Corpus-based Learning of Semantic Relations by the ILP System, Asium

Claire Nedellec

Inference and Learning group
Laboratoire de Recherche en Informatique
UMR 8623 CNRS, bât 490
Université Paris-Sud, F-91405 Orsay
e-mail: cn@lri.fr

Abstract. This chapter presents the ILP method, Asium, that learns ontologies and verb subcategorization frames from parsed corpora in specific domains. The ontology is learned by bottom-up conceptual clustering from parsed corpora. The clustering method also generalizes the grammatical relations between verbs and complement heads as they are observed in the corpora. The set of grammatical relations learned for a given verb forms the verb subcategorization frame. This paper details Asium's learning algorithms in an ILP formalism and shows how the learned linguistic resources can be applied to semantic tagging, language control and syntactic disambiguation within the ILP framework.

1. Introduction

Many applications that handle electronic documents, especially industry-oriented applications, require specific knowledge bases and linguistic resources. Among them, predicate schemata are syntactico-semantic resources, which let the system interpret chunks of documents at the knowledge level. By predicate schemata we mean conceptual graphs which relate predicates to their semantic roles defined as semantic sets of terms such as

```
Pouring[1]    object   liquid, powder
              place    recipient, preparation
```

where `liquid = {milk, wine, ...}`. Such resources are useful in many applications such as Information Extraction (Riloff, 93), Information Retrieval, and Question Answering. So far, there is no fully automatic method for learning this kind of predicate schemata from parsed training corpora. However, verb subcategorization frames and semantic classes can already be learned and this is a step in the right

[1]The examples illustrated in this paper have all been taken from the application of Asium to the cooking recipes domain.

direction towards predicate schemata learning. In addition, they are useful for the same applications as predicate schemata. For instance, the subcategorization frame of « to pour » could be defined as

```
To pour  direct object                    liquid, powder
         adjunct (preposition = in, into)   recipient,
preparation
```

Compared to predicate schemata, the predicate here is a verb or a noun and the relations are syntactic. Semantic classes are defined in a similar way as in predicate schemata.

Previous work attempted to learn verb syntactic relations and semantic classes by observing syntactic regularities in parsed corpora. The work reported in (Grishman and Sterling, 94), (Hindle, 90), (Grefenstette, 92), (Pereira et al., 93), (Dagan et al., 96) among others, is based on Harris' hypothesis of the distributional foundation of meaning in sublanguages (Harris et al., 89): syntactic schemata that consists of semantic word classes that reflect the knowledge domain can be identified by analyzing syntactic regularities in specific domain parsed corpora. Most of them learn flat classes (Grefenstette, 92), (Hindle, 90) or distances between terms which represent their semantic similarities (Grishman & Sterling, 94), (Sekine et al., 92), (Dagan et al., 96), (Resnik, 95).

Hierarchies of concepts attached to verb frames (Pereira et al., 93), (Hogenhout & Matsumoto, 97) are more understandable than flat classes and they allow semantic interpretations of document chunks at the appropriate level of abstraction. This is particularly useful in query extension and question answering as shown by WordNet exploitation (Resnik & Hearst, 94), (Yarowsky, 92) but unnecessary for syntactic disambiguation which is the goal of most of the work cited above.

Here, we present Asium, a method that learns verb subcategorization frames and concepts from parsed texts in a specific domain. It learns graphs of concepts instead of concept hierarchies so that a given concept may be related to more than a "mother". This property is required for representing the different meanings and roles of terms. Asium is based on a bottom-up clustering method as in previous work by Pereira et al (93) and Hogenhout and Matsumoto (97). The clustering method forms semantic classes of head terms, which represent the concepts of the ontology. It applies a novel distance that proves to be robust when applied to noisy corpora. The grammatical relations observed in the corpora between verbs and terms are also generalized so that they link verbs and all the classes of acceptable terms, not only the ones actually occurring in the corpora. The set of learned grammatical relations for a given verb forms its subcategorization frame where the selection restrictions are filled by the concepts of the ontology.

Asium is formalized as an ILP method as opposed to previous work described in the statistics framework. The ILP formalism increases the comprehensibility of the learned knowledge and makes the comparison and the integration with other ILP methods easier. The nature of Asium input and output is intrinsically relational: it consists of relations between verbs and their complements and generality relations between concepts. Moreover, Asium has to be coupled with tools that can handle relational data and knowledge. For instance, parsing disambiguation and semantic tagging will apply coverage testing operations such as filtering, saturation and

resolution. In the current implementation of the platform, data and knowledge are all stored in a relational database. In addition, learning predicate schemata from verb subcategorization frames and ontologies requires a relational representation for expressing the dependencies among concepts of the semantic roles. The ILP framework is thus viewed as a unified framework for integrating these tools and methods.

