

This is the peer reviewed version of the following article:

Melis: an incremental method for the lexical annotation of domain ontologies / Bergamaschi, Sonia; P., Bouquet; D., Giacomuzzi; Guerra, Francesco; Po, Laura; Vincini, Maurizio. - In: INTERNATIONAL JOURNAL ON SEMANTIC WEB AND INFORMATION SYSTEMS. - ISSN 1552-6283. - STAMPA. - 3:3(2007), pp. 57-80. [10.4018/jswis.2007070103]

Terms of use:

The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

11/01/2026 20:18

MELIS: an incremental method for the lexical annotation of domain ontologies^{*}

Sonia Bergamaschi[†], Paolo Bouquet[‡], Daniel Giacomuzzi[‡], Francesco Guerra[#],
Laura Po[†], and Maurizio Vincini[†]

[†]DII-Università di Modena e Reggio Emilia
via Vignolese 905, 41100 Modena
firstname.lastname@unimore.it

[‡]DIT - Università di Trento
Via Sommarive, 14 – 38050 Trento, Italy
firstname.lastname@unitn.it

[#]DEA-Università di Modena e Reggio Emilia
v.le Berengario 51, 41100 Modena
firstname.lastname@unimore.it

Abstract. In this paper, we present MELIS (Meaning Elicitation and Lexical Integration System), a method and a software tool for enabling an incremental process of automatic annotation of local schemas (e.g. relational database schemas, directory trees) with lexical information. The distinguishing and original feature of MELIS is its incrementality: the higher the number of schemas which are processed, the more background/domain knowledge is cumulated in the system (a portion of domain ontology is learned at every step), the better the performance of the systems on annotating new schemas.

MELIS has been tested as component of MOMIS-Ontology Builder, a framework able to create a domain ontology representing a set of selected data sources, described with a standard W3C language wherein concepts and attributes are annotated according to the lexical reference database. We describe the MELIS component within the MOMIS-Ontology Builder framework and provide some experimental results of MELIS as a standalone tool and as a component integrated in MOMIS.

1 Introduction

The growth of information available on the Internet has required the development of new methods and tools to automatically recognize, process and manage information available in web sites or web-based applications. The aim of the semantic web is to build a web of data by providing a common framework which enables data sharing and reuse across application, enterprise, and community boundaries. The Semantic Web relies on the use of shared schemas and ontology

^{*} This work was partially supported by MIUR co-funded project WISDOM (<http://www.dbgroup.unimo.it/wisdom>).

which should provide a well-defined basis of shared meanings for data integration and reuse.

However, practical experience in developing semantic-enabled web applications and information systems shows that the simple and intriguing vision sketched above is not a solution to all problems. In particular, we stress the following issues:

- selecting an appropriate ontology for describing an application’s data may be very difficult. Indeed, engineering a new ontology from scratch can be extremely time consuming, and expensive and requires appropriate skills; finding a pre-existing ontology which perfectly fits local data is very unlikely, as most available ontologies are either too generic (and therefore semantically poor) or too specific (and therefore not suited for data different from those of the original application). Moreover, there is no standard recommendation or specification for referencing ontologies in information sources and different tools use different languages and techniques to add annotations. Several proposals and tools have been developed for including references to ontologies in HTML pages. However, such operation is typically executed off-line by adding “annotations” to the sources.
- because of the intrinsically distributed nature of knowledge on the web, different applications may refer to different ontologies to specify the meaning of their data.

The two issues above led the Semantic Web and Database communities to address two very hard problems: *ontology learning* (inducing ontologies from data/schemas) and *ontology matching/integration* (bridging different ontologies). There is a vast literature on these topics, and we will review part of it in Section 5. However, we observe that several methods and tools developed to address the two problems rely – in different ways – on the use of lexical information. The reason is simple: beyond the syntactic and semantic heterogeneity of schemas and ontologies, it is a fact that their elements and properties are named using natural language expressions, and that this is done precisely because they bring in useful (but often implicit) information on the intended meaning and use of the schema/ontology under construction. Therefore, it should not come as a surprise that a large number of tools for ontology learning and schema/ontology matching includes some lexical resource (mainly WordNet¹) as a component, and uses it in some intermediate step to annotate schema elements and ontology classes/properties with lexical knowledge. To sum up, lexical annotation seems to be a critical task to develop smart methods for ontology learning and matching.

In this context, we developed MELIS (**M**eaning **E**licitation and **L**exical **I**ntegration **S**ystem), an incremental method and a software tool for the annotation of data sources. MELIS is based on the integration and the extension of the lexical annotation module of the MOMIS-Ontology Builder² [7, 4] and some components from CTXMATCH2.0, a tool for eliciting meaning and matching pairs of

¹ See <http://wordnet.princeton.edu> for more information on WordNet.

² See <http://www.dbgroup.unimore.it> for references about the MOMIS project.

nodes in heterogeneous schemas, using an explicit and formal representation of their meaning [9, 10]. CTXMATCH2.0 was extended with respect to [9, 10] with a set of heuristic rules to generate new annotations on the basis of the knowledge provided by a given set of annotations; *WNEditor* was modified in order to jointly work with CTXMATCH2.0, by providing a customized lexical database.

The distinguishing feature and the novelty of MELIS is its incremental annotation method: the more sources (including a number of different schemas) are processed, the more background/domain knowledge is cumulated in the system, the better the performance of the systems on new sources.

MELIS supports three important tasks: (1) the source annotation process, i.e. the operation of associating an element of a lexical reference database (WordNet in our implementation, but the method is independent from this choice) to all source elements, (2) the customization of the lexical reference with the introduction of new lexical knowledge (glossa, lemma and lexical relationships), and (3) the extraction of lexical/semantic relationships across elements of different data sources.

The outline of the paper is the following: section 2 introduces the MELIS motivation and method. MELIS effectiveness is evaluated in section 3. Section 4 describes how MELIS improves the features of the MOMIS Ontology Builder, a framework able to create a domain ontology represented a set of selected data sources: a running example is provided; section 5 describes some related works and finally in section 6 we sketch out some conclusions and future works.

2 MELIS: the lexical knowledge component

In most real world applications, ontology elements are labeled by natural language expressions. In our opinion, the crucial reason for this aspect of ontology engineering is the following: while conceptual annotations provide a specification of how some terminology is used to describe some domain (the standard role of OWL ontologies), natural language labels (lexical annotations) provide a natural and rich connection between formal objects (e.g. OWL classes and properties) and their *intended* meaning. The intuition is that grasping the intended interpretation of an ontology requires not only an understanding of the formal properties of the conceptual schema, but also knowledge about the meaning of labels used for the ontology elements. In other words, an OWL ontology can be viewed as a collection of *formal* constraints between terms, whose intended meaning also depends on lexical knowledge.

In most cases, lexical knowledge is used for annotating schema/ontology labels with lexical information, typically WordNet senses. However, lexical annotation is a difficult task, and making it accurate may require a heavy user involvement. Here are some of the reasons:

- **coverage:** a complete lexical database including all possible terms does not exist. WordNet, for example, contains a very large number of general terms, but does not cover specialized domains, whereas specialized lexical databases tend to disregard general terms;

- **polysemy**: in natural language, many terms are polysemous, namely may have many possible meanings. The choice of the specific meaning associated to the term is context dependent, and therefore this choice (called word sense disambiguation in NLP) is very difficult to automate;
- **compound terms**: schemas and ontologies are often labeled with compound nominal expressions, like “full professor”, “table leg”, “football team”. Compound terms do not appear in any lexical database, unless they form a stable compound (e.g. “station wagon”). Their annotation is therefore more difficult, as the choice of the right lexical meaning often depends on determine, it is difficult to associate meaning to the relationship between term in a compound term;
- **integration** a standard model/language for describing lexical databases does not exist. Consequently, it is difficult to integrate different lexical resources.

That is why several tools which were developed for annotating sources only provide a GUI for supporting the user in the manual execution of the task (see Section 5 for references). However, this manual work can be highly time consuming, and very tedious for humans.

