A CORRESPONDENCE REPAIR ALGORITHM BASED ON WORD SENSE DISAMBIGUATION AND UPPER ONTOLOGIES

Angela Locoro
DIBE, Biophysical and Electronic Engineering Department, University of Genoa, Via Opera Pia 11/A, Genova, Italy
angela.locoro@unige.it

Viviana Mascardi
DISI, Computer Science Department, University of Genoa, Via Dodecanneso 35, Genova, Italy
viviana.mascardi@unige.it

Keywords: Ontology Matching, Upper Ontologies, Correspondences Repair, Word Sense Disambiguation.

Abstract: In an ideal world, an ontology matching algorithm should return all the correct correspondences (it should be complete) and should return no wrong correspondences (it should be correct). In the real world, no implemented ontology matching algorithm is both correct and complete. For this reason, repairing wrong correspondences in an ontology alignment is a very pressing need to obtain more accurate alignments. This paper discusses an automatic correspondence repair method that exploits both upper ontologies to provide informative context to concepts $c \in o$ and $c' \in o'$ belonging to an alignment a, and a context-based word sense disambiguation algorithm to assign c and c' their correct meaning. This meaning is used to decide whether c and c' are related, and to either keep or discard the correspondence $< c, c' > \in a$, namely, to repair a. The experiments carried on are presented and the obtained results are provided. The advantages of the approach we propose are confirmed by a total average gain of 11,5% in precision for the alignments repaired against a 2% total average error.

1 INTRODUCTION

Ontology matching is the process that, given two ontologies o and o' together with a set of parameters and optional inputs, returns a set of correspondences (an "alignment") between entities of o and entities of o'.

A "correspondence repair algorithm" takes an alignment a as input (with additional inputs and parameters if it is the case) and returns a repaired version of a, namely a', that has fewer wrong correspondences than a, hence achieving a higher precision[1].

Deciding that a correspondence $< c, c' >$ is wrong is an hard task. To this aim, the correspondence repair algorithm may exploit the structure of the two ontologies o and o' from which c and c' are drawn and a originates, or the constraints over $c \in o$ and $c' \in o'$ (such as disjointness, equivalence, etc), or the semantics of c and c' seen as meaningful pieces of information.

The correspondence repair algorithm that we discuss in this paper adopts the latter approach. In particular, the way we give a semantics to concepts c and c', in order to decide if they have the same meaning, is based on computational linguistic techniques.

According to (Agirre and Edmonds, 2007), *"in the field of computational linguistics, word sense disambiguation (WSD) is defined as the problem of computationally determining which "sense" of a word is activated by the use of the word in a particular context"*. The idea upon which our proposal roots is that *concepts in an ontology can be given a sense* in almost the same way as words in a text. The problem for making this idea work is finding a *context* for the concepts whose sense must be identified.

In our previous work, we implemented different algorithms that use upper ontologies [2] for boosting the ontology matching process (Mascardi et al., 2009). We run experiments with SUMO-OWL, a restricted

[1] Precision is the number of correctly found correspondences with respect to the reference alignment (true positives), divided by the total number of found correspondences (true positives and false positives) (Do et al., 2002).

[2] Quoting Wikipedia, an upper ontology (also named top-level ontology, or foundation ontology) is *"an attempt to create an ontology which describes very general concepts that are the same across all domains"*.

version of SUMO translated into OWL (Niles and Pease, 2001), OpenCyc, the open version of Cyc which is a commercial ontology (Lenat and Guha, 1990), and DOLCE (Gangemi et al., 2002), and we demonstrated that upper ontologies can be profitably used for enhancing either the precision or the recall of the obtained alignment, depending on the algorithm and the upper ontology used.

The goal of upper ontologies, namely supporting semantic interoperability among a large number of ontologies accessible "under" them, has been achieved: upper ontologies may provide an informative context for domain-dependent ontologies, and two domain-dependent ontologies may be matched with better results if they share the same context.