The remainder of this chapter is organized as follows. Section 2 introduces the settings Asium uses, while Section 3 details the learning algorithms. Section 4 then presents some potential applications of the learning results. Future work is discussed in Section 5 and finally, Section 6 compares the approach with other related work.

2. Settings

2.1 Subcategorization Frames

Subcategorization frames describe the grammatical relations between verbs and their arguments and adjuncts. The verb adjuncts are the verb complements such as the *place*, *time*, and *means* adjuncts that are optional as opposed to verb arguments (subject, direct object and indirect objects). A grammatical relation represents the type of complement and preposition needed if any. The verb sucategorization frames include selection restrictions, which define the concepts that are valid for a given grammatical relation. For example, Table 1 represents the subcategorization frame for the verb "to pour" as Asium learns it. Each clause represents a single grammatical relation between "to pour", and one argument or one adjunct. X represents the grammatical relation and Y represents the complement.

Table 1. The subcategorization frame for the verb "to pour".

```
c(pour,X) :-
  verb(pour,X),comp(dobj,Y,X),prep(none,Y),head(liquid,Y).
c(pour,X) :-
  verb(pour,X),comp(dobj,Y,X),prep(none,Y),head(preparation,Y)
 .
c(pour,X) :-
  verb(pour,X),comp(padj,Y,X),prep(into,Y),head(liquid,Y).
c(pour,X) :-
  verb(pour,X),comp(padj,Y,X),prep(into,Y),head(recipient,Y).
```

The two first Horn clauses say that "to pour" allows two concepts "liquid" and "preparation" as selection restrictions for the direct object (denoted "dobj"). The last two clauses show that "to pour" also allows the two concepts "liquid" and "recipient" as selection restrictions for the place adjunct (denoted "padj") and they are both introduced by the preposition "into".

2.2 Ontology

The concepts filling the selection restrictions of the verb frames are defined in the ontology. The ontology concepts learned by the method are defined both extensionally and intensionally. The extensional definition defines a concept as a class of terms. Term should be understood in the linguistic sense. For instance "baking powder" is a term. The membership relation of a term T to a concept C is represented by a Horn clause, `C(X) :- T(X).` For instance, the clauses

```
liquid(X) :- milk(X).
liquid(X) :- water(X).
liquid(X) :- red_wine(X).
```

define the concept "liquid" as the set of terms "{milk, water, red_wine}".

The intensional concept definition defines concepts as being related to other concepts by a generality relation. The generality relation on the ontology concepts is defined by the inclusion between term classes. A concept `C` is more general than a concept `C'` if the class of terms of `C'` is included in the class of terms of `C`. The generality relation between a concept `C` and a concept `C'` is represented by the Horn clause `C(X) :- C'(X)`, as exemplified by the clauses below.

```
liquid(X) :- milky_liquid(X).
milky_liquid(X) :- milk(X).
milky_liquid(X) :- cream(X).
liquid(X) :- alcoholic_beverage(X).
```

Concepts and terms are not explicitly distinguished in this representation; both are represented by unary predicates. The corpus terms are the predicates that do not occur as a head of Horn clauses in the ontology, they are terminal nodes. The structure of the ontology is a directed acyclic graph. The only relation represented in the ontology is the generality relation among the concepts.

2.3 Applications

The applications of this kind of linguistic resource are numerous. Among others, they are useful for disambiguating syntactic parsing. For instance, the noun phrase "3 minutes", in the clause "cook until golden, 3 minutes per side", could be interpreted by a syntactic parser in two ways: as the direct object or as a time adjunct. No syntactic hint can help the parser here; additional semantic knowledge is needed. The appropriate ontology allows the parser to identify "3 minutes" as belonging to the "duration" concept. With the subcategorization frame of "to cook", the parser recognizes that the duration concept plays the role of time adjunct without a preposition, and the result is that the interpretation of "3 minutes" as time adjunct is selected and its interpretation as direct object is removed. Section 4 gives a more complete description of verb subcategorization frame use and it also presents how learned linguistic resources can also be used for semantic tagging and language control.

2.4 Learning Input and Output

The concepts of the ontology and the generality relation among them are learned from a parsed training corpus. The argument and adjunct heads of the verbs in the training corpus form Asium's input. The head of a complement is its main term. For example, in "pour the *hot milk* into the prepared pan. ", "milk" is the head of the noun phrase "*hot milk*". In the following, *head* should be understood in the linguistic sense and not the logical one. The output of the algorithm is a set of subcategorizations frames and ontologies as defined above.