MELIS supports the annotation process by automatically providing a candidate lexical annotation of the source terms as the combination of lexical knowledge (from WordNet) and domain knowledge (if available). In addition, MELIS uses the *WNEditor* [4] to support customized extensions of WordNet with missing words and senses.

In the following we describe the MELIS method, its heuristic rules and the main features of *WNEditor*.

2.1 The MELIS method

The way MELIS works is depicted in Figure 1. We start from a collection of data sources which cover related domains, e.g. hotels and restaurants. In general we do not assume that a domain ontology is initially available, though this may be the case. The process is a cycle which goes as follows:

1. a schema, which can be already partially annotated with lexical information, is given as input to MELIS, together with a (possibly empty) domain ontology (considered as a reference ontology for the system). Lexical information is extracted from WordNet which may be extended with words/senses which are not available by interacting with *WNEditor*;
2. the automatic lexical annotation process starts; its output is a partial annotation of schema elements, together with a list of discovered relations across different elements. This annotation, whose main rules are described below, is obtained by using two main knowledge sources: WordNet (for lexical senses and relations across them), and the reference ontology, if not empty (it provides non-lexical – domain dependent – relations across senses, e.g. between “hotel” and “price”). Pre-existing lexical annotations are not modified, as

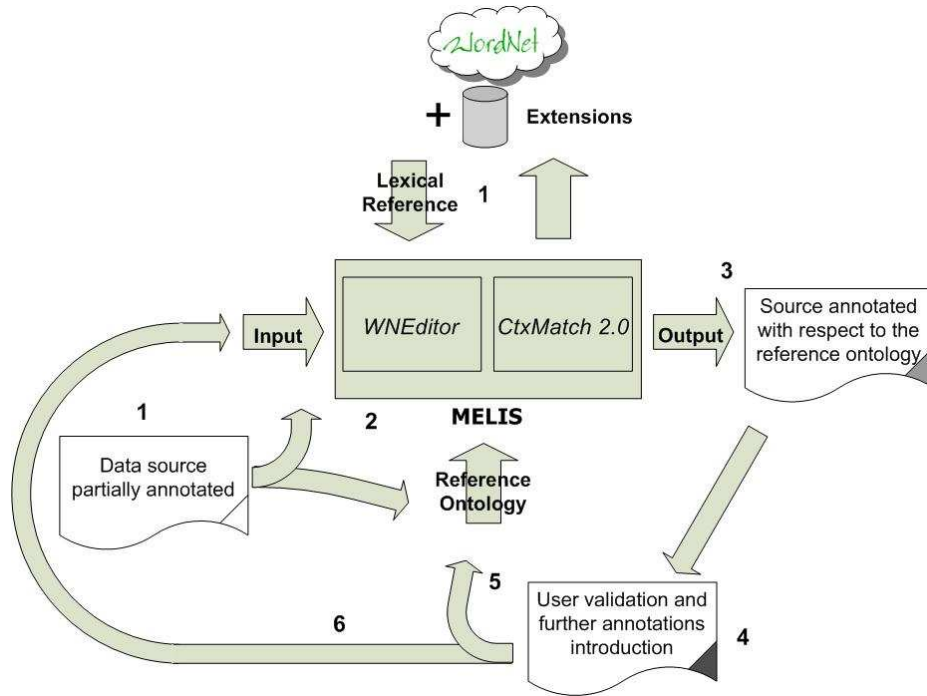


Fig. 1. Functional representation of MELIS

they may come either from manual annotation or from a previous annotation round;

3. the resulting annotated schema is passed to a user, who may validate/complete the annotation produced by MELIS;
4. the relations discovered across terms of the schema are added to the reference ontology (which means that an extended – and lexically annotated – version of the domain ontology is produced, even if initially it was empty);
5. the process restarts with the following schema, if any; otherwise it stops.

The process is incremental, as at any round the lexical database and the reference ontology may be extended and refined. As we said, the process might even start with an empty reference ontology, and the ontology is then constructed incrementally from scratch.

2.2 The rules for generating new annotations

A crucial part of the process has to do with the rules which are used to produce the MELIS lexical annotations. The core rules are derived from CTXMATCH2.0, and are described in previous publications. However, to improve the precision and recall of MELIS, we added a few specialized heuristic rules.

The annotation process takes as input a schema O and works in two main steps: first, for every label in O , the method extracts from WordNet all possible senses for the words composing the label; then, it filters out unlikely senses through some heuristic rules. The remaining senses are added as lexical annotation. Ideally, the system produces just one annotation, but in general it may be impossible to select a single annotation. Below is a general description of the heuristic rules used by MELIS. See appendix A for a graphical representation and an example of application of each rule.

To illustrate the MELIS heuristic rules, we will use the following notational conventions:

- Letters: capital letters (A, B, C, \dots) stand for class labels, low case letters (a, b, c, \dots) stand for datatype property labels, letters followed by “# n ” (where n is a natural number) refer to the n -th sense of the label for which the letter stands in WordNet (e.g. $b\#2$ is the synset associated to the second sense of the word occurring in the label “ b ”).
- Arrows: arrows denote a subclass relation when link two classes, arrows linking a class and a property denote datatype properties, dashed arrows denote object properties.
- Ontologies: O is used for the ontology to be annotated, DO_i for the i -th domain ontology available for the current elicitation process.

The elicitation process takes as input an ontology O and works in two main steps:

1. first, for each class and property label in O , the method extracts all candidate senses from WordNet, i.e. the candidate synsets consisting in sets of synonym words or collocations;
2. second, it filters out candidate senses following some heuristic rules, i.e. it selects the appropriate synset(s).

Below is a detailed description of the heuristic rules used by MELIS in the second step of the elicitation process. In Section 4.6 we provide a running example of their application on a specific domain (tourism), whereas in Appendix A, we provide a graphical representation of each rule, together with an intuitive example of its application.

Rule 1 *If in O we find a class labeled A with a datatype property b , and in some DO_i we find a class annotated as $A\#i$ with a datatype property annotated as $b\#j$, then we conclude that the annotations $A\#i$ and $b\#j$ are acceptable candidate annotations for A and b in O (see figure 6).*

Rule 2 *If in O we find a class labeled A with a datatype property b , and in some DO_i we find a class annotated as $B\#j$, with a datatype property annotated as $b\#k$ and a subclass $A\#i$, then we conclude that the annotations $A\#i$ and $b\#k$ are acceptable candidate annotations for A and b in O (see figure 7).*

Rule 3 *If in O we find a class labeled A with a datatype property b , and in some DO_i we find a class annotated as $A\#i$, with a subclass $B\#j$, and the latter has associated a datatype property annotated as $b\#k$, then we conclude that the annotations $A\#i$ and $b\#k$ are acceptable candidate annotations for A and b in O (see figure 8)³.*

Rule 4 *If in O we find a class labeled A with a datatype property b , and in some DO_i we find a class annotated as $C\#k$ with two subclasses annotated as $A\#i$ and $B\#j$, and there is a datatype property annotated $b\#h$ associated to $B\#j$, then we conclude that the annotations $A\#i$ and $b\#h$ are acceptable candidate annotations for A and b in O (see figure 9).*

Rule 5 *If in O we find a pair of classes labeled A and B , connected through any object property, and in DO_i we find a pair of classes annotated as $A\#i$ and $C\#k$, and $C\#k$ has a subclass $B\#j$, then we conclude that the annotations $A\#i$ and $B\#j$ are acceptable candidate annotations for A and B in O (see figure 10).*

Rule 6 *If in O we find a pair of classes labeled A and B (with B subclass of A), and in DO_i we find a pair of classes annotated as $A\#i$ and $B\#j$ (with $B\#j$ subclass of $A\#i$), then we conclude that the annotations $A\#i$ and $B\#j$ are acceptable candidate annotations for A and B in O (see figure 11).*

Rule 7 *If in O we find a pair of classes labeled A and B (with B subclass of A), and in some DO_i we find a subclass hierarchy in which two classes are annotated as $A\#i$, ..., $B\#j$ (with at least one intermediate class in between), then we conclude that the annotations $A\#i$ and $B\#j$ are acceptable candidate annotations for A and B in O (see figure 12).*

When all heuristic rules are applied, we discharge any candidate pair of annotations which is not supported by any of the rules above.