Starting from the above considerations, in this paper we exploit upper ontologies as a means to enrich the semantic context of concepts $c \in o$ and $c' \in o'$ belonging to an alignment a, and we use a context-based WSD algorithm to assign c and c' their correct meaning. This meaning is used to decide whether c and c' are related, and to either keep or discard the correspondence $< c, c' > \in a$, namely, to repair a.

The paper is organised as follows: Section 2 introduces the main definitions and surveys the state of the art in correspondence repair algorithms. Section 3 describes our algorithm for correspondence repair based on the Adapted Lesk approach and on upper ontologies, whereas Section 4 presents experiments and results. Section 5 concludes.

2 BACKGROUND

In this section we provide a short background on the topics upon which our research roots: WordNet, the Lesk algorithm for word sense disambiguation, ontology matching, and alignment repair.

2.1 WordNet

WordNet (Fellbaum, 1998) is a large lexical database of English, developed under the direction of George A. Miller. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept. A synset or synonym set is defined as a set of one or more synonyms that are interchangeable in some context without changing the truth value of the proposition in which they are embedded.

Most synsets are connected to other synsets via a number of semantic relations that include:
– **Semantic relations between nouns.** *Hypernyms*: Y is a hypernym of X if every X is a (kind of) Y ; *hyponyms*: Y is a hyponym of X if every Y is a (kind of) X; *coordinate terms*: Y is a coordinate term of X if X and Y share a hypernym; *holonym*: Y is a holonym of X if X is a part of Y ; *meronym*: Y is a meronym of X if Y is a part of X.
– **Semantic relations between verbs.** *Hypernym*: the verb Y is a hypernym of the verb X if the activity X is a (kind of) Y ; *troponym*: the verb Y is a troponym of the verb X if the activity Y is doing X in some manner; *entailment*: the verb Y is entailed by X if by doing X you must be doing Y ; *coordinate terms*: those verbs sharing a common hypernym.

While semantic relations apply to all members of a synset because they share a meaning but are all mutually synonyms, words can also be connected to other words through lexical relations, including antonymies (opposites of each other) and derivationally related, as well.

2.2 Lesk Algorithm

The Lesk algorithm is an algorithm for word sense disambiguation introduced by Michael E. Lesk in 1986 (Lesk, 1986).

It is based on glosses, which are definitions of each sense of a word, used to disambiguate a pair of words in a sentence by looking at the highest overlap (number of common words between the glosses).

In 2002, Satanjeev Banerjee and Ted Pedersen (Banerjee and Pedersen, 2002) adapted Lesk algorithm by using WordNet as the source of glosses. Hereafter we give a very general description of the main steps of Adapted Lesk approach:

1. take a sentence s, a target word tw to be disambiguated in s, and a context window cw of $2n+1$ words, with n words before and n words after tw.

2. for each word w in cw retrieve the set WN of Wordnet synonyms, hypernyms, hyponyms, holonyms, meronyms, troponyms, attributes, and all the associated glosses in WN, be them $g_1(w), g_2(w), ..., g_n(w)$;

3. for each pair of words wa, wb in cw and for each couple of glosses, $g_i(wa)$ and $g_j(wb)$, define an overlap as the longest sequence of common consecutive words: this overlap results in a score stating how much $g_i(wa)$ is the right definition for wa, and $g_j(wb)$ is the right definition for wb;

4. to disambiguate the target word tw, select the gloss in $g_1(tw), ..., g_m(tw)$ with highest score.

2.3 Ontology Matching

This section is based on (Euzenat and Shvaiko, 2007) and uses the same terminology adopted there.

Definition 1: Matching Process. A matching process can be seen as a function f which takes two ontologies o and o', a set of parameters p and a set of oracles and resources r, and returns an alignment A between o and o'.

Definition 2: Correspondence. A correspondence (or mapping) between an entity e belonging to ontology o and an entity e' belonging to ontology o' is a 5-tuple $< id, e, e', R, conf >$ where:

- id is a unique identifier of the correspondence;
- e and e' are the entities (e.g. properties, classes, individuals) of o and o' respectively;
- R is a relation such as "equivalence", "more general", "disjointness", "overlapping", holding between the entities e and e'.
- $conf$ is a confidence measure (typically in the $[0,1]$ range) holding for the correspondence between the entities e and e';

Definition 3: Alignment. An alignment of ontologies o and o' is a set of correspondences between entities of o and o'.