3. Learning

Ontology and verb subcategorization frame learning consists of three phases: corpus pre-processing, an initial generalization step, and further generalization steps. Corpus pre-processing transforms the training corpus into a set of examples. The initial generalization step builds the leaves of the ontology while further generalization steps build the higher levels and learn the valid grammatical relations between verbs and concepts.

3.1 Corpus Pre-processing

First of all, a syntactic parser analyzes the training corpus. In the current implementation, SYLEX (Ingenia corp.) is used (Constant, 95). In each clause, the parser identifies the verb, its arguments and its adjuncts. It identifies the head and the preposition, if any, of each argument or adjunct (Section 3.1.1). Properly identifying a head term as a single word or a complex phrase depends on whether or not the terminology dictionary of the parser is suitable for the domain. Only grammatical relations between verbs and heads plus prepositions are relevant as input to Asium. Asium extracts all grammatical relations independently of each other (Section 3.1.2).

Concept construction by clustering classes of terms is based on a predicate invention operator. Thus, we need a pre-processing flattening step that will turn terms into predicates (Section 3.1.3).

3.1.1 Parsing Output

Parsed sentences (clauses) are represented by Horn clauses in the following form,

```
verb(V_label,Clause_id) :- [
  comp(Gram_rel,Phrase_id,Clause_id),
  prep(Prep_label,Phrase_id),
  head(Head_term,Phrase_id)]*.
```

A verb can have several complements, each of which has a preposition and a head. Each triple represents a complement of the verb.

- The `verb` literal represents the verb, the `comp` literal, a complement of the verb, argument or adjunct, the `prep` literal, the preposition of the given complement, and the `head` literal, the head of the given complement.
- `V_label`, `Prep_label` and `Head_term` respectively represent the labels of the verb, of the preposition and of the head term. In case where the argument of the verb is not introduced by a preposition, `Prep_label` is equal to `none`.
- `Clause_id` and `Phrase_id` represent the unique identifiers of the clause and the phrase in the training corpus.
- `Gram_rel` represents the grammatical relation in the clause between the arguments and the verb, as subject (subj), object (dobj), indirect object (iobj), and between the adjuncts and the verb as time adjunct (tadj), place adjunct (padj), etc. When the parser is not able to specify the type of adjunct, the more general label "adj" is given.

Example. The Horn clause `E` represents the parsing of the clause, "pour the hot milk into the prepared pan".

```
E:  verb(pour,c1) :-
      comp(dobj,p11,c1), prep(none,p11), head(milk,p11),
      comp(padj,p12,c1), prep(into,p12), head(pan,p12). □
```

Subjects are generally absent from cooking recipes. However, the method treats subjects as the other verb arguments. When the syntactic analysis is insufficient for disambiguating multiple interpretations, all interpretations are kept as input.

3.1.2 Extraction of Verb Grammatical Relations

The learning method generalizes the observed grammatical relations independently from each other. Thus single grammatical relations are extracted from parsed examples: extraction splits the input Horn clauses into the same number of new Horn clauses as there are grammatical relations between the verb and the complements occurring in the input Horn clause.

Input Horn clauses are connected with a maximal depth of 3. Connection paths form a tree and this tree's root is `Clause_id` in the head literal. Extraction copies the head of the input clause as the input of each new clause and follows each possible connection path from the head clause variables through the variables of the body literals until it finds the deepest variables. It amounts to partitioning the variable set into subsets of k-local variables, where k = 4 (Cohen, 93).

Example. The clause `E` from the above example is split into the two Horn clauses E_1 and E_2, with one clause per grammatical relation in `E`.

```
E1: verb(pour,c1) :-
      comp(dobj,p11,c1), prep(none, p11), head(milk, p11).
```

E_2`: verb(pour,`c_1`) :-`
`      comp(tadj,p12,c1), prep(into, p12), head(pan, p12).` □

Candidate hypotheses are represented in the same way. Thus hypothesis generation and coverage tests are simplified during learning. In particular, this allows example storage in relational databases for faster coverage test. However, this results in

overgeneralization since the grammatical relations and selection restrictions for a given verb are generalized independently of each other (Section 5).