2.3 The *WNEditor*

WNEditor aids the designer during the creation of (additional) specific-domain lexicon addressing the issue of consistent extension of WordNet with specific domain knowledge.

³ Despite the first impression, this rule does not correspond to a form of inverse inheritance from child to parent nodes. The rule covers the case when in a domain ontology we find a property attached to the parent node and in the ontology to be annotated we find a pattern which corresponds to the property attached to the child node. This situation is very frequent. An intuitive example is given in the Appendix A. In a sense, this rule covers many situations in which the domain ontology and the ontology to be annotated are specified at a different level of granularity.

Extending WordNet WordNet is distributed *as-it-is* and external applications are not allowed to directly modify its data files. Therefore, *WNEditor* addresses two important issues:

1. providing a physical structure where WordNet and all its possible extensions are stored and efficiently retrieved;
2. developing a general technique which can support users in consistently extending WordNet.

The first issue is technically solved by storing the original WordNet (and all its possible extensions) in a relational database. The second issue is addressed by giving ontology designers the possibility to perform the following basic operations:

1. **Inserting new synsets.** To insert a new synset (i.e. meaning) for a term, the designer has to preliminary check whether such a synset already exists in the database. The *WNEditor* provides an *approximate matching technique* that computes *syntactic* and *semantic* similarity between the definitions associated to two synsets. The syntactic similarity function performs an approximate text match based on the edit distance or the name match[22]. The semantic similarity function exploits the heuristic known in literature as *definition match* approach. In particular, two different well-known IR techniques are implemented: *vector space model match*[2] and *latent semantic indexing match*[14].
2. **Inserting new lemmas.** We developed an *approximate string match* algorithm to perform the similarity search on the whole synset network based on the edit distance and on a *reverse index*, representing, at any time, the set of terms used within the reference ontology to build sense definitions.
3. **Inserting new relations.** *WNEditor* supports the designer in the definition of new relationships between synsets: given a source synset, the designer is assisted in searching for the most appropriate target synset. The implemented algorithm exploits synset definitions and the definition match heuristic for providing a list of candidate synsets the user has to confirm (see [4] for details).

Exporting WordNet extensions WordNet extensions may be exported and then reused in other applications. For this purpose, a basic technique supporting the sharing of different extension was developed:

- WordNet extensions are marked with the name of the data sources/ontology and the user inserting them; when user2 imports WordNet extended by user1, user1 extensions are temporary and user2 may decide what extensions have to be added to his local WordNet version.
- all users are allowed to include new lemmas, synsets and relations in their local WordNet version;

- the system includes in the exported version of an annotated source/ontology, both a code identifying the original WordNet version and a minimal subset of the extended annotations (i.e. it has to contain all and only the elements needed for rebuilding the new annotations starting from scratch);

3 MELIS Evaluation

A MELIS evaluation was done in the context of web directories. In particular, we selected the first three levels of a subtree of the Yahoo and Google directories (“society and culture” and “society”, respectively), which amounts to 327 categories for Yahoo and 408 for Google, arranged in two different subtrees.

MELIS results have been evaluated in terms of recall (the number of correct annotations made by MELIS divided by the total number of annotations, i.e. one for each category, as defined in a golden standard) and precision (the number of correct annotations retrieved divided by the total number of annotations retrieved). Moreover, since the algorithm is incremental, the evaluation is done after a first run and after a number of runs until a fixed point has been reached.

The process exploits knowledge provided by the initial annotation of the sources to generate the remaining annotations. Consequently, the initial set of annotations may highly affect the result. For this reason we considered eight different starting points:

1. **No Annotation:** the two subtrees are given to MELIS with no annotations at all.
2. **Y1-G0:** only the first level of the Yahoo subtree has been manually annotated.
3. **Y(1&2)-G0:** the Yahoo subtree have been manually annotated.
4. **Y0-G1:** only the first level of the Google subtree has been manually annotated.
5. **Y0-G(1&2):** the Google subtree have been manually annotated.
6. **Y1-G1:** the first level of the Yahoo and Google subtrees has been manually annotated.
7. **Y1-G0: WN enriched:** only the first level of the Yahoo hierarchy has been annotated. The annotator extended WordNet with 6 new terms and synsets to properly represent the subtree elements.
8. **Y(1&2)-G0: WN enriched:** the first two levels of the Yahoo hierarchy has been annotated. The annotator extended WordNet with 18 new terms and synsets to properly represent the subtree elements.

The MELIS method is supervised, i.e. the user may check the results calculated after each run and eventually correct the imprecise annotations. Such operation surely improves the result quality, but it is dependent on the user knowledge about the source domains and WordNet. Table 1 shows the evaluation results when MELIS is executed without any user intervention. As a consequence, the results we present are a kind of “worst case” scenario for MELIS.

	1st run		Fix point	
	Recall	Precision	Recall	Precision
1. No Annotation	22.90%	82.01%	24.08%	79.51%
2. Y1-G0	24.82%	85.28%	26.88%	85.85%
3. Y(1&2)-G0	74.45%	98.82%	76.96%	98.49%
4. Y0-G1	23.78%	78.54%	25.85%	77.78%
5. Y0-G(1&2)	73.41%	98.81%	74.89%	98.45%
6. Y1-G1	29.69%	85.17%	29.69%	85.17%
7. Y1-G0:WN enriched	26.29%	85.99%	28.36%	86.49%
8. Y(1&2)-G0:WN enriched	78.88%	98.89%	81.39%	98.57%

Table 1. MELIS evaluation

As described in section 2, MELIS associates a set of senses to each element and then on the basis of some rules, one or more appropriate senses are selected. Consequently, in Table 1 only the annotations perfectly fitting with the reference provided by the user are evaluated as correct annotations (when MELIS returns more than one annotation – possibly including the correct one – for a label, such result is taken as wrong). By analyzing Table 1, we observe that:

- the automatic execution of MELIS without any supervision generates at the fix point results in terms of precision not always better than the ones obtained after the first run. This is because some incorrect annotations, generated after the first run, propagate wrong knowledge in the following runs;
- the results are highly dependent on the input annotations (see for example case 3 and the symmetric case 5) and on the lexical database reference: by using a lexical database where specific terms and relationships are included improves the results (see the measures of cases 7 and 8).

In order to evaluate the results in case of a user assisted process, next table shows the recall and precision values obtained by considering an element as properly annotated if the annotation given by the user is included in the set of annotations calculated by MELIS.

By analyzing Table 2, we observe that:

- the results are very interesting and they allow us to hypothesize that a supervised MELIS use may provide very valuable support to the annotation task;
- the results allow us to show another way of using our tool: MELIS may suggest to the user a set of candidate annotations and among them it may indicate the most promising one. The user then can confirm such annotation.

Our experience shows that the annotation process supported by MELIS is less time-consuming and, in general, it converges to the final result after three runs.

	1st run		Fix point	
	Recall	Precision	Recall	Precision
1. No Annotation	50.22%	60.39%	53.03%	58.85%
2. Y1-G0	50.52%	61.73%	55.83%	62.38%
3. Y(1&2)-G0	88.48%	92.30%	93.80%	90.84%
4. Y0-G1	53.32%	64.35%	56.72%	63.79%
5. Y0-G(1&2)	79.76%	94.08%	82.72%	92.72%
6. Y1-G1	61.15%	66.45%	61.15%	66.45%
7. Y1-G0:WN enriched	52.29%	62.54%	57.61%	63.11%
8. Y(1&2)-G0:WN enriched	92.91%	92.64%	98.23%	91.22%

Table 2. MELIS evaluation - second test

4 MOMIS coupled with MELIS

We also tested the MELIS approach by coupling it with MOMIS. MOMIS is a framework that starts from a collection of data sources and provides a collection of tools for:

1. semi-automatically building a customized ontology which represents the information sources;
2. annotating each source according to the resulting ontology;
3. mapping the created ontology and the original sources into a lexical database (WordNet) to support interoperability with other applications.

