to another set of relations $r'_1 \dots r'_n$ in the second ontology, it is possible that two classes, which are the domains of relations in those two sets, are similar too.

- This principle can also be extended to the composition of relations, i.e., instead of considering only the relations asserted at a class, one can consider their composition with relations starting at the domain of this relation. (for instance, one can consider the composition $\text{author} \rightarrow first\ name$).

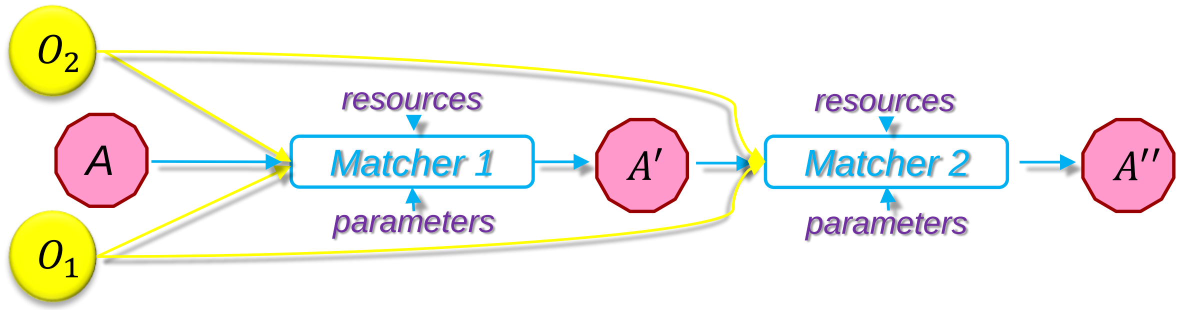
- ***Children.*** The similarity between nodes of the graph is computed based on similarity of their children nodes, that is, two non leaf entities are structurally similar if their immediate children sets are highly similar.

- ***Leaves.*** The similarity between nodes of the graphs is computed based on similarity of leaf nodes, that is, two non leaf schema elements are structurally similar if their leaf sets are highly similar, even if their immediate children are not.

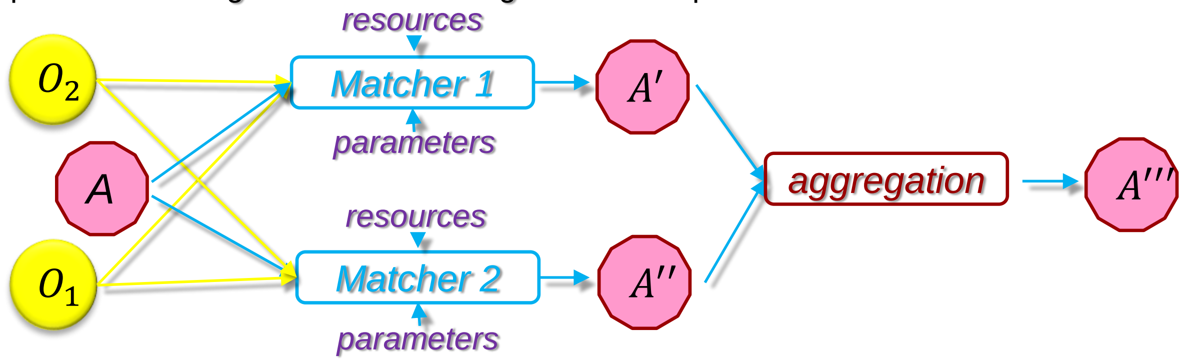
Ontology Alignment

Matcher composition

- *Sequential composition:* A natural way of composing the basic matchers consists of improving the matching through the use of sequential composition.



- *Parallel composition:* Another way to combine algorithms by running several different algorithms independently and aggregating their results. There are two main kinds of parallel composition: *heterogeneous* and *homogeneous* compositions.



Ontology Alignment

- **Aggregation operations** There are many different ways to aggregate matcher results, usually depending on confidence/similarity *[6]*:
  - *Triangular norms (min, weighted products)* useful for selecting only the best results.

    min: $\forall x, x' \in O, \delta(x, x') = \min_{i=1,n} \delta(x_i, x_i')$ *Weighted products:* $\forall x, x' \in O, \delta(x, x') = \prod_{i=1}^{n} \delta(x_i, x_i')^{w_i}$

  - *Multidimensional distances (Minkowski distance, weighted sum)* useful for taking into account all dimensions.

    Minkowski distance: $\forall x, x' \in O, \delta(x, x') = \sqrt[p]{\sum_{i=1}^{n} \delta(x_i, x_i')^p}$ *Weighted sum:* $\forall x, x' \in O, \delta(x, x') = \sum_{i=1}^{n} w_i * \delta(x_i, x_i')$

    Euclidean distance: $p = 2$
    Manhattan distance: $p = 1$
    Chebichev distance: $p = +\infty$

  - *Fuzzy aggregation (min, weighted average)* useful for aggregating the results of competing algorithms and taking advantage of all of them.

    Weighted average: $\forall x, x' \in O, \delta(x, x') = \frac{\sum_{i=1}^{n} w_i * \delta(x_i, x_i')}{\sum_{i=1}^{n} w_i}$

  - *Other specific measures* (e.g., ordered weighted average).

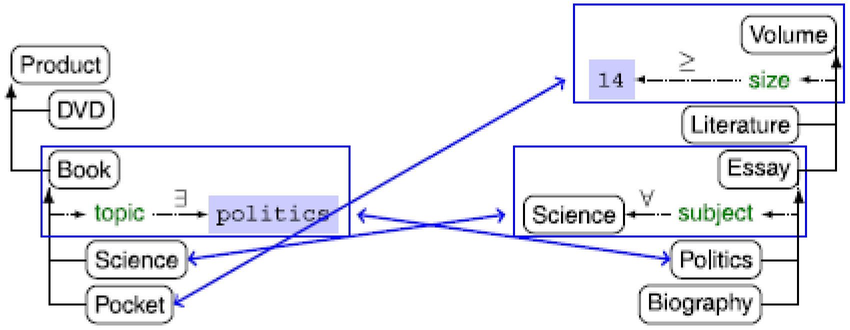
    Ordered weighted average: $f(x_1, \dots, x_n) = \sum_{i=1}^{n} w_i * x_i'$

    Where:
    - $w_i, \dots, w_n$ is a sat of weights in $[0\ 1]$ such that $\sum_{i=1}^{n} w_i = 1$;
    - x'_i is the i-th largest element of $(x_1, \dots, x_n)$.

Part 2

Ontology Alignment

Alignment formats



First-order logic:

$$\forall x, \quad Pocket(x) \equiv Volume(x) \land size(x,y) \land y \geq 14$$

$$\forall x, \quad Science(x) \equiv Essay(x) \land (\forall y, subject(x,y) \Longrightarrow Science(y))$$

$$\forall x, \quad Book(x) \land topic(x, politics) \equiv Politics(x)$$

$$\forall x, \quad Writer(x) \Longleftarrow \exists y, \quad author(y,x)$$

Picture is taken from [6]

Ontology Alignment

Alignment formats

- *MAFRA Semantic Bridge Ontology (SBO)* does not define a real exchange format for ontology alignment. Rather, it provides an ontology, called the *Semantic Bridge Ontology*. The instantiation of this ontology is an ontology mapping document.
  - The main concepts in this ontology are *SemanticBridges* and *Services*. A *SemanticBridge* is tied to the *Services* that are able to implement the bridge as a data transformation.
  - *Service* can be thought of as a function: $\mathrm{f}: Arg^n \longrightarrow Arg^m$ that maps tuples of arguments into tuples of arguments.
  - *SemanticBridges*, which in turn can be *ConceptBridges* or *PropertyBridges*, express a relation between two sets of entities by composing elementary services that are applied to them.

$$\boldsymbol{ConceptBridge}: \boldsymbol{Volume2Pocket}$$

$$\boldsymbol{x}: <\boldsymbol{o2} \# \boldsymbol{Volume}>; \boldsymbol{o2}: \boldsymbol{size} \geq \boldsymbol{14} \rightarrow <\boldsymbol{o1} \# \boldsymbol{Pocket}>$$

$$\boldsymbol{ConceptBridge}: \boldsymbol{Book2Politics}$$

$$\boldsymbol{x}: <\boldsymbol{o1} \# \boldsymbol{Book}>; \boldsymbol{o1}: \boldsymbol{topic} ==' \boldsymbol{politics}' \rightarrow <\boldsymbol{o2} \# \boldsymbol{Politics}>$$

But:

- The format provided by the Semantic Bridge Ontology is not very clear since the language is described in UML. This is a minor problem that could be solved by exposing some RDF/XML format.
- This format is a relatively complex language that is tied to the MAFRA architecture.
- It does not separate the declarative aspect of relations from the more operational aspects of services: relations are described with regard to services able to implement them.

