The PARDON WSD systen

Document Number	WP6.M
Project ref.	IST-2001-34460
Project Acronym	MEANING
Project full title	Developing Multilingual Web-scale Language Technologies
Project URL	http://www.lsi.upc.es/~nlp/meaning/meaning.html
Availability	Public
Authors: Jordi Atserias (UPC)	

INFORMATION SOCIETY TECHNOLOGIES

Project ref.	IST-2001-34460
Project Acronym	MEANING
Project full title	Developing Multilingual Web-scale Language Technologies
Security (Distribution level)	Public
Contractual date of delivery	March 2004
Actual date of delivery	February 28, 2005
Document Number	WP6.M
Type	Report
Status & version	v 1 Draft
Number of pages	15
WP contributing to the deliberable	WP 6
WPTask responsible	Eneko Agirre
Authors	Jordi Atserias (UPC)
Other contributors	
Reviewer	
EC Project Officer	Evangelia Markidou
Authors: Jordi Atserias (UPC)	
Keywords: WSD, Ontologies, WordNet, EuroWordNet, MCR, Semantic Parsing, Role Labeling	
Abstract: .	

Contents

1 Introduction **3**

2 Pardon's Architecture **4**
2.1 PARDON's Knowledge Representation 5
2.2 Model Application 5

3 CLP formalization **8**
3.1 Relaxation Labeling Algorithm 9
3.2 Determining the Initial State 10
3.3 Specific Constraints 10

4 Source Data **10**

5 Evaluation **12**

6 Results **13**

7 Conclusions **14**

8 Future Work **15**

1 Introduction

There is no doubt of the complexity of any human language, and the inherent difficulty of its understanding. A single coma can make a sentence to change completely the meaning (e.g. "*eats shoots and leaves*" versus "*eats, shoots and leaves*"[1]) or worst, to mean the opposite ("*Don't stop*" versus "*Don't, stop*" or "*Women, without her men, is nothing*" versus "*Women, without her, men is nothing*"). The aim of this work is to explore new architectures for processing natural language as robust and flexible as possible.

An important step in any process that implies Natural Language Understanding is Semantic Interpretation. *Semantic Interpretation* can be defined as the process of obtaining a suitable meaning representation for a text [Brill and Mooney, 1997].

In order to obtain that representation of a context independent meaning of a sentence, two important sub-tasks can be distinguished within Semantic Interpretation: *Word Sense Disambiguation* (WSD) and *Semantic Parsing*.

Usually Word Sense Disambiguation and Semantic Parsing are considered separately but they are strongly related. WSD can improve results in Semantic Parsing (as different senses have different syntactic behaviours (specially verbs)) and vice-versa (e.g. using selectional preferences to WSD [Carroll and McCarthy, 2000]).

(PARDON) ia orthogonal to the traditional NLP task decomposition, which applies any kind of knowledge (syntactic, semantic, linguistic, statistical) at the earliest opportunity but retaining an independent representation of the different kinds of knowledge.

NLP architectures are basically determined by both Process and Knowledge integration. PARDON aims to give a general framework in which different NLP tasks can be easily formalized. So that, these different tasks can be tested separately or carried out simultaneously.

PARDON aims to explore the limits of NLP, without wrongly filtering partial solutions, or over constraining the interaction between modules/knowledge. We use Consistent Labeling Problem (CLP) as the framework to integrate different NLP process and to apply any kind of knowledge (syntactic, semantic, linguistic, statistical) at the earliest opportunity, while retaining an independent representation of the different kinds of knowledge.

PARDON aims to give a general framework, that is multilingual and open domain, in which different NLP tasks can be easily formalized. So that, these different task can be tested separately or carried out simultaneously (following an integrated approach).

Althouh, there is no doubt that syntax can help WSD, there are a few example of a real use of syntactic information in WSD systems, [**?**], [**?**]. Most WSD system relais on low level attributes (e.g. bag of words) ignoring syntax or the using syntax in a shallow manner, such as selectional preferences.

In order to demostrate the flexibility of PARDON's architeture, we applied the PARDON approach to a well definied WSD task, the SENSEVAL-II english lexical sample, using rich models with both, semantic (that is WordNet senses) and syntactic information (grammatical relations).