MELIS has been experimented in MOMIS in order to improve the MOMIS methodology in two main directions: supporting the semi-automatic annotation of the original data sources (currently the process is manually executed), and providing methods for extracting rich relations across terms by exploiting lexical and domain knowledge.

MOMIS provides a double level of annotation for data sources and the resulting ontology: for each source, conceptual annotations map the original structure into a formalized ontology and lexical annotations assign a reference to a WordNet element for each source term. Moreover, the ontology structure is formalized by means of a standard model and each concept is annotated according to a lexical reference. MELIS inside MOMIS allows a greater automation in the process of source annotation, and provides a way for discovering relationships among sources elements.

Figure 2 shows the MOMIS architecture coupled with the MELIS component, where the process of creating the ontology and defining the mappings is organized in five step (each task number is correspondingly represented in figure 2) : (1) local source schema extraction, (2) lexical knowledge extraction performed with MELIS, (3) common thesaurus generation, (4) Global Virtual View (GVV) generation, and (5) GVV and local sources annotation. The following sections describe the details of these steps.

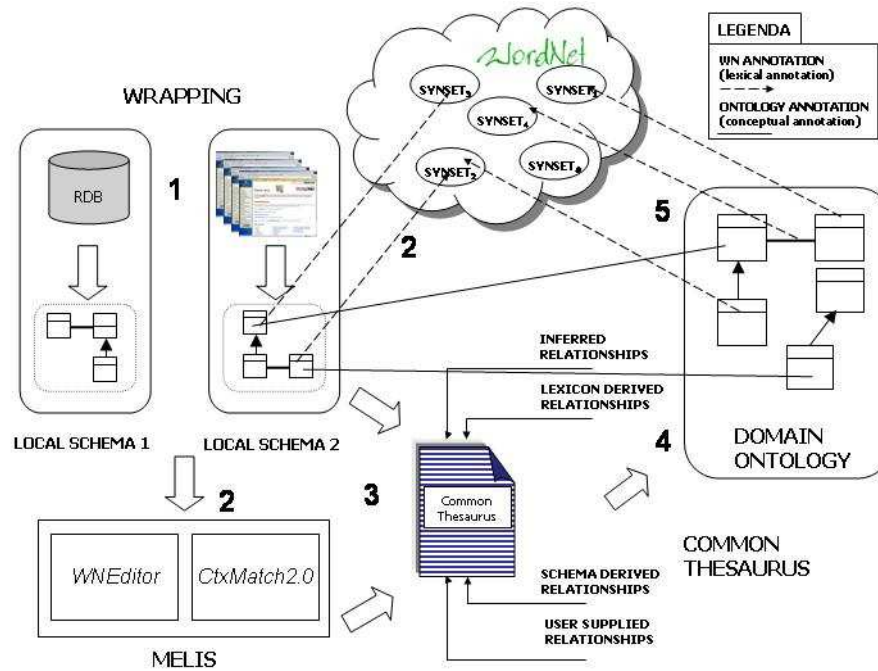


Fig. 2. Functional representation of MOMIS and MELIS

4.1 Local source schema extraction

To enable MOMIS to manage web pages and data sources, we need specialized software (wrappers) for the construction of a semantically rich representations of the information sources by means of a common data model. Wrappers in MOMIS logically converts the data source structure into the internal object language ODL_{I^3} , an extension of the standard ODMG-ODL [12] which may deal with semistructured information sources.

4.2 Lexical knowledge extraction

The extraction of lexical knowledge from data sources is typically based on an annotation process aiming at associating to each source element an effective WordNet meaning.

MELIS supports the user in this task by providing an effective tool for decreasing the boring manual annotation activity.

4.3 Common thesaurus generation

The common thesaurus is a set of relationships describing inter- and intra-schema knowledge about the source schemas. The following ODL_{I^3} relationships may be specified:

- SYN (Synonym-of), defined between two terms that are considered synonyms in every considered source.
- BT (Broader Terms), defined between two terms such as the first one has a broader, more general meaning than the second one. The opposite of BT is NT (Narrower Terms).
- RT (Related Terms) defined between two terms that are generally used together in the same context.

The common thesaurus is constructed through a process that incrementally adds four types of relationships: schema-derived relationships, lexicon derived relationships, designer-supplied relationships and inferred relationships.

- **Schema-derived relationships.** The system automatically extracts these relationships by analyzing each schema separately and applying a heuristic defined for the specific kind of source managed. For example, when analyzing XML data files, MOMIS generates BT and NT relationships from pairs of IDs and IDREFs.
- **Lexicon-derived relationships.** These relationships, generated by MELIS, represent complex relations between meanings of terms annotated with lexical senses. These relations may be inferred not only from lexical knowledge (e.g. by querying WordNet for relations across senses), but also from background knowledge (e.g. domain ontologies) which are available at the time of the annotation. As we have said before (section 2), at any step MELIS can (re)use any piece of ontology generated by the current extraction process as a source of domain knowledge to incrementally refine the extraction of new relations.
- **Designer-supplied relationships.** To capture specific domain knowledge, designers can supply new relationships directly.
- **Inferred relationships.** MOMIS exploits description logic techniques from ODB-Tools [6] to infer new relationships by applying subsumption computation to "virtual schemas" obtained by interpreting BT and NT as subclass relationships and RT as domain attributes.

4.4 GVV generation

The GVV consists of a set of classes (called Global Classes), plus mappings to connect the global attributes of each global class and the local sources attributes. Such a view conceptualizes the underlying domain; you can think of it as an ontology describing the sources involved.

Going into details, the GVV generation is a process where ODL_{I^3} classes describing the same or semantically related concepts in different sources are identified and clustered in the same global class by means of the ARTEMIS tool [11].

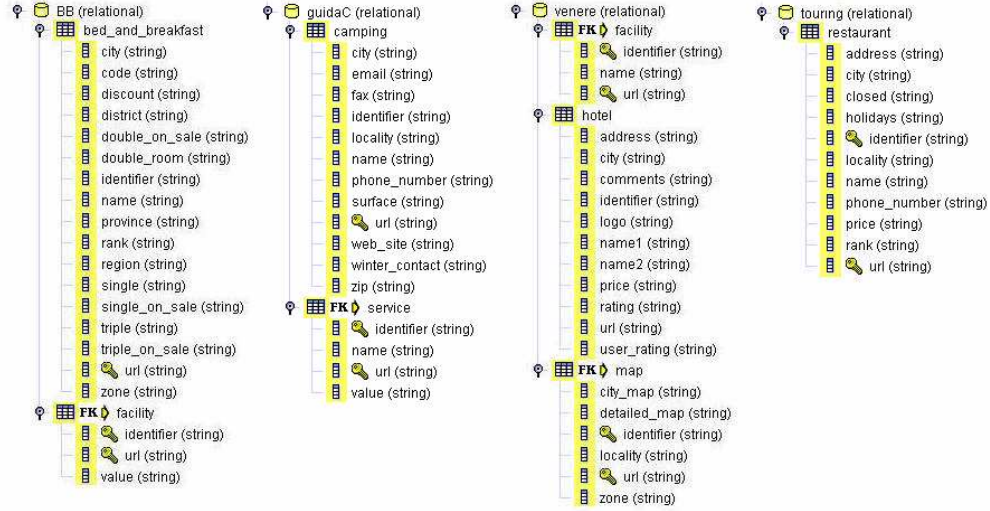


Fig. 3. the data source of the tourist domain

4.5 GVV and local sources annotation

MOMIS automatically proposes a name and meanings for each global class of a GVV [7]. Names and meanings have to be confirmed by the ontology designer. Local sources are conceptually annotated according to the created GVV.

4.6 Running example

We tested MELIS coupled with MOMIS by building an ontology for a set of data-intensive web sites containing data related to the tourist domain (see figure 3). The web sites were wrapped, and the corresponding data were structured and stored into four relational databases. The main classes extracted from the four sources are: **hotel** (from the “venere” database⁴), **restaurant** (from the “touring” database⁵), **camping** (from the “guidaC” database⁶) and **bed and breakfast** (the “BB” database⁷).