Examples:

- One of the implementation examples of SBO approach using OWL *[8]*

Ontology Alignment

Alignment formats

- *OWL* can be considered as a language for expressing correspondences between ontologies.
  - The *equivalentClass* and *equivalentProperty*, *subClassOf* and *subPropertyOf* primitives have been introduced for relating elements and entities.

```
…
<owl:Property rdf:about="&onto1;#author">
    <owl:equivalentProperty rdf:resource="&onto2;#author"/>
</owl:Property>
<owl:Class rdf:about="&onto1;#Book">
    <owl:equivalentClass rdf:resource="&onto2;#Volume"/>
</owl:Class>
<owl:Property rdf:about="&onto2;#title">
    <owl:subProperty ="&onto1;#name"/>
</owl:Property>
```

```
…
<owl:Class rdf:ID="&onto1;#Science">
    <owl:equivalentClass>
        <owl:Class>
            <owl:subClassOf rdf:resource="&onto2;#Essay"/>
            <owl:subClassOf>
                <owl:Restriction>
                    <owl:onProperty rdf:resource="&onto2;#subject"/>
                    <owl:allValuesFrom rdf:resource="&onto2;#Science"/>
                </owl:Restriction>
            </owl:subClassOf>
        </owl:Class>
    </owl:equivalentClass>
</owl:Class>
```

But:

- It forces the use of a particular ontology language: OWL.
- It mixes correspondences and definitions. This is a problem for the clarity of alignments as well as for lightweight applications which do not want to interpret the OWL language..
- It cannot express data transformation.

Ontology Alignment

Alignment formats

- *Contextualized OWL (C-OWL)* is an extension of OWL to express mappings between heterogeneous ontologies.

  The new constructs in C-OWL, with respect to OWL, are called *bridge rules*, and they allow the expression of relations between classes, relations and individuals interpreted in heterogeneous domains. Bridge rules (*cowl:bridgeRule*) are oriented correspondences, from a source ontology O_1 to a target ontology O_2 . They use five relations:
  - equivalent ≡ (*cowl:Equivalent*);
  - more general ⊒ (*cowl:Onto*);
  - more specific ⊑ (*cowl:Into*);
  - disjoint ⊥(*cowl:Incompatible*);
  - overlap ≬(*cowl:Compatible*).

```
…
<cowl:Mapping>
  <cowl:sourceOntology>
    <owl:Ontology rdf:about="&onto1;"/>
  </cowl:sourceOntology>
  <cowl:targetOntology>
    <owl:Ontology rdf:about="&onto2;"/>
  </cowl:targetOntology>
  <cowl:bridgeRule>
    <cowl:Into>
      <cowl:source>
        <owl:Class rdf:about="&onto1;#Book"/>
      </cowl:source>
      <cowl:target>
        <owl:Class rdf:about="&onto2;#Volume"/>
      </cowl:target>
    </cowl:Into>
  </cowl:bridgeRule>
  <cowl:bridgeRule>
    <cowl:Onto>
      <cowl:source>
        <owl:Class rdf:about="&onto1;#name"/>
      </cowl:source>
      <cowl:target>
        <owl:Class rdf:about="&onto2;#title"/>
      </cowl:target>
    </cowl:Onto>
  </cowl:bridgeRule>
</cowl:Mapping>
```

But:

- It can express relatively simple alignments: no constructed classes are expressed, only named classes are used.

Ontology Alignment

Alignment formats

- *SWRL(Semantic Web Rule Language)* is a rule language for the semantic web. It extends OWL with an explicit notion of rule that is interpreted as first order Horn clauses. These rules can be understood as correspondences between ontologies, especially when elements from the head and the body are from different ontologies.

  SWRL mixes the vocabulary from RuleML for exchanging rules with the OWL vocabulary for expressing knowledge. It defines a rule (*ruleml:imp*) with a body (*ruleml:body*) and head (*ruleml:head*) parts.

```
…
<ruleml:imp>
  <ruleml:_body>
    <swrlx:classAtom>
      <owlx:Class owlx:name="&onto1;#Book" />
      <ruleml:var>p</ruleml:var>
    </swrlx:classAtom>
    <swrlx:datavaluedPropertyAtom swrlx:property="&onto1;#topic">
      <ruleml:var>p</ruleml:var>
      <owlx:DataValue   owlx:datatype="&xsd;#string">politics</owlx:DataValue>
    </swrlx:datavaluedPropertyAtom>
  </ruleml:_body>
  <ruleml:_head>
    <swrlx:classAtom swrlx:property="&onto2;#Politics">
      <ruleml:var>p</ruleml:var>
    </swrlx:classAtom>
  </ruleml:_head>
</ruleml:imp>
```

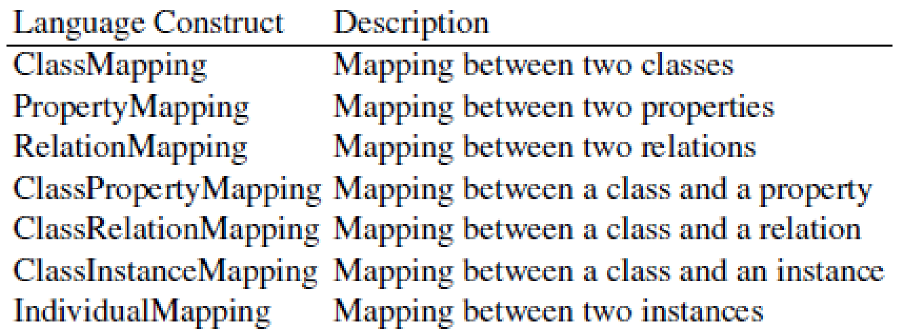
- *Rule Interchange Format (RIF)*

```
Group (
  Forall ?x (
      rdf:type( ?x onto2:Politics ) :-And( rdf:type( ?x onto1:Book ) onto1:topic( ?x "politics" ) )
  )
)
```

Ontology Alignment

Alignment formats

- *SEKT-ML* is an ontology mapping language developed in SEKT European project to specify mediators in semantic web services as defined in the Web Services Modeling Ontology (WSMO)



Language Construct	Description
ClassMapping	Mapping between two classes
PropertyMapping	Mapping between two properties
RelationMapping	Mapping between two relations
ClassPropertyMapping	Mapping between a class and a property
ClassRelationMapping	Mapping between a class and a relation
ClassInstanceMapping	Mapping between a class and an instance
IndividualMapping	Mapping between two instances

```
classMapping(
    annotation(<"rdfs:label"> 'Book to Volume')
    annotation(<"http://purl.org/dc/elements/1.1/description">
        'Map the Book concept to the Volume concept')
    unidirectional
    <"&onto1;#Book">
    <"&onto2;#Volume">)

relationMapping(
    annotation(<"rdfs:label"> 'authors to authors')
    bidirectional
    <"&onto1;#authors">
    <"&onto2;#authors">)
```

```
classMapping(
    annotation(<"rdfs:label"> 'conditional Book to Politics')
    unidirectional
    <"&onto1;#Book">
    <"&onto2;#Politics">
    attributeValuecondition(<"&onto1;#topic"> '= "politics"))

classMapping(
    annotation(<"rdfs:label"> 'conditional Volume to Pocket')
    unidirectional
    <"&onto2;#Volume">
    <"&onto1;#Pocket">
    attributeValuecondition(<"&onto2;#size"> '< 14))
```

Table is taken from [6]

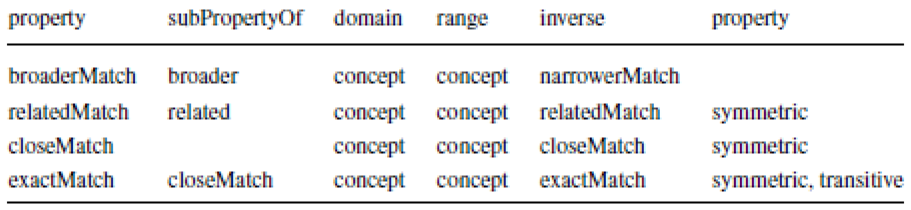
Ontology Alignment

Alignment formats

- *Simple Knowledge Organization System (SKOS)* core vocabulary is an RDF Schema aiming at expressing relationships between lightweight ontologies, e.g., folksonomies or thesauri. SKOS allows the identification of concepts in lightweight ontologies via the *skos:Concept class*.

```
<skos:Concept rdf:about="&onto1;#Book">
    <skos:prefLabel>Book</skos:prefLabel>
    <skos:altLabel xml:lang="fr">Livre</skos:altLabel>
    <skos:definition>A book is a set of sheets of papers bound together so that …</skos:definition>
    <skos:broader rdf:resource="&onto1;#Product"/>
    <skos:narrower rdf:resource="&onto1;#Pocket"/>
<skos:Concept/>
```

SKOS defines a mapping vocabulary which aims at expressing relationships across thesauri (kind of an alignment vocabulary). It is based on the properties:



property	subPropertyOf	domain	range	inverse	property
broaderMatch	broader	concept	concept	narrowerMatch	
relatedMatch	related	concept	concept	relatedMatch	symmetric
closeMatch		concept	concept	closeMatch	symmetric
exactMatch	closeMatch	concept	concept	exactMatch	symmetric, transitive

```
<skos:Concept rdf:about="&onto1;#Book">
    <skos:broaderMatch rdf:resource="&onto2;#Volume"/>
    <skos:narrowerMatch rdf:resource="&onto2;#Critics"/>
    <skos:relatedMatch rdf:resource="&onto2;#Work"/>
<skos:Concept/>
```

Table is taken from [6]

Ontology Alignment

Alignment formats

- *Alignment Format* is simpler than most of the alignment representations presented here. It aims at being producible by most matching tools yet preserving the capability to handle complex alignment definitions. The alignment description is *an envelope in which the correspondences are grouped*. Alignments are made of:

- ***References*** to matched ontologies
- ***A set of correspondences*** which expresses the relation holding between entities of the first ontology and entities of the second ontology
  - ***entity1***: the first matched entity;
  - ***entity2***: the second matched entity;
  - ***relation***: the relation holding between the two entities. It is not restricted to the equivalence relation, but can be more sophisticated, e.g., subsumption, incompatibility, or even some fuzzy relation. *The default relation is equivalence*.
  - ***strength***: the confidence that the correspondence under consideration holds (for instance, a float value between 0 and 1). The default strength is maximum value.
  - ***id***: an identifier for the correspondence.
- ***Level*** used for characterizing the type of correspondence. The Alignment format has been designed for offering a common format to different needs. Depending on the *expressiveness of the matched entities*, it offers several alignment levels which correspond to different options for expressing entities:
  - ***Level 0 ("0")***: These alignments relate *entities identified by URIs*. Any algorithm can deal with such alignments. This first level of alignment has the advantage to not depend on a particular language for expressing these entities. On this level, the matched entities may be classes, properties or individuals. However, they also can be any kind of a complex term that is used by the target language as soon as it is identified by a URI.
  - ***Level 1 ("1")***: These alignments *replace pairs of entities by pairs of sets (or lists) of entities*. A level 1 correspondence is thus a slight refinement of level 0. It can be easily parsed and is still language independent.
  - ***Level 2 ("2EDOAL")***: More general correspondence expressions may be useful (for instance, bridges from an ontology of services to the currently existing semantic web service description languages in first order logic). These are no longer language independent and *require the knowledge of the language used for parsing the format*. Correspondences can be expressed between formulas, queries, etc.
- ***Arity (type)*** **:** (default 1:1) denoted by ***1*** for injective and total, ***?*** for injective, **+** for total and ******* for none, with each sign concerning one mapping and its converse

Ontology Alignment

Alignment formats

- *Alignment Format* allows the expression of alignments *without commitment to a particular language*. It is not targeted towards a particular use of the alignments and *offers generators for many other formats*. One of its good features is its *openness* which allows the *introduction of new relations* and, if necessary, *new types of expressions* while keeping the compatibility with poorly expressive languages. *URL: http://alignapi.gforge.inria.fr/format.html*

```
<rdf:RDF>
…
<Alignment>
    <xml>yes</xml>
    <level>0</level>
    <type>**</type>
    <time>7</time>
    <method>fr.inrialpes.exmo.align.impl.method.StringDistAlignment </method>
    <onto1>
        <Ontology rdf:about="&onto1">
            <location>file:examples/rdf/onto1.owl</location>
        </Ontology>
    </onto1>
    <onto2>
        <Ontology rdf:about="&onto2">
            <location>file:examples/rdf/onto2.owl</location>
        </Ontology>
    </onto2>
```

```
    <map>
        <Cell>
            <entity1 rdf:resource='&onto1;#Book'/>
            <entity2 rdf:resource='&onto2;#Volume'/>
            <relation>&lt;</relation>
            <measure rdf:datatype='&xsd;float'>0.636  </measure>
        </Cell>
    </map>
    <map>
        <Cell>
            <entity1 rdf:resource='&onto1;#name'/>
            <entity2 rdf:resource='&onto2;#title'/>
            <relation>&gt;</relation>
            <measure rdf:datatype='&xsd;float'>1.0</measure>
        </Cell>
    </map>
</Alignment>
</rdf:RDF>
```

&lt; = ” < ”
&gt; = ” > ”

The Alignment format can be manipulated through the ***Alignment API*** (*http://alignapi.gforge.inria.fr*). It has been used as the format for the OAEI evaluation campaigns, so many different tools are able to output it, e.g., oMap, FOAM, OLA, Falcon-AO, HCONE, and many others.

Ontology Alignment

- **Alignment formats**
  - *Alignment Format.* Examples of Level 2 alignments.

```
<Cell>
  <entity1 rdf:resource='&onto2;#Writer'/>
  <entity2>
    <owl:Restriction>
      <owl:onProperty rdf:resource="&onto1;#hasWritten"/>
      <owl:minCardinality
        rdf:datatype="&xsd;nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </entity2>
  <measure rdf:datatype='&xsd;float'>0.6363636363636364</measure>
  <relation>&lt;</relation>
</Cell>
```

A *Writer* class in the second ontology is some class that has property *hasWritten* in the first one.

```
<Cell>
  <entity1 rdf:resource='&onto1;#name'/>
  <entity2>
    <Apply rdf:resource='string-concatenate'>
      <args rdf:parseType="collection">
        <Path>
          <relation rdf:resource="&onto2;#firstname" />
        </Path>
        <Path>
          <relation rdf:resource="&onto2;#lastname" />
        </Path>
       </args>
    </Apply>
  </entity2>
  <measure rdf:datatype='&xsd;float'>1.0</measure>
  <relation>=</relation>
</Cell>
```

The *name* property in the first ontology is equivalent to the concatenation of the properties *firstname* and *lastname* in the second one.

Ontology Alignment

Alignment formats

- *Expressive and Declarative Ontology Alignment Language (EDOAL)* is a *level 2* language for the *Alignment API*. It is an expressive language independent from any ontology language. This independence allows for the representation of alignments between heterogeneous and weak representations, such as a thesaurus and a relational database. Thus, EDOAL has a middle man position: it is *independent from any particular language,* but *expressive enough* for covering a large part of the other languages.

```
<Cell rdf:about="pocketbook">
  <entity1>
    <edoal:Class>
      <edoal:and rdf:parseType="Collection">
        <edoal:Class rdf:about="&o;Pocket"/>
        <edoal:AttributeValueRestriction>
          <edoal:onAttribute>
            <edoal:Property rdf:about="&o;creator"/>
          </edoal:onAttribute>
          <edoal:comparator rdf:resource="&edoal;equals"/>
          <edoal:value>
            <edoal:Property rdf:about="&o;topic"/>
          </edoal:value>
        </edoal:AttributeValueRestriction>
      </edoal:and>
    </edoal:Class>
  </entity1>
```

```
  <entity2>
    <edoal:Class>
      <edoal:and rdf:parseType="Collection">
        <edoal:Class rdf:about="&o';Autobiography"/>
        <edoal:AttributeValueRestriction>
          <edoal:onAttribute>
            <edoal:Property rdf:about="&o';size"/>
          </edoal:onAttribute>
          <edoal:comparator rdf:resource="&edoal;less-than"/>
          <edoal:value>
            <edoal:Literal edoal:type="&xsd;integer" edoal:string="14" />
          </edoal:value>
        </edoal:AttributeValueRestriction>
      </edoal:and>
    </edoal:Class>
  </entity2>
  <measure rdf:datatype='&xsd;float'>1.</measure>
  <relation>=</relation>
</Cell>
```

Ontology Alignment

Alignment formats

- *Expressive and Declarative Ontology Alignment Language (EDOAL)* is supported by the Alignment API through providing an API for manipulating EDOAL alignments, parsers, and renderers in RDF, OWL, and SPARQL. EDOAL extends the Alignment format in order to capture correspondences more precisely using the following features:
    - ***Construction*** of entities from other entities can be expressed through algebraic operators. Constructed entities allows to overcome the shallowness of some ontologies.
    - ***Restrictions*** can be expressed on entities in order to narrow their scope.
    - ***Transformations*** of property values can be specified. Property values using different encodings or units can be aligned using transformations.
    - ***Linkkeys*** can be defined for expressing conditions under which, instances of the aligned entities should be considered equivalent.
- *URL: http://alignapi.gforge.inria.fr/edoal.html*

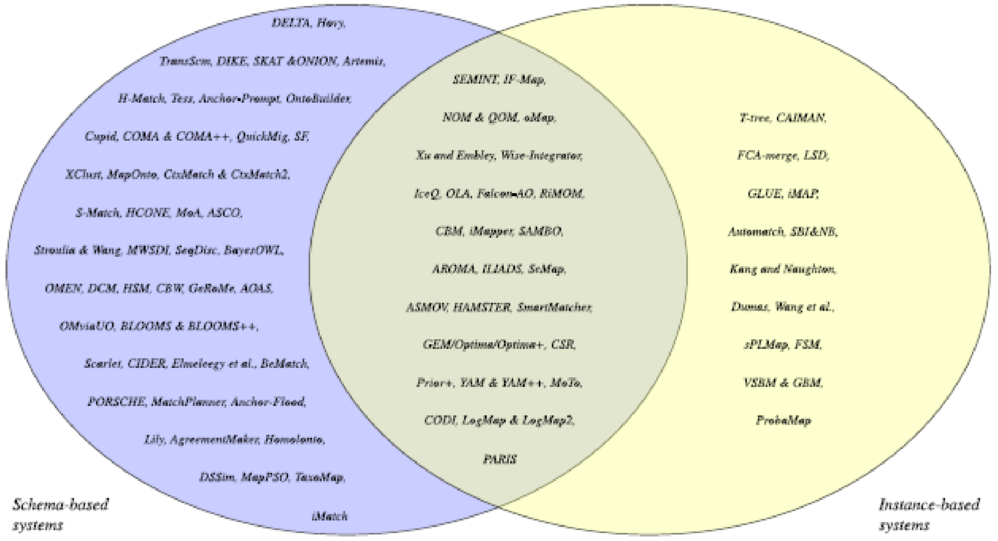
```
<Cell rdf:about="pocketbook">
  <entity1>
    <edoal:Relation>
      <edoal:and rdf:parseType="Collection">
        <edoal:Class rdf:about="&o';creator"/>
        <edoal:DomainRestriction>
          <edoal:class rdf:about="&o;Book"/>
        </edoal:DomainRestriction>
      </edoal:and>
    </edoal:Relation>
  </entity1>
  <entity2>
    <edoal:Relation rdf:about="&o';author"/>
  </entity2>
  <measure rdf:datatype='&xsd;float'>1.</measure>
  <relation>=</relation>
</Cell>
```

```
<edoal:transformation>
  <edoal:Transformation edoal:direction="o-">
    <edoal:entity1>
      <edoal:Property rdf:about="&foaf;name">
    </edoal:entity1>
    <edoal:entity2>
      <edoal:Apply edoal:operator="&edoal;concat">
        <edoal:arguments rdf:parseType="Collection">
          <edoal:Property rdf:about="vcard:firstname" />
          <edoal:Literal edoal:type="&xsd;string" edoal:string=" " />
          <edoal:Property rdf:about="vcard:middleinitial" />
          <edoal:Literal edoal:string=". " />
          <edoal:Property rdf:about="vcard:lastname" />
        </edoal:arguments>
      </edoal:Apply>
    </edoal:entity2>
  </edoal:Transformation>
</edoal:transformation>
```

Ontology Alignment

Matching systems [6][7]

- *Schema-based systems* rely mostly on schema-level input information for performing ontology matching.
- *Instance-based system* are taking advantage mostly of instances, i.e., of data expressed with regard to the ontology or data indexed by the ontology.
- *Mixed, schema-based and instance-based systems* take advantage of both schema-level and instance-level input information if they are both available.
- *Meta-matching systems* advance in the way they use and combine other matching systems rather than in the matchers themselves.