[1] The example is borrowed from the title of a U.S. bestseller about punctuation

The main goals of this set of experiments are, first to prove the capability of the PARDON to combine different sources of information to better solve a task (knowledge integration). Secondly we will also demostrate that combining syntax and semantic, even with noisy and poor coverage models we are able to obtain interesting results in WSD.

2 Pardon's Architecture

PARDON's architecture is based on the idea of *compositionally*. An element combines itself with other elements to build a new element. In most cases the new element shares or contains the representation of the combined elements. Elements can not be freely combined. The correct combinations of elements is determined by models and these models are associated to the initial elements.

Thus, the compositional system is formed by a set of combinatorial patterns or rules (*models*) associated to initial elements (hencefore *initial objects*). The system has to establish not only which combination of objects (derivational sequence) can be correctly performed to cover the whole sentence, but also which ones are more plausible than the others. We have restricted the combinational process to best suit the kind of models we applied in our two test tasks, but other kinds of model matching and combination criteria could be established and formalized in a different ways inside PARDON's architecture.

A frame-like semantic representation, as well as the compositional and pattern matching process of PARDON's architecture, could be formalized as a CLP. This formalization can be done in several different ways. Different formalization could lead to different performance or even to converge to different solutions when using algorithms that do not assure the global optimization. Unfortunately, there are few works on the impact of the different possible modelization in the performance [Borrett and Tsang, 1996] and besides for the empirical results it is not clear which general properties must hold a good formalization. moreover, CLP as framework allow us again to integrate the identification of a model, combination process with other constraints or preferences in a natural way

Thus, PARDON's Architecture is similar to a rule-based system and has three main components:

- **Knowledge Representation** That is how a the semantics either for partial analysis or the whole sentence are represented.
- **Model Application** That is, in which conditions and how a model is applied. In most cases, the application (or learning) of models involves the definition of a similarity function or a distance, some kind of unification process or pattern matching. These mechanisms allow to compare the models and the input. So that, several parts of the model/pattern could be identified in the input.
- **Inference Engine** That is, how and when it is decided when to apply a model.

Next subsections will explain in more detail the components which are used in the application of PARDON to WSD, tha is, Knowledge Representation and Model Application.

Variable	Values
C1.POS	{ NN1 }
C1.HEAD	{ cat }
C1.SENSE	{ cat#n#1 }

Figure 1: Variables associated with the frame-like representation of *cat*

2.1 PARDON's Knowledge Representation

A frame-like representation can be straightforwardly formalized in a CLP by representing each slot-value as a pair of variable-value. When the attribute contains a complex attribute structure of we will use a reference. Figure 1 shows the equivalent CSP representation for the frame *cat* in figure **??**).

However, most of the problems which are naturally modeled as a CLP don't have and implicit structure. We will use a kind-of dependency representation between objects, '*flattening*' our problem. The combination of objects by means of a model is represented using two variables, a variable named *model* which represents the model which is applied and another variable named *role* which represents the dependency between the two objects. There is one special model, named NONE, to represent the null-model (that is the no application of any model) and one special role, named TOP, to represent the null-role (that is that the object do not take part in any model). Figure 2 shows two different CLP representations for the "The cat eats fish", on the left using references and on the right using two special variables *model* and *role* for each object.

In order to identify a role from a model label we need a triplet *(role, object, model)*. For instance, the role NP_1 of the *MS* model for the object *eat* is represented as *(NP_1, eat, MS)*.

Since a CLP always assigns a label to all the variables; we will use the two null-labels defined previously: NONE for the model variables (objects which do not have/use a model, usually leaf semantic objects with no sub-constituents) and the label TOP for the role variables (objects not playing a role in the model of a higher constituent, e.g. the sentence head).

2.2 Model Application

In order to see whether a model can be applied or not, we should determine which combination of objects could be used to fill the model's roles (hencefore instantiate). First we will establish which roles an object can play in isolation, that is regardless which objects fulfill the other roles of the model. For instance if our models needs a number agreement between two roles we will initially oversee this constraint since it involves knowing which object is instantiating the other role.