The incremental annotation process starts with the partial annotation of the data sources (notice that, in principle, this step can be skipped, which means that the entire work is delegated to automatic annotation); for some source elements, the ontology designer selects one or more corresponding WordNet meanings. Figure 4 shows some WordNet meanings and the lexical relationships among some data sources elements. In particular:

⁴ <http://www.venere.com>.

⁵ <http://www.touringclub.com>.

⁶ <http://www.guidacampeggi.com>.

⁷ <http://www.bbitalia.it>.

- **hotel#1** and **restaurant#1** are siblings, i.e. they have a common direct hypernym;
- **hotel#1**, **house#3**, **restaurant#1** are direct hyponyms of **building#1** (though we observe that this relationship may appear a bit misleading: typically restaurants are viewed as buildings, but rather as places where a service is provided);
- **bed_and_breakfast#1** is an hyponym of **building#1**;
- the closest hypernym that **campsite#1** and **building#1** share is **physical_object#1**, a top level synset in WordNet. This relationship does not allow the system to find lexical connections between the class “camping” and the other classes. Consequently, by means of the MELIS component *WNEditor*, a direct relationships between **campsite#1** and the hierarchy of **building#1** is introduced.

Notice that the choice of annotations can be tricky, even for simple structures as the ones we selected. For example, if we annotated the source element “camping” with the only WordNet meaning associated to the word “camping” (i.e. **camping#1**), we get a wrong meaning (in this context), as its gloss is “the act of encamping and living in tents in a camp”; whereas the intuitively correct synset in our context is **campsite#1**, defined as “the site where people can pitch a tent”. Finally, to test a larger number of implemented heuristic rules, **hotel#1** has been annotated as its hypernym: “building” through *WNEditor*.

The first annotated schema is then passed to MELIS both as an input and as a domain ontology. The tool starts the meaning elicitation process and produces a set of inferred lexical annotations of the schema elements. Figure 5 illustrates the results of a sample test of incremental annotation on one of our schemas. It shows the annotations manually provided by the ontology designer, a fraction of the new annotations generated after a first run of MELIS, and the additional annotations generated after a second run, when the outcome of the first run was provided as additional background knowledge in input; the numbers on the arrows refer to the heuristic rule which was used to generate the annotation. Notationally, a square near a class/attribute means that the element was manually annotated, a circle means that the element was automatically annotated after the first run, and a rhombus that it was incrementally annotated after the second run.

For illustration purposes, for every heuristic rule, we explain one of the generated annotations.

- Rule 1: the attribute “identifier” of the class “facility” in the source “VENERE” is annotated as **identifier#1** of the class “facility” in the source “BB”, since both the classes are annotated with the same synset.
- Rule 2: because of the hyponym relationships generated by the annotations of the classes “hotel”, “campsite”, “bed and breakfast” and “building”, the attribute “city” of the class “building” in the source “VENERE” produces the annotation of the same attribute in the sources “BB”, “touring” and “guidaC”.

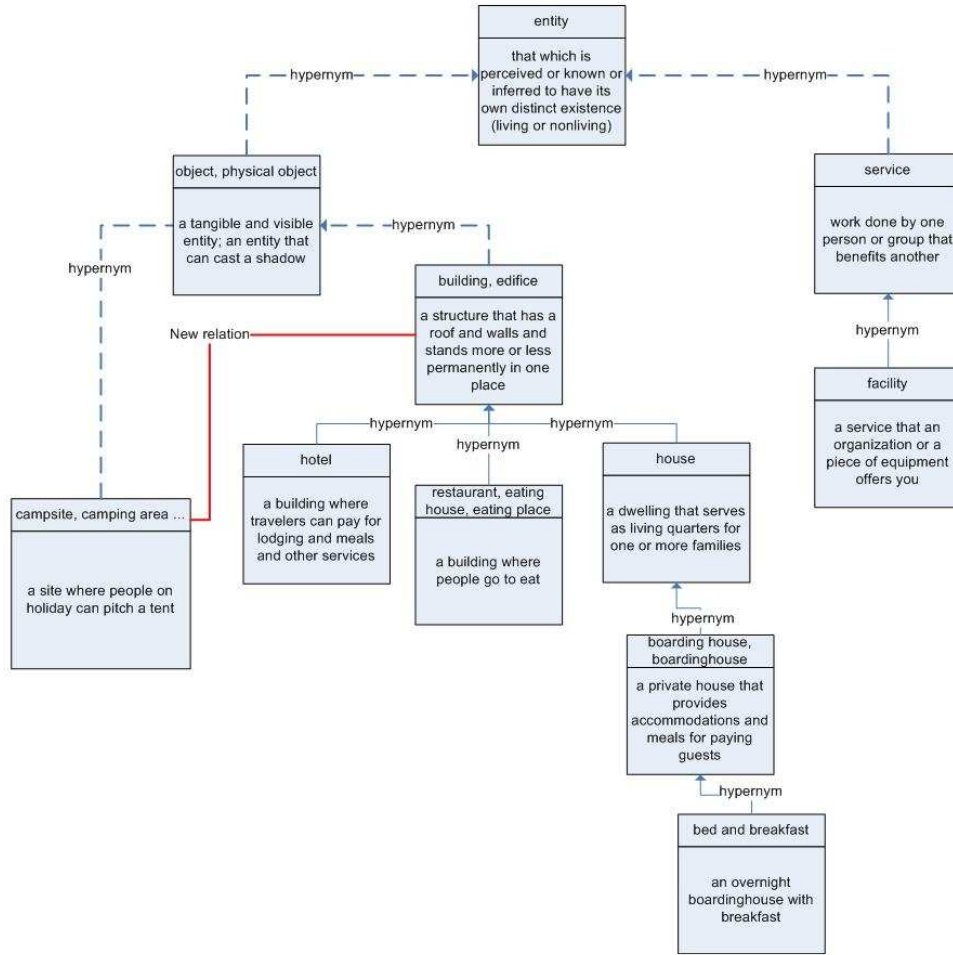


Fig. 4. Lexical relationships among the data sources elements

- Rule 3: because of the hypernym relation generated by the annotations of “building” and “bed and breakfast”, the attribute “identifier” of the class “bed and breakfast” in the source “BB” generates the annotation of the same attribute in the source “VENERE”. By executing a second run of the MELIS process, the attribute “identifier” on the class “building” generates the annotation of the same attribute on the classes “campsite” and “restaurant” of the sources “guidaC” and “touring” (application of Rule 2).
- Rule 4: because of the new relationship introduced in WordNet, **campsite#1** is a sibling of **restaurant#1**. Consequently, the attribute “locality” is annotated in the same way in the sources “guidaC” and “touring”.