In our approach we only consider classes (c) as entities and equivalence as relation, thus we can drop R from the 5-tuple, as it is always "$\equiv$". Our repair algorithm does not exploit confidence and unique identifiers, so we drop them too for sake of readability. For the purposes of this paper, a correspondence is then a couple $< c, c' >$.

2.4 Alignment Repair

To the best of our knowledge, very few attempts to solve the problem of repairing correspondences between ontologies exist (Haeri et al., 2006; Meilicke et al., 2007; Meilicke et al., 2008a; Meilicke et al., 2008b).

The intuition behind (Haeri et al., 2006) is that an ontology alignment between o and o' gives a measure for similarity across the two ontologies, which can be seen as an estimate for the distance between each pair of points in two directed acyclic graphs g and g' embedded in a metric space. A set of techniques for "alignment refinement" is based on the definition of the partial order on the possible isomorphisms between g and g' and on a ranking of the possible alignments between o and o'.

The approach followed in (Meilicke et al., 2007) and (Meilicke et al., 2008a) is to interpret the problem of identifying wrong correspondences in an ontology alignment as a diagnosis task. The authors formalise correspondences as "bridge rules" in distributed description logics and analyse the impact of each correspondence on the ontologies it connects. A correspondence that correctly states the semantic relations between ontologies should not cause inconsistencies in any of the ontologies. If it does, the method computes sets of correspondences that jointly cause a symptom and repairs each symptom by removing correspondences from these sets.

In (Meilicke et al., 2008b) a supervised machine learning technique is presented as a means to learn disjointness axioms in lightweight ontologies, where a stronger axiomatization is needed in order to apply complex reasoning techniques to the debug of mappings.

To the best of our knowledge, WSD methodologies have not yet been exploited for boosting the mapping repair process and, among WSD techniques, the Adapted Lesk algorithm, that characterizes our approach, has been taken into consideration neither for ontology matching nor for mappings repair. The recent survey on the exploitation of WordNet in ontology matching (Lin and Sandkuhl, 2008) supports our claim: it demonstrates that the use of WordNet has become more and more widespread in ontology matching methodologies, but it never mentions the Adapted Lesk approach, though it is based on WordNet too.

3 ADAPTED LESK- AND UPPER ONTOLOGY-BASED CORRESPONDENCE REPAIR

Our approach to correspondence repair is based on the Adapted Lesk algorithm shortly discussed in Section 2.2. It also exploits one upper ontology *uo* and the WordNet thesaurus.

Suppose that we are given an alignment a between o and o', and we want to repair correspondences related to concept $c_1 \in o$. We first generate two partial alignments o-*uo* and o'-*uo* that will be used for providing a context to c_1 and to the concepts related to c_1 in the alignment a.

Based on that, three weighted context windows CW1, CW2 and CW3 are created. They contain:

- **CW1**
  1. all the concepts derived from all the correspondences $< c_1, c_{2i} >$ that belong to the alignment

a to be repaired (note that there may be more $c_{2i} \in o'$ such that $< c_1, c_{2i} > \in a$);

2. all the concepts derived from all the correspondences $< c_1, uoc_j >$ that belong to the alignment *o-uo* (again, there may be more $uoc_j \in uo$ such that $< c_1, uoc_j > \in$ *o-uo*).
   - CW1 has weight 1: it contains those concepts that are closer to c_1 because either there is a direct correspondence between c_1 and them in a, or there is a direct correspondence between c_1 and them in the *o-uo* alignment.

- **CW2**
  1. all the concepts in *uo* belonging to correspondences between concepts c_{2i} in CW1 and *uo*;
  2. all the concepts in *uo* directly related by means of "subClassOf" relation or other kinds of user-defined relation, to one $uoc_j \in uo$ belonging to CW1.
     - CW2 has weight 0.5: a step further has been moved from c_1
- **CW3** all the concepts in *uo* directly related to one $uoc_j \in uo$ belonging to CW2.
  - CW3 has weight 0.25: it contains even more distant concepts.