In order to know if an object can play a role, we distinguish three different kinds of attributes:

Variable	Values
C1.POS	{ NN1 }
C1.HEAD	{ cat }
C1.SENSE	{ cat#n#1}
C2.POS	{ NN1 }
C2.HEAD	{ cat }
C2.SENSE	{ fish#n#2 }
C3.POS	{ VVZ }
C3.HEAD	{ cat }
C3.SENSE	{ eat#n#2 }
C3.MODEL	{ transitive }
C3.AGENT	{ c1 }
C3.PATIENT	{ c2 }

Variable	Values
C1.POS	{ NN1 }
C1.HEAD	{ cat }
C1.SENSE	{ cat#n#1}
C1.MODEL	{ NONE }
C1.ROLE	{ agent.transitive.c3}
C2.POS	{ VVZ }
C2.HEAD	{ eat }
C2.SENSE	{ eat#n#2 }
C2.MODEL	{ transitive }
C3.ROLE	{ TOP }
C3.POS	{ NN1 }
C3.HEAD	{ fish }
C3.SENSE	{ fish#n#2 }
C3.MODEL	{ NONE }
C3.ROLE	{ patient.transitive.c3}

Figure 2: Two different CLP formalization for the frame-like representation of *The cat eats fish*

- **Compulsory**: The object attribute must match the role attribute.
- **Optional**: If the object attribute do not match the role attribute, the matching function will penalise it. However, the object will be considerated as a possible filler of the role.
- **Ignore**: The attribute is not taken into account (e.g. a attribute containing the description of the role or the name). We do not consider these attributes at all.

We define the function $match(object, model)$ to return a measure of how well an object fills a role in isolation. That is, if an object matches all the compulsory attributes of the role without taking into account the objects that fills the other roles of the model. The final matching/similarity measure between a role and an object is calculated as:

$$match(role, object) \triangleq \frac{\sum_{a \in Atts} sim(role.a, object.a)}{\#Atts}$$

For instance, in the current example of a CFG, we will only allow an object to fill a role if both have the same category (Compulsory). Thus, we define as

$$match(object, role) = \begin{cases} 1 & if object.catg = role.catg \\ -1 & otherwise \end{cases}$$

Using this match function will result in set of possible role–objects instantiation shown in Table 1. Once the possible fillers for each role are determined, we should choose which

ones could be used together to instantiate the full model (For instance, which pairs of objects hold the number agreement, that is both objects filling the roles simultaneously have the same number). A typical CFG will force the element to be contiguous and in a determined sequential order. Thus, in the current example *The* could not fill the role *D* from the *fish*'s *MNP* model.

Role	**Object**
D.MNP.cat	{ The }
D.MNP.fish	{ The }
NP_1.MS.eat	{ cat, fish}
NP_2.MS.eat	{ cat, fish}
NP_1.MS.cat	{ cat, fish}
NP_2.MS.cat	{ cat, fish}

Table 1: Possible CLP Assignments using the Match function

However, the correspondence between the input and the models is not usually perfect. In a general framework we will need more powerful models. In different NLP tasks, the applicability conditions of the models could vary greatly, e.g. we could relax that conditions and allow the non contiguity of the elements, or even the disorder.

Moreover, the application of a model does not only need to formalize all the possible combination of objects that can instantiate a model but also to establish which one among all the possible instantiation is better. This measure mostly depends on the kind of pattern matching needed to apply the models. For instance, a particular instantiation of a model can be penalized according to different criteria, e.g. the number of gaps, the disordered fillers, the number of optimal roles that are not instantiated, etc.

Approximate pattern matching techniques based on edit operations ([Tsong-li *et al.*, 1994], [Shasha *et al.*, 1994]) are the most commonly used to deal with inexact or error-tolerant methods. For example to Semantic Parsing [**?**], [**?**]. One of the main drawbacks of the tree-edit matching approaches was the difficulty to integrate them with other types of knowledge or constraints. However, [Torsello and Hancock, 2003] prove that it is possible to approximate a tree edit distance matching using a more general framework, that is, CLP. Thus, CLP will allow us to modelize different kind of a model application (pattern matching), e.g. disorder, gaps, optional roles, and also integrate it with any other processes or knowledge we can formalize as a set of constraints.