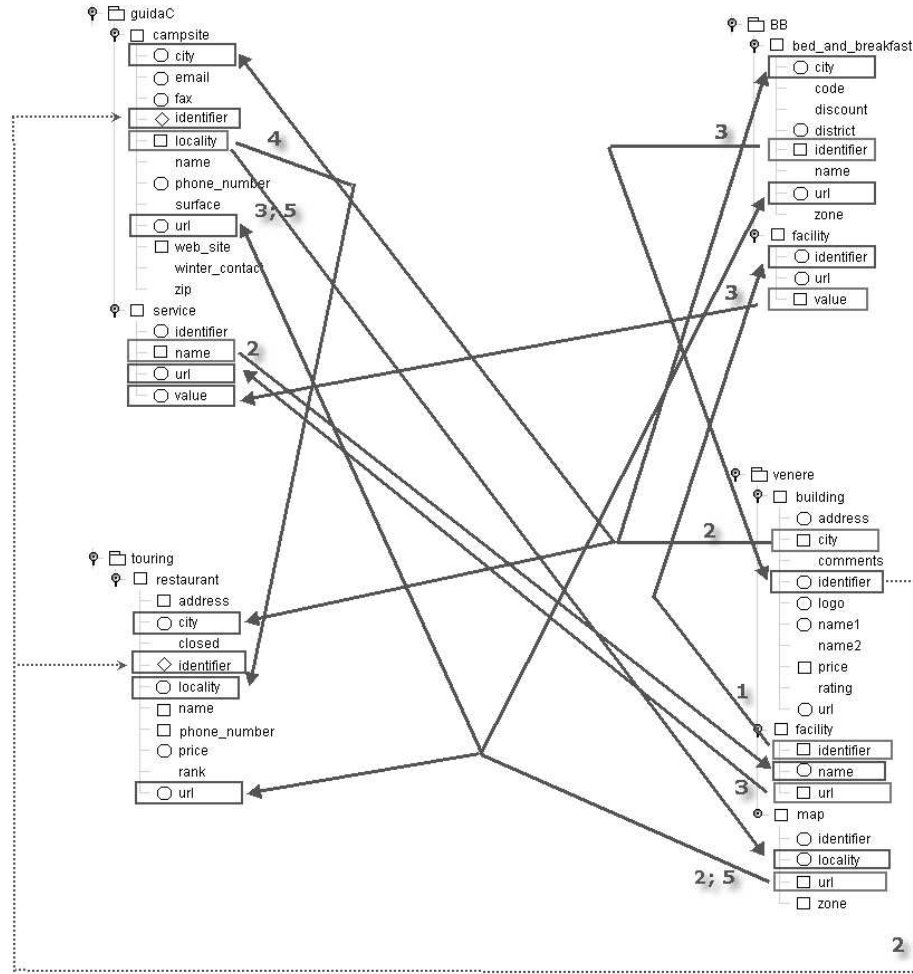


Fig. 5. Annotations generated by MELIS

- Rule 5: in the relational database there are foreign keys that represent a connection between two classes. In the source “VENERE”, the class “map” has a foreign key: the attribute “url” that refers to the class “hotel”. As this relationship joins with hierarchical relationships **hotel#1**, **campsite#1**, **bed_and_breakfast#1** and **building#1**, the attribute “url” of the class “map” in the source “VENERE” generates the annotation of the same attribute in the classes “campsite”, “bed and breakfast” and “restaurant” of the other sources.

The heuristic rules 6 and 7 are not exploited in our test. In fact, to fire these rules, we would need hierarchical structures.

5 Related Work

Works related to the issues discussed in this paper are in the area of languages and tools for annotations, techniques for extending WordNet, and systems for ontology management.

5.1 Languages and tools for annotations

In the research community, there is a substantial agreement about the “general” meaning of annotation, as the operation of inserting “extra information associated with a particular point in a document or other piece of information”⁸. On the other hand, different visions are expressed regarding the features and the information provided with such annotations and the languages for expressing them. For example in [25], annotations are classified on the basis of several dimensions: formal vs. informal, explicit vs. tacit, permanent vs. transient, published vs. private, and so on. The authors in [3] point out a problem regarding the lack of a common understanding of the annotation process: for this reason they refer to “the semantics of semantic annotation” as the provision of a consistent interpretation of assumptions that we make and the context within which such annotation should be interpreted.

In the Semantic Web, these differences are stressed because of the massive use of these statements. In this community, annotating generally means the operation of associating metadata with web resources. In particular, [3] qualifies such metadata as semantic metadata since they provide some indications about the contents of a resource. The proposed COHSE system includes three kinds of annotation: textual annotation, i.e. the process primarily targeted at human readers whereby notes or commentaries are added to the resources; link annotation, i.e. the content of the annotation is given by a link destination; semantic annotation, where the content of the annotation consists of some semantic information accessible to machine-processing. This last idea of semantic annotation has been pursued in Ontobroker [13], SHOE [21] and in the KIM Platform [29]. In [31] some further features of semantic annotations are introduced: in particular they are semantically interlinked, and they need to be congruent with ontology definition. This fact generates new issues, related to the manual execution of the annotation process and the management of changes in the ontologies that compels to have an annotation framework able to handle ontology creation, mapping and versioning. In [3] several types of annotation uses are specified: as decoration, i.e. for providing commentaries, as link, as instance identification, as instance reference, for specifying aboutness and pertinence.

A vision similar to our approach is described in [20], where a parallelism between annotation and mapping is highlighted. By means of the “deep annotation”, the authors define an annotation process that utilizes information proper, information structures and information context in order to derive mappings between information structures. In a similar way, our framework builds annotations for mapping local source contents into a domain ontology and a lexical reference.

⁸ <http://en.wikipedia.org/wiki/Annotation>.

Several tools supporting users in annotation have been developed. Among them, Annotea [24] is an open software that provides RDF metadata by means of a simple user interface. The annotation metadata can be stored locally or in one or more annotation servers. Annotea is part of the Semantic Web efforts at the World Wide Web Consortium.

5.2 Techniques for extending WordNet

Different extension of WordNet are proposed by many researchers.

Some researches aims at producing a formal specification of WordNet as an axiomatic theory. Among them, the OntoWordNet project [19] derives an ontology from WordNet. WordNet synset taxonomies and relations are reorganized and enriched, by extracting, interpreting and axiomatizing semantic relations implicitly encoded.

An other approach tries to classify WordNet synsets [26] using a method to create set of semantically similar WordNet synset and organizing them in categories. Such categories, as the context in MELIS, are then used for assigning the correct meaning to elements.

The Ontoling system, that jointly works with the Protégé editing tool, has been proposed [28]. The system goals are very similar to our purposes, as it provides a graphical interface for browsing linguistic resources (thesauri, dictionaries, wordnets...), linguistically enriching ontologies with elements from these linguistic resources, building new ontologies, starting from existing linguistic resources. Ontoling does not permit to add new relationships into the lexical database whereas in MELIS we allow the user to extend the WordNet relationships.

5.3 Ontology management

Several surveys about the approaches in the ontology management area have been published. This topic is generally divided into three categories: ontology development, ontology and schema matching and ontology alignment.

Concerning the ontology development, the ONTOWEB project published a complete technical report⁹ where tools are classified on the basis of the implemented methodologies (from scratch, reengineering ontologies, based on a cooperative construction, and managing the evolution). Several researches address topics in the ontology matching area, i.e. the techniques for identifying similar concepts in different ontologies: in [30] several systems are evaluated on the basis of the generated mappings (five kinds of criteria are identified), [27] focuses on mapping discovery, reasoning and representation. The ontology alignment, i.e. the automated resolution of semantic correspondences between the elements of heterogeneous ontologies, is one of the new challenge in the ontology management and it includes ontology mapping and schema mapping. The

⁹ <http://www.ontoweb.org> deliverable 1.4

Knowledgeweb Network of Excellence ¹⁰ has largely investigated about this issue publishing several reports.

Ontology and schema matching and ontology alignment tool are deeply analyzed in [23], where four phases for semantic matching are individuated:

1. Pre-integration preparation (normalization, lifting);
2. Similarity discovery;
3. Similarity representation (also includes reasoning);
4. Similarity execution (post-process).

For each stage, the methodologies adopted by the 38 analyzed tools/systems are compared against each other. The results offering a complete vision of the state of the art in this area are represented in table 3, where MOMIS has been included (no information about the pre-integration phase is given because it is a task typically executed by wrappers).

Tools/Systems	Similarity Discovery	Similarity Representation	Similarity Execution
Clio	x		x
COMA	x		
GLUE and iMAP	x		
OBSERVER	x		x
FCA-MERGE	x		
PROMPT	x		
MOMIS	x	x	x

Table 3. Classification of semantic integration system

Clio [32] is a research prototype providing to the user a graphical interface in order to support the creation of mappings between two data representations. There are many differences between Clio and MOMIS with MELIS: first, in the Clio framework the focus is on schema mapping issues, while the focus of our proposal is the semi-automatic generation of a “target” schema common to each source (the Global Virtual View). Moreover, our proposal relies on structural and lexical relationships among the sources.