Figure 1 graphically shows the concepts in the three context windows, their distance from c_1, and their weights.

The concept c_1 is added to CW1, CW2 and CW3.

The following steps, based on Adapted Lesk, are then taken for each CW:

1. retrieve all the Wordnet synsets for c_1
2. for each synset of c_1, $syn(c_1)$
   (a) retrieve all its hypernyms (hype), hyponyms (hypo), holonyms (holo) and meronyms (mero)
   (b) build the list A of all these glosses, namely $allGlosses(syn(c_1, hype, hypo, holo, mero))$
3. for each concept c in CW such that $c \in$ CW and $c \neq c_1$
   (a) retrieve all its synsets and all the hype, hypo, holo and mero of all synsets
   (b) build the list B of all these glosses, namely $allGlosses(allsyn(c, hype, hypo, holo, mero))$
4. for each element of the list A
   (a) for each element of the list B
      i. compare each couple of glosses and compute of the longest consecutive sequence of common words
      ii. save the obtained score in a (A) x (B) matrix
5. for each synset of c_1 compute the max score for each c ($score(g_{max}(c))$ by extracting the max value of each (A) x (B) matrix.
6. save the max score in the CW matrix, with number of rows equal to the total number of synsets for c_1, and number of columns equal to the total number of context words in CW.

Finally, for each synset of c_1, sum by row the scores obtained from the matrixes associated with CW1, CW2, and CW3 by applying the weights corresponding to each context window. The following formula is an example of the computation for synset s_i of c_1, with $cw1_{ij}$, $cw2_{ij}$ and $cw3_{ij}$ as cells of the three CW matrixes:

$$s_i = \sum_{j=1}^{N} cw1_{ij} * 1 + \sum_{j=1}^{M} cw2_{ij} * 0.5 + \sum_{j=1}^{K} cw3_{ij} * 0.25 \quad (1)$$

The "winner" gloss for c_1, namely the gloss that is more likely to correctly define c_1, is the one associated to max s_i.

The same process is repeated for each $c_{2i} \in o'$ that belongs to the set of correspondences $< c_1, c_{2i} > \in a$, and the "winner" gloss for c_{2i} is found.

In the end, for each correspondence, a comparison of the "winner" glosses is done and classified in one of the following cases:

- the glosses are equal;
- one of the glosses is equal to that of a synset of either c_1 or c_2;
- one of the glosses is equal to that of a hype or hypo or holo or mero of the winner glosses of either c_1 or c_2;
- none of the glosses satisfies the above conditions.

The mapping $< c_1, c_2 >$ is discarded if and only if it falls under the last case otherwise it is kept.

3.1 A Practical Example

This section shows an example taken from one of our tests, to repair all the correspondences for the c_1 concept *Culture*. The set of correspondences (only one) in the final alignment for it is: $< Culture, Sculpture >$. Looking at the common radix of the two concepts (-ulture) there is a strong evidence that every string-based ontology matching method would have matched them, although they are not equivalent concepts. The creation of the three following context windows (CWs) for *Culture* results in the following:

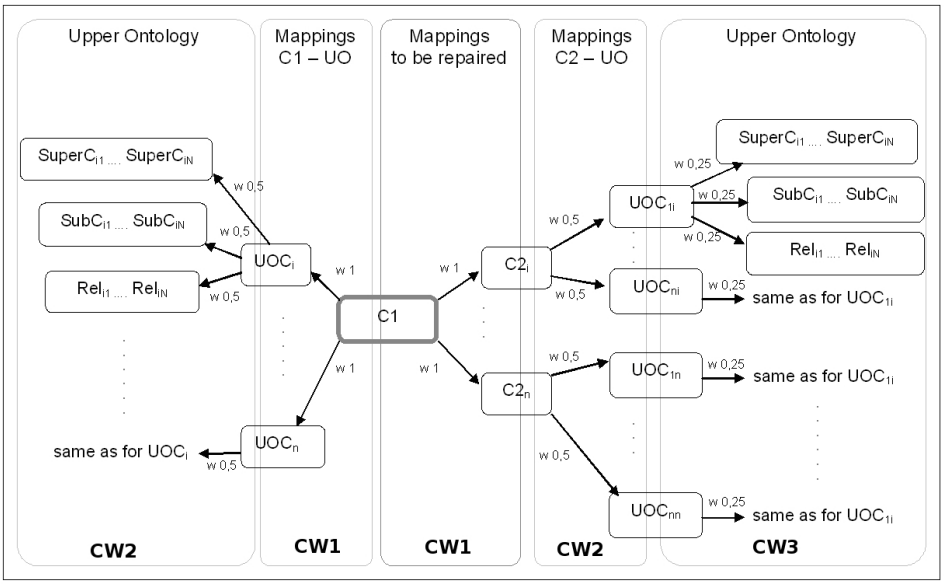

Figure 1: Weighted context windows.

- CW1(culture, sculpture, vulture), the concepts of the correspondence plus concepts of *uo* from c_1-*uo* mappings (namely vulture);
- CW2(culture, sculpture, art, work, bird), c_1 plus *uo* concepts from c_{2i}-*uo* mappings (namely sculpture, art, work), plus *uo* concepts related to c_1-*uo* in *uo* (namely bird);
- CW3(culture, art, work) c_1 plus *uo* concepts related to c_{2i}-*uo* in *uo* (namely art, work).
- The total number of synsets (Ss) for *Culture* is 7, with synset offset (the unique identifier for a Wordnet synset given its syntactic category, POS) 8287844, 5751794, 5984936, 920510, 14459824, 6194409, 917759.
- The number of hype, hypo, holo, mero for each of them is: (1,7,0,1), (1,3,0,0), (1,1,0,0), (1,1,0,0), (1,0,0,0), (1,3,0,0), (1,4,0,0) respectively.
- The number of Ss and related terms for the other words in the three CWs are:
  - for CW1: *sculpture* (2 S, 4 hype and 10 hypo), *vulture* (2 S, 2 hype and 4 hypo);
  - for CW2: *sculpture* (same as above), *art* (4 S, 4 hype and 43 hypo and 1 holo), *work* (7 S, 7 hype and 87 hypo), *bird* (5 S, 5 hype, 37 hypo and 20 mero);
  - for CW3: *art* (same as above), *work* (same as above).
- If we apply Equation (1) on the matrixes associated with CW1, CW2 and CW3, whose values are depicted in Table 1, the winner gloss for *Culture* turns out to be the one associated to S offset 14459824, with gloss *"a highly developed state of perfection; having a flawless or impeccable quality"*.

The disambiguation procedure of the word *Sculpture* is obtained considering all the correspondences in which *Sculpture* is involved, in our example $< Art, Sculpture >$ and $< Culture, Sculpture >$. The three CWs are:

- CW1(sculpture, culture, art), all the concepts from all the correspondences;
- CW2(sculpture, artery, art, work, vulture), c_2 plus *uo* concepts from c_{1i}-*uo* mappings (namely artery, art, work, vulture), plus *uo* concepts related to c_2-*uo* in *uo* (namely sculpture, which is redundant and has been deleted);
- CW3(sculpture, blood, vessel, artifact, art, work, bird) c_2 plus *uo* concepts related to c_{1i}-*uo* in *uo* (namely blood, vessel, artifact, art, work, bird).
- The number of Ss for *Sculpture* is 2, with S offset 4157320, with 2 hype and 9 hypo, and 937656, with 2 hype and 1 hypo. For the other CWs words they are:
  - for CW1: *culture* (7 S, 7 hype, 19 hypo and 1 mero), *art* (same as above);
  - for CW2: *artery* (2 S, 2 hype, 76 hypo), *art* (same as above), *work* (same as above), *vulture* (same as above);

Table 1: Score matrixes for word Culture.