Constraints are noted as follows: $[A = x] \sim^w [B = y]$ denotes a constraint stating a compatibility degree w when variable A has label x and variable B has label y. The compatibility degree w may be positive (stating compatibility) or negative (stating incompatibility). For simplicity we will also use the symbol $\nsim$ to denote incompatibility.

In the current example we will extend our CFG formalism to allow optional roles in a model, e.g. D^*, $N \Longrightarrow NP$ will stand for allowing an optional role *D*.

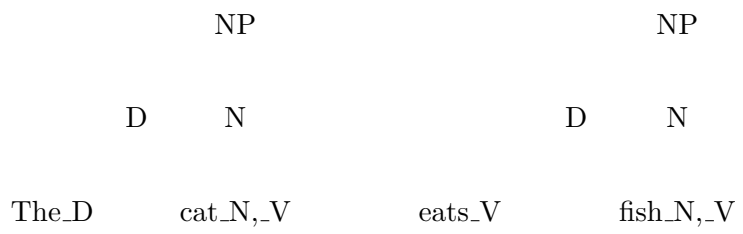

Figure 3: Instantiation of rule

3 CLP formalization

As seen in the previous subsections, PARDON's framework can be formalized as a CLP. Once a NLP task is modeled as a CLP using this framework, it can be solved using well known optimization methods (e.g. the relaxation labeling algorithm) to find the most consistent solution. A CLP with weighted constraints does not distinguish between *hard* and *soft* constraints. Some hard-constraint are implicit in the formalization and thus can not be violated (e.g. role unicity), some part of the *hard* constraints are applied during the CLP formalization to filter out values as a-kind-of arc consistency (e.g. matching constraints between the role and objects), and the remaining are relaxed to *soft* constraints giving to them an arbitrary large (infinite) weight to force the system to satisfy them on convergence. However, as we will see later on the experiments, using a relaxation labeling technique, if the final states do not hold all these constraints (because we have converged to a local maxima or there is no state which could satisfy all the constraints) a mixed partial or multiple models could be combined in the solution.

We are not concerned about the well formedness of the input. We are dealing with a communication event, where there is no doubt that, even when the utterance is not well formed and contain any kind of error there is an intended meaning. Thus, from our point of view, robustness in NLP means to find always the more reliable solution, even if the input is not well formed. So that, we allow to break hard constraints if there is no other way to find a possible solution, codifying them as soft constraints with a high weight.

On CLP the different assignments of a variable are incompatible. Thus, using this formalization the *Role Uniqueness* and *Model Uniqueness* constraints are ensured by the algorithm itself.

The next step in the formalization of PARDON as a CLP is to establish the possible assignments, that is, to determine which are the possible models, and which roles of these models can be played by the initial objects. Thus, we have to determine which of the restrictions expressed by the model must hold (*Hard Constraints*) and which constraints can be softened in order to find a solution (*Soft Constraints*), (i.e. Selectional Preferences, heuristics or knowledge that we know could be inconsistent or incomplete).

When formalizing the problem, the models that can not fill any of their compulsory roles should not be taken into account and neither should all their associated roles (and their possible assignments). The *hard-constraints* involved in the identification of roles are

applied in the function **match**. Using this function, we would determine whether an object could play a role (represented as a possible assignment) or not. The following algorithm in pseudo-code describes the general procedure for building the CLP once the initial objects are created:

```
foreach model M associated to a initial object do
        add <M,A> to the activeModels list
end foreach

foreach <M,A> in the activeModels do
        if(all the compulsory roles of M have at least one match) then
          foreach role R of model M do
                  foreach object SO do
                          if (match Role SO) then
                            add SO as possible player of role R
                          end if
                  end foreach
          end foreach
          add M as possible model for A
        end if
end foreach
```

3.1 Relaxation Labeling Algorithm

Roughly speaking, the relaxation labeling algorithm consists of:

- Start in a random labeling P_0
- For each restriction compute the influence that each assignment (variable-label) receives from the current weights. The influence measures how compatible is the assignment with the current weight according to that constraint.
- Update the weight of each variable label according to the support obtained by each of them.
- Iterate the process till a convergence criterion holds.