DELTA, Hovy,
TransScm, DIKE, SKAT &ONION, Artemis,
H-Match, Tess, Anchor-Prompt, OntoBuilder,
Cupid, COMA & COMA++, QuickMig, SF,
XClust, MapOnto, CtxMatch & CtxMatch2,
S-Match, HCONE, MoA, ASCO,
Stroulia & Wang, MWSDI, SeqDisc, BayesOWL,
OMEN, DCM, HSM, CBW, GeRoMe, AOAS,
OMviaUO, BLOOMS & BLOOMS++,
Scarlet, CIDER, Elmeleegy et al., BeMatch,
PORSCHE, MatchPlanner, Anchor-Flood,
Lily, AgreementMaker, Homolonto,
DSSim, MapPSO, TaxoMap,
iMatch
Schema-based systems
SEMINT, IF-Map,
NOM & QOM, oMap,
Xu and Embley, Wise-Integrator,
IceQ, OLA, Falcon-AO, RiMOM,
CBM, iMapper, SAMBO,
AROMA, ILIADS, ScMap,
ASMOV, HAMSTER, SmartMatcher,
GEM/Optima/Optima+, CSR,
Prior+, YAM & YAM++, MoTo,
CODI, LogMap & LogMap2,
PARIS
T-tree, CAIMAN,
FCA-merge, LSD,
GLUE, iMAP,
Automatch, SBI&NB,
Kang and Naughton,
Dumas, Wang et al.,
sPLMap, FSM,
VSBM & GBM,
ProbaMap
Instance-based systems

***Useful link**:* *http://www.ontologymatching.org/*
http://oaei.ontologymatching.org/
http://www.mkbergman.com/1769/50-ontology-mapping-and-alignment-tools/
http://www.mkbergman.com/2129/30-active-ontology-alignment-tools/
http://www.ke.tu-darmstadt.de/resources/ontology-matching

Picture is taken from [6]

Ontology Alignment

Alignment frameworks

- *COMA3.0/COMA++/COMA (University of Leipzig)* is a schema matching infrastructure that provides an extensible library of *matching algorithms*, a framework for *combining* obtained results, and a platform for the *evaluation* of the effectiveness of the different matchers. It enables *importing*, *storing* and *editing* schemas (or models), and allows various operations on the alignments (*compose*, *merge* and *compare*) to be applied to schemas for *transforming* or *merging* them. It provides user connection through GUI, a SaaS-solution and an API. It has Community and Business Editions.
  - *URL: http://dbs.uni-leipzig.de/en/Research/coma.html*
- *ATOM (University of Leipzig)* is an approach for *automatic target-driven ontology* (especially taxonomies) *merging* [4]. It supports asymmetric or target-driven merging where it merges the source ontology into the target ontology. It gives preference to the target ontology and guarantee its complete preservation that contributes to its stability which is valuable for applications. ATOM has been integrated within the COMA ontology/schema match system and is now part of the COMA 3.0 Business Edition.
  - *URL: http://dbs.uni-leipzig.de/en/atom*
- *GOMMA (University of Leipzig)* is an infrastructure for Generic Ontology Matching and mapping Management targeting the evolution of life science ontologies and mappings. It is made of three levels:
  - repository (responsible for data management, such as handling versions of ontologies and mappings);
  - functional components operating over data (match, diff and evolution);
  - tools, built on top of the previous levels: Ontology Matcher (used to find mappings), complex ontology diff (COntoDiff)(used to detect basic and complex changes between different ontology versions), or ontology evolution explorer (OnEx)(used to explore and visualize the changes).

  GOMMA supports parallel execution of matchers on multiple computing nodes or CPU cores as well as ontology partitioning to handle large-scale ontologies.
  - *URL: http://dbs.uni-leipzig.de/en/gomma*

Ontology Alignment

Alignment frameworks

- *Harmony (The MITRE Corporation)* is a ***schema integration workbench*** in which multiple tools share a common RDF-based knowledge repository. It is an innovative, open source ***schema matching tool***, available both as a standalone and as part of the OpenII information integration tool suite. Harmony is a semi-automated tool that *greatly speeds the finding of correspondences across two data schemas.* The workbench includes Schemr, a ***schema search engine*** that helps users ***discover and visualize*** relevant schemas in information integration tasks through sharing and reuse.
  - *URL: http://openii.sourceforge.net/index.php?act=tools&page=harmony*

- *MAFRA (Instituto Politecnico do Porto and University of Karlsruhe)* is an interactive, incremental and dynamic MApping FRAmework [5] for mapping distributed ontologies. It allows ***Normalization***, ***Similarity calculation***, ***Semantic bridging*** (data translation), etc.
  - *URL: http://mafra-toolkit.sourceforge.net/*

- *FOAM (University of Karlsruhe)* is a Framework for Ontology Alignment and Mapping [6], is a general tool for processing similarity-based ontology matching. The FOAM framework bundles several algorithms and strategies as well as matching systems such as NOM, QOM, and APFEL (more systems can be integrated). It offers a web-based interface. It is also available as a Protégé plug-in. Finally, it offers translation tools from and to the Alignment format and other formats.

Ontology Alignment

- **Alignment frameworks**
  - *The Protégé Prompt Suite (*Stanford University*)* an environment that provides some matching methods and alignment visualization within Protégé ontology edition environment. It is an interactive framework for ***comparing***, ***matching*** (Anchor-Prompt tool), ***merging*** (iPrompt tool), ***maintaining versions*** (PromptDiff tool), and ***translating*** between different knowledge representation formalisms.
    - *URL: http://protegewiki.stanford.edu/wiki/PROMPT*
  - *Alignment API (*INRIA Rhˆone-Alpes*)* a Java ***API for manipulating alignments*** in the Alignment format and EDOAL. It defines a set of interfaces and a set of functions that they can perform. The Alignment API provides support for manipulating alignments. It offers the following functions: ***Parsing and serializing***, ***Computing***, ***Thresholding***, ***Hardening***, ***Comparing*** and ***Outputting*** alignments. The Alignment API also supports a server to ***store*** and ***share alignments*** on the web.
    - *URL: http://alignapi.gforge.inria.fr/*
    - *Systems using the Alignment API: http://alignapi.gforge.inria.fr/impl.html*
  - *The NeOn Toolkit Alignment Plug-in.* The NeOn Toolkit for ontology management features run-time and design-time ontology alignment support. It is based on the Alignment API and provides following functionalities: ***retrieving alignments***, ***matching ontologies***, ***trimming alignments*** under various thresholds, ***storing*** them in permanent stores, and ***rendering*** them in numerous output formats.
    - *URL: http://neon-toolkit.org/wiki/Main_Page*

Ontology Alignment

- **Some of the matching systems**
  - *S-Match* S-Match is a semantic matching framework, which provides several semantic matching algorithms and facilities for developing new ones. It contains implementation of the original S-Match semantic matching algorithm, as well as minimal semantic matching algorithm and structure preserving semantic matching algorithm.
    - *URL: https://sourceforge.net/projects/s-match and https://github.com/s-match*
  - *AgreementMaker* aims at being a user friendly, powerful, and flexible ontology and schema matching system. It is comprising a wide range of automatic matches, an extensible and modular architecture, a multi-purpose user interface, a set of evaluation strategies, and various manual, e.g., visual comparison, and semi-automatic features, e.g., user feedback. *AgreementMakerLight* – an automated and efficient ontology matching system derived from *AgreementMaker*.
    - *URL: https://agreementmaker.github.io/ and https://github.com/agreementmaker*
  - *Falcon-AO* is an automatic divide-and-conquer approach to ontology matching. It handles ontologies in RDFS and OWL. It has been designed with the goal of dealing with large ontologies (of thousands of entities) via: *partitioning*, *matching blocks*, and *discovering alignments*.
    - *URL: http://ws.nju.edu.cn/falcon-ao/*
  - *HerTUDA* is a simple, fast ontology matching tool, based on syntactic string comparison and filtering of irrelevant mappings. Despite its simplicity, it outperforms many state-of-the-art ontology matching tools.
    - *URL: http://www.ke.tu-darmstadt.de/resources/ontology-matching/hertuda*
  - *BLOOMS* is a tool for ontology matching. It utilizes information from Wikipedia category hierarchy and from the web to identify subclass relationship between entities.
    - *URL: https://github.com/jainprateek/BLOOMS*

Ontology Alignment

- **Challenges:**
  - *Large-scale matching evaluation*
    - Large tests involving 10.000, 100.000, and 1.000.000 entities per ontology are to be designed and conducted.
    - We need a wider automation for acquisition of reference alignments, e.g., by minimizing the human effort while increasing an evaluation dataset size.
    - More accurate evaluation quality measures are needed.
  - *Efficiency of matching techniques*
    - The efficiency of matchers (*execution time*) is very important for dynamic applications (especially, when a user cannot wait too long for the system to respond or when memory is limited).
    - Current ontology matchers are mostly design time tools which are usually not optimized for resource consumption.
    - Good execution time can be achieved by using a large amount of main memory.
    - Scalable ontology matching reference solutions might help to perform matching on handheld computers or smartphones.
  - *Matcher selection, combination, self-configuration and tuning*
    - There is no single matcher that clearly dominates others.
    - It is necessary to be able to take advantage of the best configuration of matchers taking into account various constrains and requirements, tasks and application domains.