	CW1		CW2				CW3	
S offset	sculpture	vulture	sculpture	art	work	bird	art	work
8287844	1	0	1	1	2	1	1	2
5751794	1	0	1	2	1	1	2	1
5984936	1	0	1	1	1	0	1	1
920510	1	0	1	1	2	1	1	2
14459824	3	1	3	3	2	1	3	2
6194409	1	0	1	2	2	1	2	2
917759	1	1	1	2	3	1	2	3

Table 2: Score matrixes for word Sculpture.

	CW1		CW2				CW3					
S offset	culture	art	artery	art	work	vulture	blood	vessel	artifact	art	work	bird
4157320	1	5	1	5	2	1	1	1	1	5	2	2
937656	2	25	1	25	2	0	1	1	1	25	2	0

  - for CW3: *blood*(5 S, 5 hype, 12 hypo), *vessel* (3 S, 3 hype, 50 hypo,10 mero and 1 holo), *artifact*(1 S, 1 hype and 45 hypo), *art* (same as above), *work* (same as above), *bird* (same as above).

- The final CW1, CW2 and CW3 matrixes are depicted in Table 2. According to them The winner gloss for *Sculpture* has result the one associated to S offset 937656, with gloss *"creating figures or designs in three dimensions"*.
- the repair procedure has then compared the two winner glosses and discarded the correspondence as a wrong one.

4 EXPERIMENTS AND RESULTS

In order to conduct our experiments we have used the alignments obtained from our previous work (Mascardi et al., 2009). A subset of the ten tests previously run has been selected, only considering the final alignments obtained matching the two ontologies via SUMO and OpenCyc. In particular, we have chosen the final alignments via *uo* obtained with two different matching algorithms: the first one, named NS, simply matches an ontology o with the *uo* comparing them directly; the other one, named WS, also considers the *uo* structure (sub- and super-concepts of the *uo* concept involved in the matching), in order to obtain the final correspondences. For each test also the partial alignments $o - uo$ and $o' - uo$ have been reused to build the context windows depicted in the previous section.

Our repairing algorithm is written in Java and uses the Alignment API [3], to manage the alignments as well as the OWL version of the *uo*. The Java library used to manage WordNet 3.0 is also embedded in the API. The whole mappings repair procedure, with the inclusion of the disambiguation step, based on the Adapted Lesk algorithm, has been implemented from scratch as no Java version of the algorithm seems to exist at the moment of writing.

Our complete mappings repair procedure implementation is as follows:

1. the *SplitAlignment* function is first called, it receives as input an alignment and splits it in all the subsets of the alignment corresponding to the mappings of concept c_1 from o with every concept c_{2i} from o' as well as the subsets of the inverse alignment (all the mappings of concept c_2 from o' with every concept c_{1i} from o)
2. the *MakeContextWindows* function is run on every alignment subset in order to generate the three context window for each concept, which contain all the context words listed in Section 3. The algorithm pre-processes each context window by deleting duplicate concepts and by removing those concepts that do not belong to WordNet. For composite concepts (concepts with more than one term, e.g. *ComputerHardware*) a tokenization procedure is carried on to split them into simple terms and for each term the three context windows are built, each containing any other term of the composite concept as a context word (referring to the above example the context windows for the term *Computer* will also contain the word *Hardware* and the same for the second term). If the composite concept contains stop words[4] we remove them.

[3] available at `http://alignapi.gforge.inria.fr/`

[4] The list is available at `http://ir.dcs.gla.ac.uk/resources/linguistic_utils/stop_words`.

3. the *AdaptedLesk* function provides the disambiguation task shown in Section 3. Before computing the maximum consecutive overlap each gloss has been treated with a stop words cleaning and a word stemming procedure (the algorithm we used to obtain a word stem from a term, namely a process for removing the commoner morphological and inflexional endings from words, is the Porter Stemmer, whose Java version is freely available at Porter's web page[5]). For this first prototype the score computation is limited to the longest overlap between two glosses. In case there is more than one overlap, only the best is considered and the other ones are not taken into account in the final score computation. The output winner gloss is in form of the WordNet offset of the synset containing the gloss as well as the label of the concept and the score of the gloss.