The three main points of the algorithm are:

- The *Support Function* which determines how to combine the different influences given by the constraints in order to calculate the support for an assignment. Usually it us computed as the addition of influences (1) but there are alternative formulas (e.g (2) or (3)).

  Falta definir la Influencia INF

$$(1)\ S_{ij} = \sum_{r} Inf(r,i,j)$$

$$(2)\ S_{ij} = \prod_{G \in P(V)} \sum_{r \in G} Inf(r,i,j)$$

$$(3)\ S_{ij} = \prod_{G \in P(V)} \max_{r \in G} Inf(r,i,j)$$

- The *Updating Function* which determines how the weight for a variable-label assignment is updated according to the supports given. The more common general formulas are (1) and (2).

$$(1)\ P_j^i(n+1) = \frac{P_j^i(n) \times (1 + S_{ij})}{\sum_{k=1}^{m_i} p_k^i(n) \times (1 + S_{ik})}$$

$$(2)\ P_j^i(n+1) = \frac{P_j^i(n) \times S_{ij}}{\sum_{k=1}^{m_i} p_k^i(n) \times S_{ik}}$$

- The *Convergence criteria* which determines when the algorithm has reach an acceptable solution (e.g a fixed number of iteration, measures of the weight variation, etc.). Relaxation labeling is proved to converge under certain criteria [**?**]. This conditions require simple models -e.g. to use only symmetric binary constraints- which is not likely to hold on a real NLP application.

3.2 Determining the Initial State

In order to initialise the weight of each assignment in the CLP (initial state), the weights can be equally distributed between all the possible values of a variable or particular heuristics can be set according to the particular task.

3.3 Specific Constraints

One of the main advantages of CLP is that it can be easily added arbitrary sets of task-specific constraints (e.g. statistical information, selectional preferences) which enforces the applications of the models or assures other preferences or desirable characteristics of the solution (e.g. non crossing of syntactic dependencies).

4 Source Data

Our aim is to demostrate the PARDON architecture's robustness and flexibility more that to obtain a *better than yours WSD* system. Thus, although we are fully aware that this desition will have a big impact on the performance of the system, we decide to automatically obtain models by parsing the most widely used sense tagged corpus, SemCor [Miller *et al.*,

1993]. Once we have build models with WordNet sense information, we can enrich those models using all the information inside the MCR [**?**].

For instance, starting from the dependency analysis shown in figure 4 for the sentence "*The cat eats fish*", we can enrich the representation using the MCR. Adding for each word in the sentence, all the semantic information available in the MCR for its different sense. Figure 5 shows the enriched object resulting for the noun *fish*, related to its two senses, the food sense (fish#n#1) and the animal sense (fish#n#2).

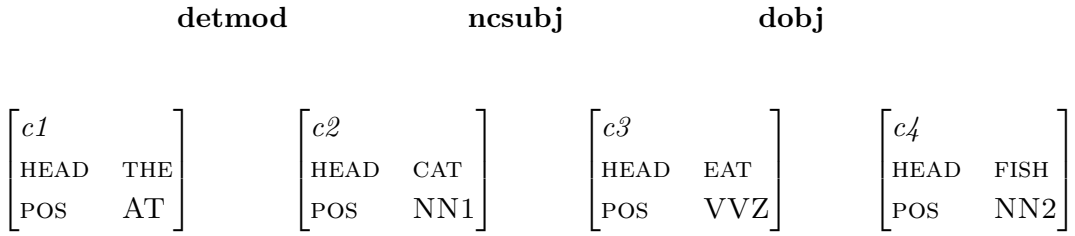

Figure 4: Dependencies for "*The cat eats fish*"