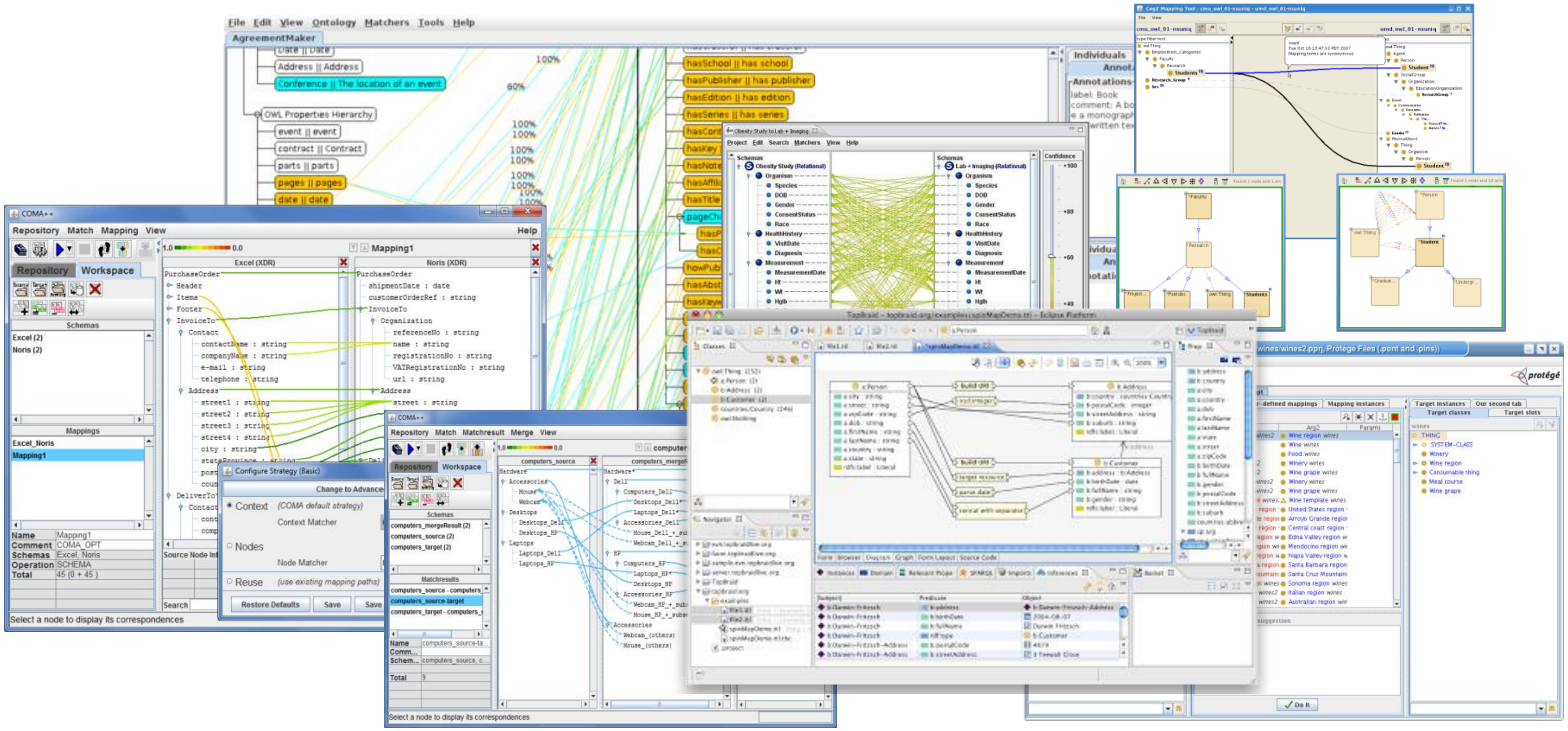
Ontology Alignment

- **Challenges:**
  - *User involvement*
    - Matching performed at *design time* is screened by human-users before being accepted.
    - In *dynamic applications*, users are generally not ontology matching specialists who can be asked to inspect the alignments.
    - Involvement should be less stressful for the user:
      - *at design time*, interaction should be both natural and complete;
      - *at run time*, it should be hidden in the user task.
    - Matching tools have to be configurable and customizable.
    - Involving end users in an active manner in a matching project would increase its *impact*.
  - *Explanation of matching results*
    - In order to facilitate informed decision making, to gain a wider acceptance and to be trusted by users, matching systems should provide explanations of their results to users or to other programs that exploit them in a simple, yet clear and precise, way.
  - *Social and collaborative matching*
    - Social support, as an approach to solve hard and large problems, can be also applied to ontology matching. But this calls for algorithms able to rank a massive amount of correspondences.
    - Alignment supporting tasks are relatively easy for humans, but difficult for machines.
    - How to individuate and deal with malicious users, and which incentive schemes promise to facilitate user participation in establishing alignments collaboratively?

Ontology Alignment

- **User involvement:**
  - *Visually supported alignment tools* [12][13]

Tree-based representations use the standard tree widget to present the class hierarchy of an ontology. Supported by *AgreementMaker, COMA3.0 (COMA++), COGZ, PROMPT, MAFRA, Harmony, S-Match, SPINMap* systems.


AgreementMaker
COMA++
CogZ Mapping Tool
Configure Strategy (Basic)
TopBraid - topbraid.org/examples/spinMapDemo.ttl - Eclipse Platform
protégé

Ontology Alignment

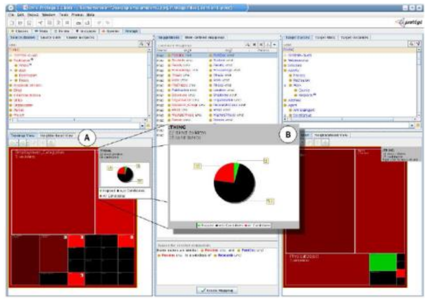
- **User involvement:**
  - *Visually supported alignment tools* [12][13]

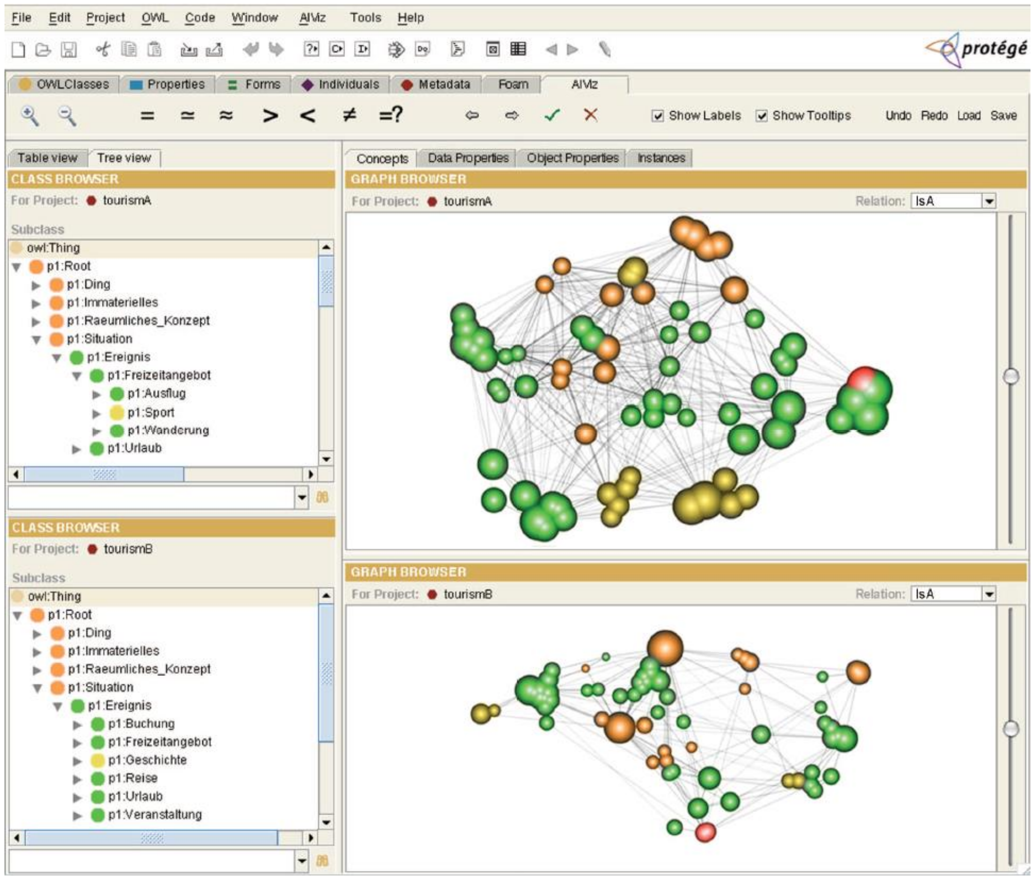
Graph-based representations is used for the navigation and exploration of ontologies, to provide insight into the ontology structure, and to show detailed information of every ontological element.