3.1.3 Head Term Flattening

Flattening then turns the head term in each Horn clause into a predicate so that predicate invention by intraconstruction can then apply. The other terms remain unchanged.

```
verb(V_label,Clause_id) :-
  comp(Gram_rel,Phrase_id,Clause_id),prep(prep_label,Phrase_id
),
  head(Head_term,Phrase_id).
```

becomes

```
verb(V_label,Clause_id) :-
  comp(Gram_rel,Phrase_id,Clause_id),prep(Prep_label,Phrase_id
),
  head(Head_id,Phrase_id), head_term(Head_id).
```

Example. F_1 results from flattening E_1.

F_1: verb(pour,c_1) :-
comp(dobj,p_{11},c_1),prep(none,p_{11}),head(h_{11},p_{11}),milk(h_{11})
. □

The flattened clauses form the input examples of the generalization algorithm. Flattening in Asium slightly differs from flattening as defined by (Rouveirol, 94) in that it applies to constant head terms instead of all constants. The other constants c_1, p_{11}, h_{11} are turned into variables by the initial generalization step.

3.2 Initial Generalization Step

The initial generalization step consists of first variabilizing the input examples (Section 3.2.1). Next, it creates the basic concepts of the ontology by predicate invention and compresses the input examples. The new predicates represent classes of head terms. This way, a given grammatical relation between a given verb and all heads occurring in the corpus is represented by a single Horn clause where a new predicate represents all possible heads in the training corpus (Section 3.2.2).

3.2.1 Variabilization

The identifiers in the input examples are variabilized, while the grammatical role, the verb and the preposition labels remain unchanged. The resulting clauses are called *variabilized examples* and define the set V.

Example.

V_1: verb(pour,*X*) :-
comp(dobj,*Y*,*X*), prep(none,X), head(*Z*,*Y*), milk(*Z*).

V_1 results from variabilizing F_1. □

One may notice that the number of input examples that are covered by a given variabilized example is the number of occurrences of the head terms occurring with the verb and in the grammatical role defined in the variabilized example. For instance clause V_1 covers as many examples as the number of clauses in the corpus where "milk" should be "poured". The number of examples covered is attached to each variabilized example V_i and denoted $Occ(V_i)$ as *number of occurrences*.

3.2.2 Predicate Invention

In Asium, predicate invention is done by applying Muggleton and Buntine's (88) intraconstruction operator to variabilized examples. It creates one new predicate for each set of head terms that occur with the same verb and the same grammatical role and preposition if any. Let us first present the intraconstruction operator.

- **Intraconstruction**

As defined in (Muggleton & Buntine, 88), it applies to two clauses, the bodies of which contain a same subpart. Let R_1 and R_2 be these clauses, the bodies of which contain a same subpart B_2,

R_1: C(X) :- $B_{11} \wedge B_2$. R_2: C(X) :- $B_{12} \wedge B_2$.

Intraconstruction creates 2 new clauses,

C_1: np(?) :- B_{11}. and C_2: np(?) :- B_{12}.

which define the new predicate np(?) as the disjunction of $B_{11}(X)$ and $B_{12}(X)$, the subparts of R_1 and R_2 that differ. Inverting the resolution with the parent clause C_1 and the resolvent R_1 yields the parent clause G:

G: C(X) :- np(X_i) $\wedge$ B_2.

as well as inverting resolution with the clauses C_2 and R_2 (Figure 1).

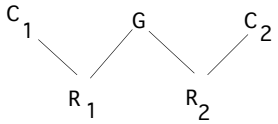

Fig. 1. Intraconstruction

- **Predicate Invention in Asium**

In Asium, predicate invention applies to the set of variabilized examples of the same verb, the same grammatical role and the same preposition, if any, but where the head terms may differ (in italics below).

```
verb(V_label,X) :-
  comp(dobj,Y,X), prep(none,X), head(Z,Y), head_term(Z).
```

Example. The three examples below concern the same verb "to pour", the same grammatical relation direct object, but different heads "milk", "water" and "wine".

```
V1:                                                    verb(pour,X) :-
     comp(dobj,Y,X),prep(none,X),head(Z,Y),milk(Z).
V2: verb(pour,X) :-       comp(dobj,Y,X),prep(none,X),head(Z,Y),
water(Z).
V3:                                                    verb(pour,X) :-
     comp(dobj,Y,X),prep(none,X),head(Z,Y),wine(Z).
```

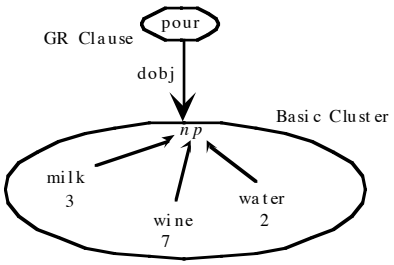

Fig. 2. Predicate invention example. □

Let us call such sets of clauses (one set per verb plus grammatical relation) *basic clusters*. These clusters form *a partition* of the set of variabilized clauses. The predicate invention operator creates one new predicate per cluster. The new predicate is defined by the disjunction of all heads in the variabilized clauses. The clauses that define the new predicate form the so-called *basic clauses* of the Domain theory (DT).