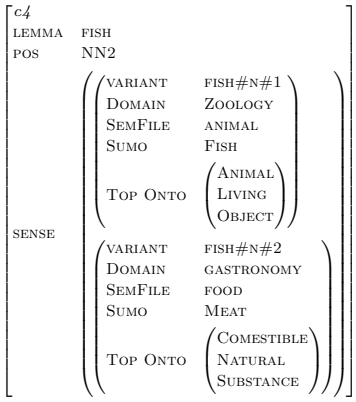

Figure 5: Object *Fish* enriched with MCR information

The criteria for defining MWEs and even for lemmatizing could vary greatly between a genaral tool like RASP and WordNet. Thus, bearing in mind that our final goal is the WSD of free running text againts WordNet, we decide to preprocess the corpus before applying RASP to make the output more compatible to WordNet. That is no only to lemmatize according to the WordNet criteria but also to recognizing the MWEs in WordNet. MWE recognition is not only relevant to WSD (as they tend to be less ambiguous) but also to PoS Tagging and Parsing as many of them has an idisincratic syntactic structure. In WordNet, MWEs do not have any inflectional information, although we allow some inflection of their sub parts according to a set of patterns ([Arranz *et al.*, 2005]).

SemCor [Miller *et al.*, 1993] is a subset (about 250,000 words) of the Brown Corpus, consisting of texts that have been tagged with POS information and WordNet senses. Sem-

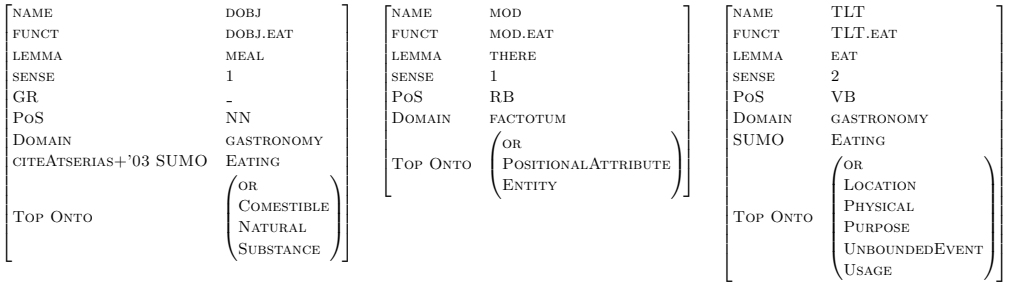

Figure 6: Model 47730 for eat_2 from file brown1/br-k09.xml p4 s9

cor consists of about 186 documents classified into 20 classes. This corpus was semantically annotated with WordNet 1.6 senses, and actually, automatically mapped to WordNet 1.7, WordNet 1.7.1 and WordNet 2.0.

The Semcor corpus is sense tagged, that means that Name Entities are tagged as person or group and MWE are identified. We should keep the original tokenization in order to be able to recover the semantic information (synsets) after the parsing.

We use RASP [Carroll *et al.*, 1998] to extract grammatical relations from Semcor. The PoS used in SemCor was automatically tagged and the tag set used is not the same that needs the RASP grammar (CLAWN5). Thus, we decide to process the Semcor corpus from scratch (row text). However, this also brings up some difficulties, the Lemmatization (e.g. holy-of-holy vs holy-of-holies) and both, the identification criteria and the set of Multiword Expressions could differ greatly between Semcor (that is WordNet) and the RASP. Thus, although we process the corpus from scratch, we keep the original tokenization[2], lematization and PoS in order to, later, look up the information in WordNet.

Once the sentence has been parsed, the set of dependecies extracted is used to build the models in a straighforward manner. For each syntctic head, we build a model containg the set of direct dependencies which arrived to it. No generalization process is carried out.

Figure 6 shows the model obtained for *eat_1* from the Semcor sentence "*I have observed that being up on a horse changes the whole character of a man, and when a very small man is up on a saddle, he'd like as not prefer to eat his meals there.*".