Supported by *AlViz* [9], *PROMPT* systems.

Treemap is a commonly used visual representation of hierarchically organized data sets. Nodes of the hierarchy are represented as nested rectangles.

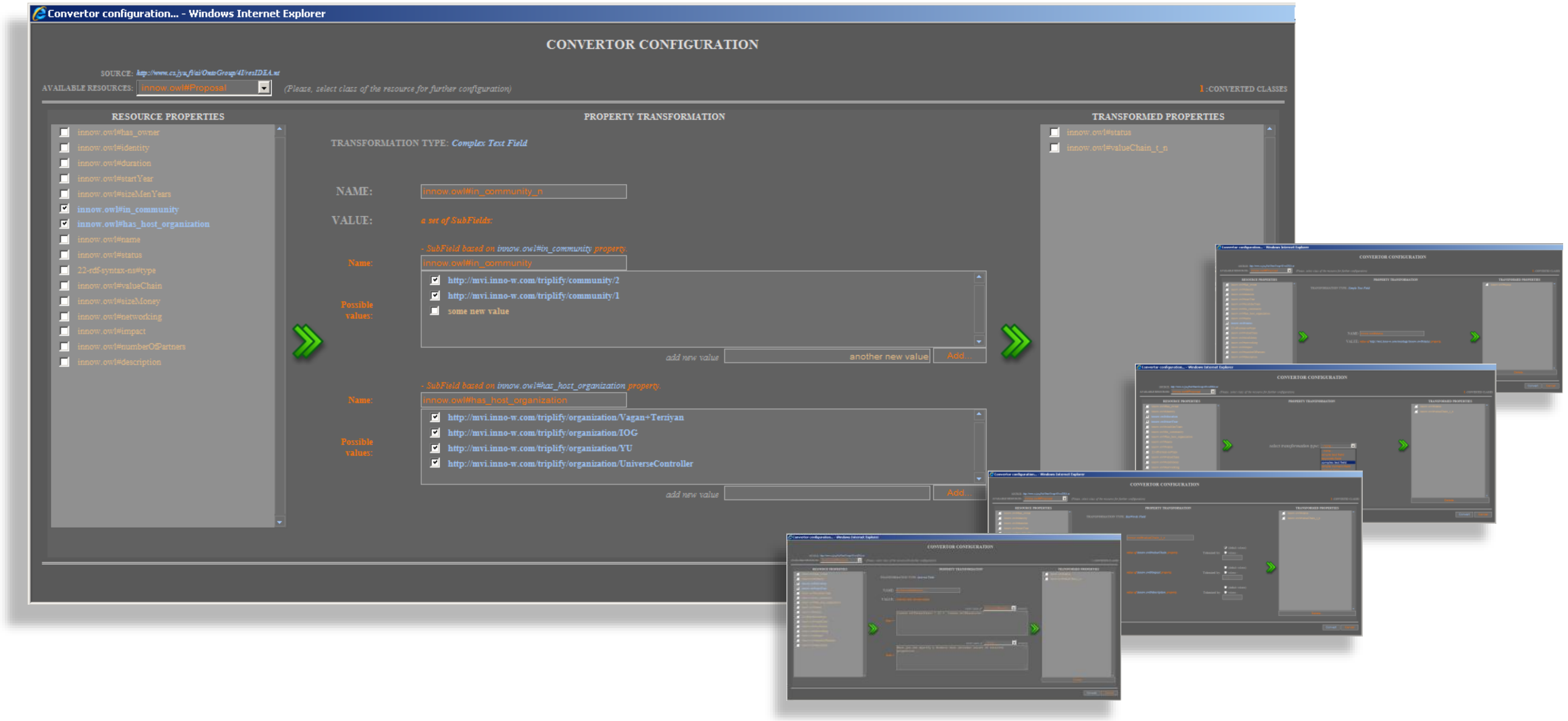
Supported by *COGZ* systems.





Ontology Alignment

- **User involvement:**
  - *Application-driven User-facilitated Ontology Alignment*


Convertor configuration... - Windows Internet Explorer

CONVERTOR CONFIGURATION

SOURCE: http://www.cs.jyu.fi/ai/OntoGroup/4I/resIDEA.nt

AVAILABLE RESOURCES: innow.owl#Proposal (Please, select class of the resource for further configuration)

1 :CONVERTED CLASSES

RESOURCE PROPERTIES

- [ ] innow.owl#has_owner
- [ ] innow.owl#identity
- [ ] innow.owl#duration
- [ ] innow.owl#startYear
- [ ] innow.owl#sizeMenYears
- [x] innow.owl#in_community
- [x] innow.owl#has_host_organization
- [ ] innow.owl#name
- [ ] innow.owl#status
- [ ] 22-rdf-syntax-ns#type
- [ ] innow.owl#valueChain
- [ ] innow.owl#sizeMoney
- [ ] innow.owl#networking
- [ ] innow.owl#impact
- [ ] innow.owl#numberOfPartners
- [ ] innow.owl#description

PROPERTY TRANSFORMATION

TRANSFORMATION TYPE: *Complex Text Field*

NAME: innow.owl#in_community_n

VALUE: *a set of SubFields:*

- SubField based on innow.owl#in_community property.

Name: innow.owl#in_community

Possible values:
- [x] http://mvi.inno-w.com/triplify/community/2
- [x] http://mvi.inno-w.com/triplify/community/1
- [ ] some new value

add new value another new value Add...

- SubField based on innow.owl#has_host_organization property.

Name: innow.owl#has_host_organization

Possible values:
- [x] http://mvi.inno-w.com/triplify/organization/Vagan+Terziyan
- [x] http://mvi.inno-w.com/triplify/organization/IOG
- [x] http://mvi.inno-w.com/triplify/organization/YU
- [x] http://mvi.inno-w.com/triplify/organization/UniverseController

add new value Add...

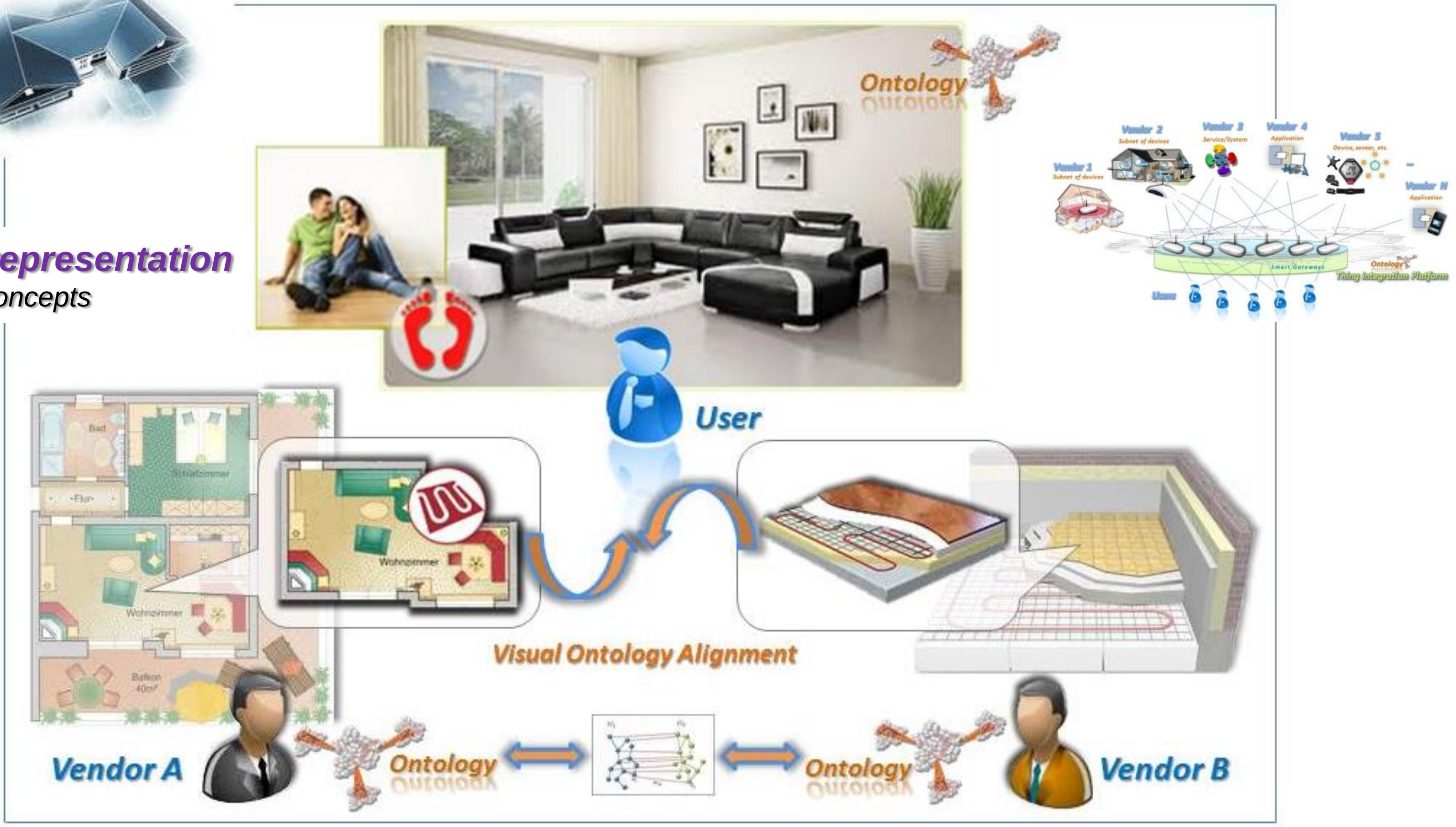
TRANSFORMED PROPERTIES

- [ ] innow.owl#status
- [ ] innow.owl#valueChain_t_n

Ontology Alignment

- **User involvement:**
  - *Application-driven Concept Visualization based Ontology Alignment* [15]