COMA [15] (and COMA++ [1]) is a composite matcher, that provides an extensible library of different matchers and supports various aggregating and selecting strategies. Matchers exploit linguistic, data-type, and structural information, plus previous matches, to produce similarity matrix. Then particular similarity values are selected as good match candidates, which are combined to a single value. This process is executed for whole schemas or for two schema elements, and is repeated after the user provides feedback. COMA supports a reuse approach focusing on existing mappings, which can be generalized for different reuse granularities, or fragment- and schema-level match results. The starting

¹⁰ <http://knowledgeweb.semanticweb.org/>

mappings (or similarity) are user-defined, unlike our approach that is mainly focused on the use of lexical dictionaries (like WordNet) for semantic relationships discovering.

GLUE and iMAP [17] are an extension of LSD system [16] whose goal is to semi-automatically find schema mappings for data integration. Like its ancestor LSD, Glue use machine learning techniques to find mappings[18]. It first applies statistical analysis to the available data (joint probability distribution computation). Then generates a similarity matrix, based on the probability distributions, for the data considered and use "constraint relaxation" in order to obtain an alignment from the similarity, that is obtained by using probabilistic definition of several similarity measures. This approach relies on data instances techniques. On the other hand, the MOMIS methodology is based on schema analysis (we are experimenting the introduction of instance based components, see [5] for some preliminary results).

6 Conclusion and future work

In this paper we presented MELIS, an innovative tool for incrementally annotating data sources according to a lexical database (WordNet in our approach). MELIS exploits the annotation of a subset of source elements to infer annotations for the remaining source elements, thus strongly improving the activity of manual annotation. MELIS provides a component, *WNEditor*, for enhancing WordNet with new terms and relationships and a tool for generating lexical relationships between annotated source elements.

We performed some evaluation tests of MELIS by analyzing the recall and precision values in annotating the first two levels of the Google and Yahoo directory "Society" assuming an unsupervised process. As user supervision may substantially improve the quality of annotations that will be reuse by MELIS in subsequent runs, the results showed in the test may be considered as the minimum values achievable.

We experimented MELIS in conjunction with the MOMIS system in order to improve the MOMIS methodology for semi-automatically creating an ontology from a set of data sources. The first results show that MELIS and MOMIS working in conjunction are an effective and performative tool for creating a domain ontology. The testing was performed within the WISDOM project¹¹, for creating an ontology from several data-intensive websites about hotels and restaurants.

As we noticed in the introduction, MELIS can be used to provide valuable input not only for ontology learning, but also for ontology matching. Indeed, a large portion of tools for ontology matching make use of lexical knowledge (very often from WordNet) as part of the matching process¹², and thus they can greatly benefit from a tool like MELIS for the lexical annotation task and, if they need it,

¹¹ <http://www.dbgroup.unimo.it/wisdom>

¹² Instead of providing a long list of references, it is sufficient to point interested readers to the web site of the *Ontology Alignment European Initiative*

for the construction of a domain ontology from sources. Here we only recall that one of the MELIS building blocks, namely CTXMATCH2.0, was developed as a tool for matching schemas (the approach and the results are presented in [8]). As CTXMATCH2.0 uses both lexical information and domain knowledge to match schemas. The way it works is as follows. It takes as input two lexically annotated schemas, and uses the lexical senses to build a Description Logic formula which represents the meaning of each element in the schemas; the system computes mappings by deducing logical relations between pairs of formulas associated to pairs of nodes of different schemas (a mapping is a triple $\langle n_1, n_2, R \rangle$, where n_1 and n_2 are nodes belonging to different schemas, and R is a relation between the meaning associated to the two nodes, e.g. logical equivalence or subsumption); mappings are computed through a standard Description Logic theorem prover like Racer¹³, Pellet¹⁴, or FaCT¹⁵.

Future work on MELIS will be addressed on improving the annotation technique in order to deal with compound terms. Our aim is to split the term into its primitive words, find an annotation for each word and introduce new relationships linking the terms. Moreover, we will introduce in MELIS more accurate stemming techniques in order to improve the matching among input terms and the words of the lexical reference database. Finally, we are developing a methodology for building and sharing among a community new lexical database entries, e.g. by establishing how and when a new noun/meaning can be "promoted" to be part of the common lexical reference.

References

1. D. Aumüller, H. H. Do, S. Massmann, and E. Rahm. Schema and ontology matching with coma++. In *SIGMOD Conference*, pages 906–908, 2005.
2. R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.
3. S. Bechhofer, L. Carr, C. A. Goble, S. Kampa, and T. Miles-Board. The semantics of semantic annotation. In R. Meersman and Z. Tari, editors, *CoopIS/DOA/ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 1152–1167. Springer, 2002.
4. R. Benassi, S. Bergamaschi, A. Fergnani, and D. Miselli. Extending a lexicon ontology for intelligent information integration. In R. López de Mántaras and L. Saitta, editors, *ECAI*, pages 278–282. IOS Press, 2004.
5. D. Beneventano, S. Bergamaschi, S. Bruschi, F. Guerra, M. Orsini, and M. Vincini. "instances navigation for querying integrated data from web-sites". In *WEBIST 2006, Proceedings of the Second International Conference on Web Information Systems and Technologies, Setubal, Portugal, April 11-13, 2006*, pages 46–53, 2005.

(<http://oaei.ontologymatching.org/>), where the best available tools for matching are compared against a common testbed of ontologies and schemas. It is easy to verify that many tools make use of WordNet one way or another.

¹³ See <http://www.sts.tu-harburg.de/~r.f.moeller/racer/>.

¹⁴ See <http://www.mindswap.org/2003/pellet/>.

¹⁵ See <http://www.cs.man.ac.uk/~horrocks/FaCT>.

6. D. Beneventano, S. Bergamaschi, C. Sartori, and M. Vincini. ODB-QOPTIMIZER: A tool for semantic query optimization in oodb. In *Int. Conference on Data Engineering - ICDE97*, 1997. <http://www.dbgroup.unimo.it>.
7. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogeneous information sources. *Journal of Data and Knowledge Engineering*, 36(3):215–249, 2001.
8. P. Bouquet, L. Serafini, and S. Zanobini. Semantic coordination: A new approach and an application. In D. Fensel, K. Sycara, and J. Mylopoulos, editors, *The Semantic Web - ISWC 2003*, volume 2870 of *Lecture Notes in Computer Science (LNCS)*, pages 130–145, Sanibel Island (FL, USA), October 2003. Springer Verlag.
9. P. Bouquet, L. Serafini, and S. Zanobini. Peer-to-peer semantic coordination. *Journal of Web Semantics*, 2(1), 2005.
10. P. Bouquet, L. Serafini, S. Zanobini, and S. Sceffer. Bootstrapping semantics on the web: meaning elicitation from schemas. In *WWW2006 Conference Proceedings*. W3C, 2006.
11. S. Castano, V. De Antonellis, and S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. *IEEE Trans. Knowl. Data Eng.*, 13(2):277–297, 2001.
12. R. G. G. Cattell, editor. *The Object Database Standard: ODMG 3.0*. Morgan Kaufmann Publishers, San Mateo, CA, 2000.
13. S. Decker, M. Erdmann, D. Fensel, and R. Studer. Ontobroker: Ontology based access to distributed and semi-structured information. In Robert Meersman, Zahir Tari, and Scott M. Stevens, editors, *DS-8*, volume 138 of *IFIP Conference Proceedings*, pages 351–369. Kluwer, 1999.
14. S. Deerwester, S.T. Dumais, T.K. Landauer, G. W Furnas, and R. H Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.
15. H. H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *VLDB*, pages 610–621, 2002.
16. A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, pages 509–520, 2001.
17. A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Learning to map between ontologies on the semantic web. In *WWW*, pages 662–673, 2002.
18. A. Doan, J. Madhavan, P. Domingos, and A. Y. Halevy. Ontology matching: A machine learning approach. In *Handbook on Ontologies*, pages 385–404. 2004.
19. A. Gangemi, R. Navigli, and P. Velardi. The ontowordnet project: Extension and axiomatization of conceptual relations in wordnet. In R. Meersman, Z. Tari, and D. C. Schmidt, editors, *CoopIS/DOA/ODBASE*, volume 2888 of *Lecture Notes in Computer Science*, pages 820–838. Springer, 2003.
20. S. Handschuh, S. Staab, and R. Volz. On deep annotation. In *WWW*, pages 431–438, 2003.
21. J. Heflin, J. A. Hendler, and S. Luke. Shoe: A blueprint for the semantic web. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web*, pages 29–63. MIT Press, 2003.
22. E. H. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. *Proceedings of the International Conference on Language Resources and Evaluation (LREC)*, pages Granada, Spain, May 28–30, AAAI Press, 1998.
23. Y. Kalfoglou, B. Hu, D. Reynolds, and N. Shadbolt. Semantic integration technologies survey. In *Technical Report, ECS, University of Southampton*, 2005.