5 Evaluation

It is difficult to compare the evaluation of diferent WSD systems if they use diferent sense repositories or test corpus. In order to adress this problem the it was dessigned the SENSEVAL WSD disambiguation framework [**?**]. SENSEVAL[3] is the international organization devoted to the evaluation of Word Sense Disambiguation Systems. Its mission is to organ-

[2]Contractions such as *n't* splitted from the verb, which must be converted to *not* so the parser could interpret them correctly

[3]http://www.senseval.org

ise and run evaluation and related activities to test the strengths and weaknesses of WSD systems in different tasks. From SENSEVAL-2 to SENSEVAL-3 there were not a significant improving in performance. The best systems are still about 65-70%. New approaches to WSD are needed.

Two different and complementary dimensions can help to the WSD problem according to [Rigau *et al.*, 2002]: multilingualism and domains. Although, working in parallel with comparable corpora in several languages will increase the complexity of the process, we believe that language translation discrepancies among word forms can help the selection of the correct word senses [Habash and Dorr, 2002]. Moreover, further reduction of the search space among sense candidates can be obtained by processing domain corpora [Gale *et al.*, 1992]. One possible shortcut to this never ending cycle by means of consecutive ACQ and WSD rounds.That is, approaching the solution to one facet will help the rest. In order to evaluate the viability of our approach, we decide to test it agains the Senseval-II English Lexical Sample.

6 Results

Table 2 shows the figures obtained for the Senseal-II English lexical sample according to the oficial scorer.

	Precision	**Recall**	**Coverage**
Fine	0.32 (691.55 / 2120)	0.16 (691.55 / 4328)	48.9 (1049.34)
Coarce	0.49	0.242	48.9

Table 2: Senseval-II English Lexical Sample Figures

The system is unable to decide a sense for 1979 sentence out of 4328. Altough the general figures are quite far from the state of the art WSD systems, we have proved the viability of this novel approach.

The current formalization, include ver restricted model and purely based on the direct syntax/semantic information of the sentence. Thus, PARDON can not dissambiguate sense from which the system do not have an example. As the current version of PARDON do not include modules to add information from the gloss, the semantic relations, domains, etc.

Similarly, PARDON can not currently overcome some errors or inconsistenes either in the knowledge base or in the tagged examples. For instance the most similar sentence to "the cat eats fish" in Semcor is "... eats eggs". The problem is that in wordnet 1.6[4] "eggs" do not have a sense as food just as animal. Thus, the word fish is wrongly disambiguated. There are several methods ([**?**], [Tomuro, 2000]) to detect this kind of regular polysemy (e.g. *animal-—food*) in WordNet, and they can be applied to both, the knowledge and the process to detect and specially treat this kind of polisemy.

[4]In wordnet 2.0 a food sense was added

PARDON can not dissambiguate words whose syntactic behavior do not vary and whose senses are closer according all the semantic information of MCR (that is, the WordNet hierarchy hirearchy and all the different semantic resources, SUMO, Top Concept Ontology, Lexicographer Files, Domains). For instance Figure **??** shows the six differences of the six different sense of child. It can be seen that they almost share all the semantic attibutes from MCR (that is, SUMO, Domains, etc) and its position in the WordNet hierarchy (all senses as descendant of *person#n#1* does not help much to disambiguate them. In most cases, discourse, extra-sentencial knowledge or statistical knowledge (e.g. frequency, co-occurences) will be needed to disambiguate among these senses.

7 Conclusions

We have applied a novel architecture (PARDON) to WSD, orthogonal to the traditional NLP task decomposition, which applies any kind of knowledge (syntactic, semantic, linguistic, statistical) at the earliest opportunity but retaining an independent representation of the different kinds of knowledge. PARDON aims to give a general framework in which different NLP tasks can be easily formalized. So that, these different task can be tested separately or carried out simultaneously (following an integrated approach).

PARDON's architecture give us a natural way to integrate different knowledge, as a set of constraints inside a CLP, to solve diferent NLP tasks. The only condition requiered is that each different source of knowledge can be related (as it is inside the MCR through the ILI record). Although the integration effort inside the MCR, these different knowldge/views could became incompatible or contradictory, CLP will also give us a natural way to integrate them. Then, NLP tasks will be faced as an optimization problem, transforming the appropriate pieces of knowledge in a set of constraints and trying to find a solution that satisfies them -to a maximum possible degree-.