Through **visual representation** of the concepts



Similarity Matching

Similarity matching tools:

- ***Alignment API*** is an API and implementation for expressing and sharing ontology alignments. Hence, it is not a matcher. A few examples of trivial matchers are provided with the Alignment API which will indeed match ontologies. *http://alignapi.gforge.inria.fr/* It is always under development. So, please check release notes: *http://alignapi.gforge.inria.fr/relnotes.html*
- ***Apache Commons*** is the Commons Text library that provides additions to the standard JDK's text handling - a consistent set of tools for processing text generally from computing distances to being able to efficiently do String escaping of various types. *https://commons.apache.org/proper/commons-text/*

  Currently it provides the following algorithm implementations for string similarity:
  - CosineDistance/Similarity,
  - FuzzyScore,
  - HammingDistance,
  - JaroWinklerDistance/Similarity,
  - LevenshteinDistance,
  - LongestCommonSubsequenceDistance,
  - etc… *https://commons.apache.org/proper/commons-text/*
- ***Ontology Alignment Evaluation Initiative*** is a coordinated international initiative to evaluate the schema or ontology matching methods. Some of the public OAEI systems are available from corresponding workshops. *http://oaei.ontologymatching.org/*

Similarity Matching

Similarity matching tools:

- ***SimMetrics***: SimMetrics is an open-source extensible library of Similarity or Distance Metrics. e.g. from edit distance's (Levenshtein, Gotoh, Jaro etc) to other metrics, (e.g. Soundex, Chapman). Work provided by UK Sheffield University funded by (AKT) an IRC sponsored by EPSRC. (*https://github.com/Simmetrics/simmetrics, https://sourceforge.net/projects/simmetrics/*)
- ***SecondString***: SecondString is an open-source Java-based package of approximate string-matching techniques. It includes abstract classes for various edit-distance based comparators, concrete implementations of several published approximate comparators. (*http://secondstring.sourceforge.net*)
- ***SimPack***: SimPack is intended primarily for the research of similarity between concepts in ontologies or ontologies as a whole

  (*https://files.ifi.uzh.ch/ddis/oldweb/ddis/research/simpack/index.html*) ***[16]***
- ***OntoSim***: OntoSim is a Java API allowing to compute similarities between ontologies. It relies on the Alignment API for ontology loading so it is quite independent of the ontology API used (JENA or OWL API) (*http://ontosim.gforge.inria.fr/*)
- ***VeeAlign:*** Multifaceted Context Representation Using Dual Attention for Ontology Alignment. Deep Learning based model that uses a novel dual-attention mechanism to compute the contextualized representation of a concept which, in turn, is used to discover alignments. *https://aclanthology.org/2021.emnlp-main.842/ , https://github.com/Remorax/VeeAlign*

Ontology Alignment

- **Java Alignment API:**

```
…
Properties params = new Properties();
try {
   URI foreignOnto = new URI(sour1);
   URI ownOnto = new URI(sour2);
// Run two different alignment methods (e.g., ngram distance and smoa)
   AlignmentProcess a1 = new StringDistAlignment();
   params.setProperty("stringFunction","smoaDistance");
   a1.init(foreignOnto,ownOnto);
   a1.align((Alignment)null, params);
   AlignmentProcess a2 = new StringDistAlignment();
   a2.init (foreignOnto,ownOnto);
   params = new Properties();
   params.setProperty("stringFunction","ngramDistance");
   a2.align((Alignment)null, params);
// Trim above .5 and .7 respectively
   a1.cut(0.5);
   a2.cut(0.7);
// Clone and merge alignments
   BasicAlignment a1a2 = (BasicAlignment)(a1.clone());
   a1a2.ingest(a2);
} catch (AlignmentException e) {
   e.printStackTrace();
} catch (URISyntaxException e) {
   e.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Ontology Alignment

- **Java Alignment API:**

```
…
// Access alignment Cells
Iterator<Cell> iterator = a1.iterator();
while (iterator.hasNext()) {
   Cell cell = iterator.next();
   String cellStr = cell.toString();
   String semantics = cell.getSemantics();
   String object1 = cell.getObject1().toString();
   String object2 = cell.getObject2().toString();
   String relation = cell.getRelation().toString();
   String strength = ""+cell.getStrength();
}
```

Ontology Alignment

- **Java Alignment API:**