4. the *RepairMappings* function takes the original alignment as input and, for each mapping, makes the checks on the semantic compatibility between the winner glosses of the respective concepts as stated in Section 3. If one or both concepts are composite ones, a cross-checking is executed on the set of all the winner glosses for each term. The output of this final step is a repaired alignment, pruned of all those mappings for which the concept glosses resulted incompatible.

In the evaluation phase of our mappings repair algorithm we have calculated the precision of the repaired alignment against a reference alignment. The results of the evaluation are depicted in Table 3. The precision has been computed using the *precEval* function provided by the Alignment API. The **P** column shows the precision of the tests without any repairing procedure whereas the **PR** column represents the precision obtained from the repaired alignment after applying our approach. The names of the ontologies used is shown in each test column header, except for the extension that is always .owl.

As stated from the results most of the tests present a higher precision after the application of the adapted lesk repairing procedure. In particular, for both SUMO and OpenCyc WS methods, the comparison with old precision always results to be in favour of the repaired alignment. For SUMO NS method one test only gives a worse precision after the pruning process while for OpenCyc NS 2 tests out of 10 give no gain and 2 tests have a lower precision. The average gain in precision after our repair process using OpenCyc WS amounts at 20%, using OpenCyc NS is 3%, while for SUMO WS is 15%, and for SUMO NS is 8%. The total average gain is 11,5%. The total average error estimation, meaning that the algorithm discards correct correspondences (false negatives) amounts at 2%.

A comparative evaluation with the other approaches depicted in section 2.4 would be difficult: methodologies are disparate, not every system is fully automatic as our and experiments were conducted on

[5] http://tartarus.org/~martin/PorterStemmer/

Table 3: Comparing Precision in Tests Results.

	P	PR
Test1 - Ka, Bibtex		
SUMO, NS	0.67	0.83
SUMO, WS	0.38	0.46
OpenCyc, NS	0.71	0.83
OpenCyc, WS	0.56	0.83
Test2 - Biosphere, Top-bio		
SUMO, NS	0.00	0.00
SUMO, WS	0.16	0.57
OpenCyc, NS	0.00	0.00
OpenCyc, WS	0.03	0.50
Test3 - Space, Geofile		
SUMO, NS	0.45	0.64
SUMO, WS	0.14	0.34
OpenCyc, NS	0.55	0.62
OpenCyc, WS	0.21	0.48
Test4 - Restaurant, Food		
SUMO, NS	0.34	0.40
SUMO, WS	0.25	0.31
OpenCyc, NS	0.42	0.36
OpenCyc, WS	0.29	0.34
Test5 - MPEG7Genres, Subject		
SUMO, NS	0.42	0.54
SUMO, WS	0.28	0.50
OpenCyc, NS	0.47	0.48
OpenCyc, WS	0.20	0.44
Test6 - Travel, Vacation		
SUMO, NS	0.31	0.30
SUMO, WS	0.28	0.29
OpenCyc, NS	0.15	0.22
OpenCyc, WS	0.13	0.24
Test7 - Resume, Agent		
SUMO, NS	0.44	0.51
SUMO, WS	0.25	0.41
OpenCyc, NS	0.42	0.44
OpenCyc, WS	0.22	0.41
Test8 - Resume, HL7_RBAC		
SUMO, NS	0.57	0.62
SUMO, WS	0.28	0.49
OpenCyc, NS	0.71	0.70
OpenCyc, WS	0.28	0.54
Test9 - Ecology, Top-bio		
SUMO, NS	0.12	0.15
SUMO, WS	0.09	0.16
OpenCyc, NS	0.14	0.14
OpenCyc, WS	0.11	0.14
Test10 - Vertebrate, Top-bio		
SUMO, NS	0.29	0.40
SUMO, WS	0.07	0.15
OpenCyc, NS	0.67	0.67
OpenCyc, WS	0.16	0.28

test sets that are not comparable[6]. Moreover, in (Haeri et al., 2006) no experiments are presented. Nevertheless, it's worth mentioning that in (Meilicke et al., 2007; Meilicke et al., 2008a) the average gain in precision is 16% but the error rate is omitted, while in (Meilicke et al., 2008b) the average gain in precision is 16% (with average error rate 1%) and 18% (with average error rate 5%), using reference disjointness and learned disjointness respectively.