Semcor and Senseval sentences are complex, in our formalization syntax biasses too much our pattern application, the system is also unable to deal with diathesis or to syntactic structures that are not present on the models. Using only the models obtained form SemCor, the results are hard to compare with other WSD systems, as our coverage of the models per sense is poor, and this has a great impact on the performance, as it cna not be know in advance for which sentences we do not have models of the correct sense. On the other hand, enriching the current model with other WSD heuristics, (such as Domain base WSD) could increase greatly our coverage.

The models obtaining from semcor do not allow to build the best semantic representation, for instance in the sentence *.. clean dental surface ...*, the semantic object for *dental surface* will be associated basically to *surface* which is not directly related to *body_part*. This will end up in the wrong dissabiguation of the verb *clean*, as cleaning a surface is more related to clean a house than to clean a body_part.

Although the limitations of the current experiments, the PARDON Architectures opens a new research line in WSD based on information integration and the integration of NLP processes.

However, neither increasing the coverage of the models, nor to improve the WSD rates are out of the scope of the current work. The results obtained are good enough to demostrate that the PARDON architecture is applicable to WSD.

8 Future Work

References

[Arranz *et al.*, 2005] Victoria Arranz, Jordi Atserias, and Mauro Castillo. Multiword expressions and word sense disambiguation. In *CICLING'2005*, 2005.

[Borrett and Tsang, 1996] James E. Borrett and Edward P.K. Tsang. Towards a formal framework for comparing constraint satistaction problems formulations. Technical Report CSM-264, Dept. of computer Science, University of Essex, 1996.

[Brill and Mooney, 1997] Eric Brill and Raymond J. Mooney. An Overview of Empirical Natural Language Processing. *Artificial Intelligence Magazine*, 18(14):13–24, Winter 1997. Special Issue on Empirical Natural Language Processing.

[Carroll and McCarthy, 2000] J. Carroll and D. McCarthy. Word sense disambiguation using automatically acquired verbal preferences. *Computers and the Humanities. Senseval Special Issue*, 2000. available http://www.cogs.susx.ac.uk/lab/nlp/mccarthy/mccarthy.html.

[Carroll *et al.*, 1998] J. Carroll, G. Minnen, and E. Briscoe. Can subcategorisation probabilities help a statistical parser? In *Proceedings of the Sixth ACL/SIGDAT Workshop on Very Large Corpora*, pages 118–126, 1998.

[Gale *et al.*, 1992] W. Gale, K. Church, and D. Yarowsky. One sense per discourse. In *Proceedings of of DARPA speech and Natural Language Workshop*, Harriman, NY, 1992.

[Habash and Dorr, 2002] N. Habash and B. Dorr. Handling translation divergences: Combining statistical and symbolic techniques in generation-heavy machine translation. In *Proceedings of the Fifth Conference of the Association for Machine Translation in the Americas, AMTA-2002*, Tiburon, CA, 2002.

[Miller *et al.*, 1993] G. Miller, C. Leacock, R. Tengi, and R. Bunker. A Semantic Concordance. In *Proceedings of the ARPA Workshop on Human Language Technology*, 1993.

[Rigau *et al.*, 2002] G. Rigau, B. Magnini, E. Agirre, P. Vossen, and J. Carroll. Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of COLLING Workshop*, Taipei, Taiwan, 2002.

[Shasha *et al.*, 1994] D. Shasha, J. Tson-Li Wang, K. Zhang, and Y. Shih. Exact and approximate algorithms for unorder tree matching. *IEEE transactions on System Man and Cybernetics*, 28(5):668–678, April 1994.

[Tomuro, 2000] Noriko Tomuro. Automatic extraction of systematic polysemy using tree-cut. In *ANLP workshop*, 2000.

[Torsello and Hancock, 2003] Andrea Torsello and Edwin Hancock. Computing approximate tree edit distance using relaxation labeling. *Pattern Recognition Letters*, (24):1089–1097, 2003. Elsevier.

[Tsong-li *et al.*, 1994] J. Tsong-li, K. Zhang, K. Jeong, and D. Shasha. A system for approximate tree matching. *IEEE transactions on Knowledge and Data Engineering*, 6(4), 1994.