```
// Save alignment to file

try {
   Alignment result = (BasicAlignment)((BasicAlignment)al).clone();
   File file = new File(path, fileName);
   FileOutputStream fop = new FileOutputStream(file);
   PrintWriter writer = new PrintWriter (
      new BufferedWriter(new OutputStreamWriter( fop, "UTF-8" )), true);
// Displays it as RDF
   AlignmentVisitor renderer = new RDFRendererVisitor(writer);
   result.render(renderer);
   writer.flush();
   writer.close();
} catch (FileNotFoundException e) {
   e.printStackTrace();
} catch (AlignmentException e) {
   e.printStackTrace();
} catch (Exception e) {
   e.printStackTrace();
}
```

Set of AlignmentVisitors to display alignments in a variety of formats: ***COWLMappingRendererVisitor, HTMLMetadataRendererVisitor, HTMLRendererVisitor, JSONRendererVisitor, OWLAxiomsRendererVisitor, RDFRendererVisitor, SEKTMappingRendererVisitor, SILKRendererVisitor, SKOSRendererVisitor, SPARQLConstructRendererVisitor, SPARQLSelectRendererVisitor, SWRLRendererVisitor, XMLMetadataRendererVisitor, XSLTRendererVisitor.***

Similarity Matching

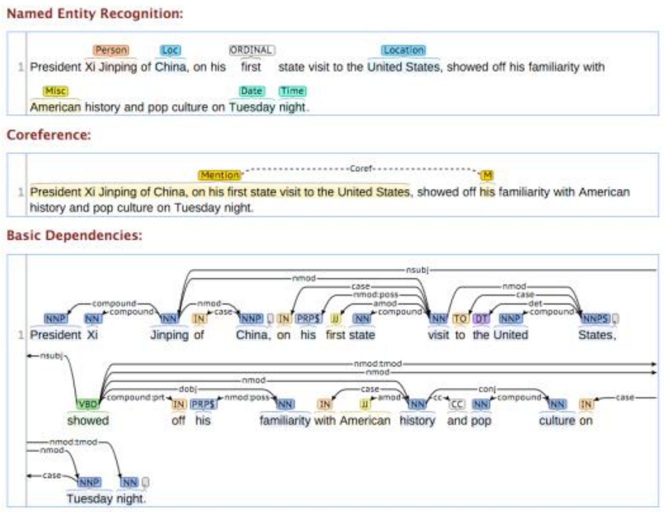
String measures available in Java packages

SimMetrics	SecondString	Alignment API	SimPack
	n-grams	*n-grams*	
Levenshtein	Levenshtein	Levenshtein	Levenshtein
Jaro	Jaro	Jaro	
Jaro–Winkler	Jaro–Winkler	Jaro–Winkler	
Needleman–Wunch	Needleman–Wunch	Needleman–Wunch	
		Smoa	
Smith–Waterman			
Monge–Elkan	Monge–Elkan		
Gotoh			
Matching coefficient			
Jaccard	Jaccard		Jaccard
Dice coefficient			Dice coefficient
	TFIDF		TFIDF
Cityblocks			Cityblocks
Euclidean			Euclidean
Cosine			Cosine
Overlap			Overlap
Soundex			

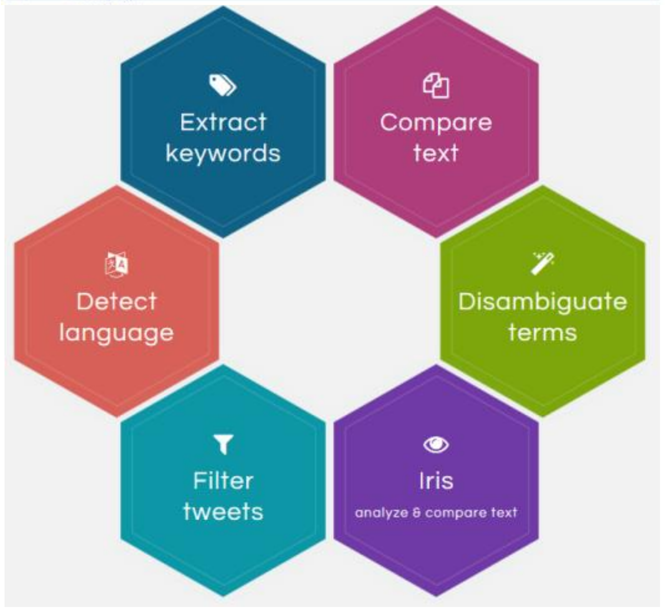
Similarity Matching

- **NLP support tools:**

  - ***Stanford NLP*** is a state-of-the-art technology for robust, broad-coverage natural-language processing in a number of languages that provides a widely used, integrated NLP toolkit, ***Stanford CoreNLP***. Particular technologies include competition-winning coreference resolution system; a high speed, high performance neural network dependency parser; a state-of-the-art part-of-speech tagger; a competition-winning named entity recognizer; and algorithms for processing Arabic, Chinese, French, German, and Spanish text. (*https://stanfordnlp.github.io/CoreNLP/*)
  - NLP related ***Cognitive Computing Services***, offered by various cloud service providers (e.g. IBM Watson, Google (DeepMind), Microsoft Azure, Amazon, etc.), help to parse and analyze texts, perform useful extractions and build intelligent text-based solutions
  - ***Cortical.io*** has developed a new machine learning approach inspired by the latest findings on the way the brain processes information. ***Semantic Folding*** proposes a statistics-free processing model that uses similarity as a foundation for intelligence. It breaks with traditional methods based on pure word count statistics or linguistic rule engines. Explore how the power of ***Cortical.io Retina technology*** can help building intelligent text-based solutions… . (*http://www.cortical.io*)
  - …



1 President Xi Jinping of China, on his first state visit to the United States, showed off his familiarity with American history and pop culture on Tuesday night.



References

- ***http://www.ontologymatching.org***
- [1] Yves R. Jean-Mary,E. Patrick Shironoshita,Mansur R. Kabuka. Ontology matching with semantic verification. Web Semantics: Science, Services and Agents on the World Wide Web, Elsevier, 2009.
- [2] P. Bouquet, G. Kuper, M. Scoz and S. Zanobini. Asking and answering semantic queries. In Proc. of Meaning Coordination and Negotiation Workshop (MCNW-04) in conjunction with International Semantic Web Conference (ISWC-04), 2004.
- [3] C. Leacock, M. Chodorow. Combining local context and WordNet similarity for word sense identification, In Fellbaum MIT Press, pp. 265–283, 1998.
- [4] Z. Wu, M. Palmer. Verb semantics and lexical selection, In 32nd Annual Meeting of the Association for Computational Linguistics, Las Cruces, New Mexico, pp. 133–138, 1994.
- [5] P. Haase, R. Siebes, F. van Harmelen. Peer Selection in Peer-to- Peer Networks with Semantic Topologies. In Proc. of International Conference on Semantics of a Networked World: Semantics for Grid Databases, 2004.
- [6] J. Euzenat and P. Shvaiko. Ontology matching. Springer-Verlag, Berlin Heidelberg (DE), 2013, isbn 978-3-642-38720-3. (http://wtlab.um.ac.ir/images/e-library/ontology/Ontology%20Matching.pdf)
- [7] L. Otero-Cerdeira, F. J. Rodriguez-Martinez, A. Gomez-Rodriguez: Ontology matching: a literature review Expert Systems with Applications, 2015. (http://disi.unitn.it/~p2p/RelatedWork/Matching/Cerdeira-Ontology%20Matching-2015.pdf)
- [8] Arch-int, N.; Arch-int, S., "Semantic Information Integration for Electronic Patient Records Using Ontology and Web Services Model," *Information Science and Applications (ICISA), 2011 International Conference on* , vol., no., pp.1,7, 26-29 April 2011
- [9] Raunich, S. & Rahm, E. (2014), 'Target-driven merging of taxonomies with Atom.', *Inf. Syst.* 42 , 1-14 .
- [10] Silva, N.; Rocha, J., "Semantic Web complex ontology mapping," *Web Intelligence, 2003. WI 2003. Proceedings. IEEE/WIC International Conference on* , vol., no., pp.82,88, 13-17 Oct. 2003
- [11] Ehrig, M., and Y. Sure, "FOAM – Framework for Ontology Alignment and Mapping; Results of the Ontology Alignment Initiative", *Workshop on Integrating Ontologies*, 2005.

References

- [12] Granitzer, Michael; Sabol, Vedran; Onn, Kow Weng; Lukose, Dickson; Tochtermann, Klaus. 2010. "Ontology Alignment - A Survey with Focus on Visually Supported Semi-Automatic Techniques." *Future Internet* 2, no. 3: 238-258.
- [13] Monika Lanzenberger, Jennifer J. Sampson, Markus Rester, Yannick Naudet, Thibaud Latour, (2008) "Visual ontology alignment for knowledge sharing and reuse", Journal of Knowledge Management, Vol. 12 Iss: 6, pp.102 - 120
- [14] Lanzenberger, M.; Sampson, J., "AlViz - A Tool for Visual Ontology Alignment," *Information Visualization, 2006. IV 2006. Tenth International Conference on* , vol., no., pp.430,440, 5-7 July 2006 doi: 10.1109/IV.2006.18
- [15] Khriyenko O., Terziyan V., and Kaikova O., "End-user Facilitated Interoperability in Internet of Things: Visually-enriched User-assisted Ontology Alignment", In: International Journal on Advances in Internet Technology (InTech), IARIA, ISSN 1942-2652, 2013, Vol. 6, No. 1&2, pp. 90-100.
- [16] SimPack: A Generic Java Library for Similarity Measures in Ontologies (https://pdfs.semanticscholar.org/0019/21834ea5bb2767fdc8b6e083abd02fec6f5a.pdf)
- Mohammadi M., Ahooye Atashin A., Hofman W., Tan Y., "Comparison of ontology alignment algorithms across single matching task via the McNemar test" https://arxiv.org/abs/1704.00045, https://arxiv.org/pdf/1704.00045.pdf
- T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, M. Herschel, and A.-C. Ngonga Ngomo. Pushing the limits of instance matching systems: A semantics-aware benchmark for linked data. In WWW, pages 105106. ACM, 2015.
- T. Saveta, E. Daskalaki, G. Flouris, I. Fundulaki, and A. Ngonga-Ngomo. LANCE: Piercing to the Heart of Instance Matching Tools. In ISWC, 2015.
- Pavel Shvaiko, Jérôme Euzenat, Ernesto Jiménez-Ruiz, Michelle Cheatham, Oktie Hassanzadeh. **OM-2017: Proceedings of the Twelfth International Workshop on Ontology Matching**. OM 2017 - 12th International Workshop on Ontology Matching, Oct 2017, Wien, Austria. No commercial editor., pp.1-233, 2017, <http://om2017.ontologymatching.org>. <hal-01674668> https://hal.archives-ouvertes.fr/hal-01674668/file/ISWC2017-OM-ws.pdf

References

- Peter Ochieng and Swaib Kyanda. 2018**. Large-Scale Ontology Matching: State-of-the-Art Analysis**. *ACM Comput. Surv.* 51, 4, Article 75 (July 2018), 35 pages. http://disi.unitn.it/~pavel/OM/articles/Ochieng_CSUR_2018.pdf
- E. Thiéblin, O. Haemmerlé, N. Hernandez, C. Trojahn: **Survey on complex ontology matching** SWJ, 2018. http://disi.unitn.it/~pavel/OM/articles/Thieblin_SWJ_2018.pdf
- AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. **Ontology Matching: A Machine Learning Approach.** https://homes.cs.washington.edu/~pedrod/papers/hois.pdf
- Prodromos Kolyvakis, Alexandros Kalousis, Dimitris Kiritsis. **DeepAlignment: Unsupervised Ontology MatchingWith RefinedWord Vectors.** https://aclweb.org/anthology/N18-1072
- Ikechukwu Nkisi-Orji, Nirmalie Wiratunga, Stewart Massie, Kit-Ying Hui, and Rachel Heaven. **Ontology alignment based on word embedding and random forest classification**. http://www.ecmlpkdd2018.org/wp-content/uploads/2018/09/682.pdf
- Matthias Jurisch, Bodo Igler. **RDF2Vec-based Classification of Ontology Alignment Changes**. https://arxiv.org/pdf/1805.09145.pdf
- Soulakshmee Nagowah, Hatem Ben Sta, Baby A Gobin. **An Overview of Semantic Interoperability Ontologies and Frameworks for IoT**. https://www.researchgate.net/publication/329952889_An_Overview_of_Semantic_Interoperability_Ontologies_and_Frameworks_for_IoT/references
- Fatima Ardjani, Djelloul Bouchiha, Mimoun Malki. **Ontology-Alignment Techniques: Survey and Analysis**. International Journal of Modern Education and Computer Science 7(11):67-78 (2015). https://www.researchgate.net/publication/288683728_Ontology-Alignment_Techniques_Survey_and_Analysis
- Imene Ouali, Faiza Ghozzi, Raouia Taktak, Mohamed Saifeddine Hadj Sassi. **Ontology Alignment using Stable Matching**. ELSEVIER Procedia Computer Science Volume 159, 2019, Pages 746-755. https://www.sciencedirect.com/science/article/pii/S1877050919314164
- Yuan He, Jiaoyan Chen, Denvar Antonyrajah, Ian Horrocks. **BERTMap: A BERT-based Ontology Alignment System.** https://arxiv.org/abs/2112.02682
- Vivek Iyer, Arvind Agarwal, Harshit Kumar. **VeeAlign: Multifaceted Context Representation Using Dual Attention for Ontology Alignment.** https://aclanthology.org/2021.emnlp-main.842/