5 CONCLUSIONS AND FUTURE WORK

This paper focuses on a new automatic approach for repairing ontology alignments based on WSD techniques, in particular on the Adapted Lesk algorithm, and on upper ontologies used to enrich the context necessary to compute the meaning disambiguation of concepts and thus process the repairing task.

Our future work will focus on the extension of the context window for each concept of a mapping by exploiting the local context available in the input ontologies, such as for example sub- and super concepts structure as well as other related concepts. Free text comments attached to the concept can also contribute to the context construction.

Another extension we wish to implement, in order to lower the error rate, is the refinement of the scores computation, considering all the possible overlaps between two glosses in the final score composition, as well as introducing a threshold to filter more relevant winner glosses.

A comparative evaluation between our approach and the other existing completely automatic methodologies for mappings repair on the same test set is another activity that we are going to accomplish.

We also wish to exploit other WSD techniques for both mappings repair and ontology matching tasks. The design and implementation of an ontology matching procedure based on Adapted Lesk and the exploitation of local context as well as upper ontologies is on its way.

ACKNOWLEDGEMENTS

The work of the first author was partly supported by the KPLab project funded by EU 6th FP. The work of the second author was partly supported by the Italian research project Iniziativa Software CINI-FinMeccanica.

[6]Our test ontologies consist of 112 concepts on average. This dimension is greater than that of the benchmark ontologies used in the above studies, based on the OntoFarm Dataset, with 39 concepts on average.

REFERENCES

Agirre, E. and Edmonds, P. (2007). *Word Sense Disambiguation - Algorithms and Applications*. Springer.

Banerjee, S. and Pedersen, T. (2002). An adapted Lesk algorithm for word sense disambiguation using WordNet. In Gelbukh, A. F., editor, *CICLing 2002*, volume 2276 of *LNCS*, pages 136–145. Springer.

Do, H. H., Melnik, S., and Rahm, E. (2002). Comparison of schema matching evaluations. In Chaudhri, A. B., Jeckle, M., Rahm, E., and Unland, R., editors, *NODe 2002*, volume 2593 of *LNCS*, pages 221–237. Springer.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer.

Fellbaum, C., editor (1998). *WordNet – An Electronic Lexical Database*. The MIT Press.

Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., and Schneider, L. (2002). Sweetening ontologies with DOLCE. In Gómez-Pérez, A. and Benjamins, V. R., editors, *EKAW 2002*, volume 2473 of *LNCS*, pages 166–181. Springer.

Haeri, S. H., Hariri, B. B., and Abolhassani, H. (2006). Coincidence-based refinement of ontology matching. In *SCIS+ISIS 2006*.

Lenat, D. and Guha, R. (1990). *Building large knowledge-based systems*. Addison Wesley.

Lesk, M. (1986). Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86*, pages 24–26. ACM.

Lin, F. and Sandkuhl, K. (2008). A survey of exploiting wordnet in ontology matching. In Bramer, M., editor, *IFIP AI*, volume 276 of *IFIP*, pages 341–350. Springer.

Mascardi, V., Locoro, A., and Rosso, P. (2009). Automatic ontology matching via upper ontologies: A systematic evaluation. IEEE Trans. Knowl. Data Eng., to appear.

Meilicke, C., Stuckenschmidt, H., and Tamilin, A. (2007). Repairing ontology mappings. In *AAAI 2007*, pages 1408–1413. AAAI Press.

Meilicke, C., Stuckenschmidt, H., and Tamilin, A. (2008a). Reasoning support for mapping revision. *J. Logic and Computation*.

Meilicke, C., Völker, J., and Stuckenschmidt, H. (2008b). Learning disjointness for debugging mappings between lightweight ontologies. In Gangemi, A. and Euzenat, J., editors, *EKAW*, volume 5268 of *LNCS*, pages 93–108. Springer.

Niles, I. and Pease, A. (2001). Towards a standard upper ontology. In Welty, C. and Smith, B., editors, *FOIS 2001*, pages 2–9. ACM.