24. M. R. Koivunen. Eswc 2005, usersweb workshop. 2005.
25. C. C. Marshall. Toward an ecology of hypertext annotation. In *Hypertext*, pages 40–49. ACM, 1998.
26. A. Montoyo, M. Palomar, and G. Rigau. Wordnet enrichment with classification systems. In *In Proc. of WordNet and Other Lexical Resources: Applications, Extensions and Customisations Workshop, NAACL-01*, pages 101–106, Carnegie Mellon Univ., Pittsburgh, USA, 2001.
27. N. F. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.
28. M. T. Pazienza and A. Stellato. An open and scalable framework for enriching ontologies with natural language content. In M. Ali and R. Dapoigny, editors, *IEA/AIE*, volume 4031 of *Lecture Notes in Computer Science*, pages 990–999. Springer, 2006.
29. B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. Kim - semantic annotation platform. In Dieter Fensel, Katia P. Sycara, and John Mylopoulos, editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 834–849. Springer, 2003.
30. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.
31. S. Staab, A. Maedche, and S. Handschuh. An annotation framework for the semantic web. In *S. Ishizaki (ed.), Proc. of The First International Workshop on MultiMedia Annotation. January. 30 - 31. Tokyo, Japan, 2001.*
32. L. L. Yan, R. J. Miller, L. M. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 30(2):485–496, 2001.

A Appendix: heuristic rules description

In this appendix we provide a graphical representation of each rule we introduced in Section 2.2, together with an intuitive example of its application.

Rule 1

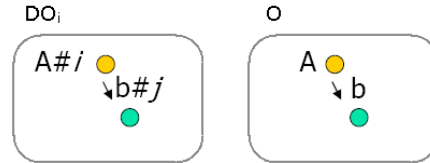


Fig. 6. Graphical representation of rule 1

For example, let us consider a class labeled **individual** with a datatype property **address**, and a DO_i where a class **person** is annotated as **person#1** with a datatype property **address** annotated as **address#2**. The application of this rule

generates the annotation **person#1** for the class **individual** and **address#2** for its datatype property **address**, since **individual** and **person** are part of the synonym words associated to **person#1**.

Rule 2

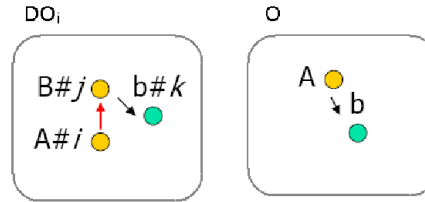


Fig. 7. Graphical representation of rule 2

For example, let us introduce a class labeled **student** with a datatype property **address**, and let DO_i contain a class **person** annotated as **person#1** with a datatype property **address** annotated as **address#2**, and a subclass **student** annotated as **student#1** (notice that the class **student** is hyponym¹⁶ of **enrollee**, which is hyponym of **person**). The application of this rule generates the annotation **student#1** for the class labeled **student**, and **address#2** for its datatype property.

Rule 3

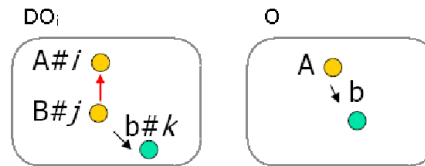


Fig. 8. Graphical representation of rule 3

¹⁶ In the WordNet terminology, we say that a noun *X* is an *hyponym* of a noun *Y* if *X* is less general than *Y* (*X* is a specialization of *Y*); conversely, we say that *X* is an *hypernym* of *Y* if *X* is more general than *Y*. Other relations across nouns are: *X* is a *holonym* of *Y* (*Y* is a part of *X*), and *X* is a *meronym* of *Y* (*X* is a part of *Y*). Different relations are used for other grammatical types, e.g. for verbs and adjectives. See <http://wordnet.princeton.edu> for more details.

For example, let us consider a class labeled **prof** with a datatype property **email**, and let us suppose that DO_i contains a class **professor** annotated as **professor#1** and a subclass **fullprofessor** annotated as **fullprofessor#1** with a datatype property **email** annotated as **email#1**. The application of this rule generates an annotation **professor#1** for the class labeled **prof**, and **email#1** for its datatype property.

Rule 4

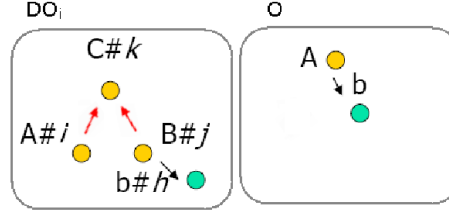


Fig. 9. Graphical representation of rule 4

For example, let us introduce a class labeled **hotel** with a datatype property **name**, and let DO_i contain a class **building** annotated as **building#1** with two subclass, the first is a class annotated as **hotel#1**, the latter is class annotated as **restaurant#1** with a datatype property **name** annotated as **name#1**. The application of this rule generates the annotation **hotel#1** for the class labeled **hotel**, and **name#1** for its datatype property.

Rule 5

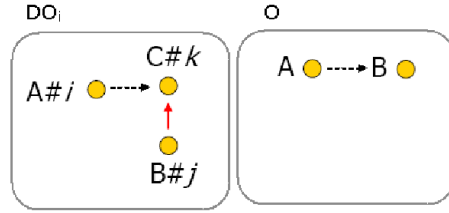


Fig. 10. Graphical representation of rule 5

For example, imagine in O we have a class labeled **restaurants** connected to a class named **seafood** by any object property (e.g. **serves**), and suppose DO_i

contains a class **restaurants** annotated as **restaurant#1** connected via some object property to a class **food** annotated as **food#2**, which in turn has a subclass **seafood** annotated as **seafood#1**. Then we conclude that **restaurant#1** and **seafood#1** are good candidates for the annotation of **restaurants** and **seafood** in *O*.

Rule 6

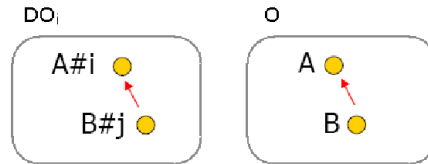


Fig. 11. Graphical representation of rule 6

For example, let us consider a pair of classes labeled **person** and **client**, where **client** is subclass of **person**, let DO_i contain a pair of classes annotated as **person#1** and **client#2**, where **client#2** is a subclass of **person#1**. The application of this rule generates the annotation **person#1** and **client#2** for the classes **person** and **client**.

Rule 7

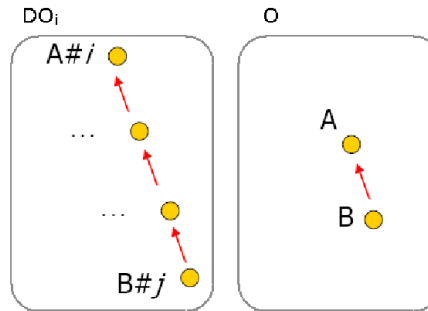


Fig. 12. Graphical representation of rule 7

For example, let us introduce a pair of classes labeled **animal** and **dog** (where **dog** is a subclass of **animal**), and let DO_i contain this subclass hierarchy: **animal#1**,

vertebrate#1, carnivore#1 and dog#1. The application of this rule generates the annotations animal#1 and dog#1 for the classes animal and dog.