The clause obtained by intraconstruction

```
verb(V_label,X) :-
  comp(Gram_rel,Y,X), prep(Prep_label,X), head(Z,Y), np(Z).
```

forms a generalization of the clauses of the corresponding basic cluster, with respect to DT according to generalized subsumption (Buntine, 88). Such clauses are called basic Grammatical Relation clauses, or basic *GR* clauses.

Example. The basic clauses

```
np(X) :- milk(X).   np(X) :- water(X).   np(X) :- wine(X).
```

are built from the clauses V_1, V_2 and V_3. G is the basic GR clause obtained by intraconstruction.

```
G: verb(pour,X) :-
      comp(dobj,Y,X), prep(none,X), head(Z,Y), np(Z). □
```

New predicates formed by Asium are named by the user as they are built (see (Faure & Nedellec, 99) for more details on user interaction).

At this stage, the set of GR clauses together with DT, comprehensively represents all the grammatical relations between verbs and terms that occur in the input corpus. As such, they represent pieces of subcategorization frames. Thus, in the example, *np* represents the set "{milk, water, wine}" of head terms observed in the corpus for the relation "pour - Direct Object" (Figure 2). Further steps will generalize these relations, extend DT and the variabilized example set.

3.3 Further Generalization Steps

All further generalization steps iterate the same way. The algorithm is given in Table 2. It outlines the generalization process detailed below. For input, each step uses the results of the previous steps (i.e., the Domain Theory DT, the GR clause set and the variabilized example set) and extends them.

Table 2. Asium generalization algorithm.

Initialization
GR ← {Basic GR clauses}
DT ← {Basic clauses}
V ← {Variabilized examples}
NewGR ← ∅

Loop

For all (G_i, G_j) ∈ GR x GR, and $G_i \neq G_j$
Compute Dist(G_i, G_j)
If $Dist(G_i, G_j)$ < Threshold, then

• *Generalization:*
Build a new predicate NewP, form NewP definition, and G_i' and G_j' generalizing G_i and G_j by intraconstruction
NewGR ← $\{G_i'\} \cup \{G_j'\} \cup$ NewGR
DT ← NewP definition ∪ DT

• *Example generation:*
Generate new examples sets VG_i' and VG_j' by partial evaluation of G_i' and G_j'.
V = $VG_i' \cup VG_j' \cup$ V

Endif
Endfor
exit when NewGR = ∅
GR ← GR ∪ NewGR
NewGR ← ∅
End loop

A generalization step consists of selecting the most *similar* pairs among current GR clauses according to a given distance and generalizing each pair into two new clauses to be added to the current set of GR clauses. The similarity between clauses is based on the number of similar variabilized examples covered. Two examples are *similar* if they have the same head term (Section 3.3.2). The effect of such a generalization step on a given clause G_i, similar to a clause G_j, is that the new clause G_i' covers the grammatical relation of G_i, associating the verb of G_i and the terms of G_i *plus* the terms of the *other* clause G_j. In other words, it extends the set of acceptable terms for a given verb-grammatical relation by adding to it the terms of another similar set of

terms for another verb - grammatical relation. The two extended sets of terms are equal and define a same new predicate (Section 3.3.1). New variabilized examples are then generated so that they can be associated to the number of occurrences required for computing distances in further generalization steps (3.3.3).

3.3.1 Generalization

All pairs of GR clauses which are separated by a distance less than a given threshold, are generalized by applying the APT predicate invention operator defined in (Nedellec, 92). As with Muggleton's and Buntine's (88), it is based on intraconstruction but results in generalization instead of compression. It operates in the following way.

Given a pair GR_i and GR_j of GR clauses, a new predicate is created, which is substituted for the head term predicates in the clause pairs. This yields two new clauses, GR_i' and GR_j'. The new predicate is defined in DT as the "mother" of the two head terms of the clause pair. The initial two clauses cannot concern simultaneously the same verb *and* the same grammatical relation and their head term differ. It is because of the way the initial GR set of clauses has been built. The new clauses GR_i' and GR_j' are thus respectively *more general* than the GR clauses GR_i and GR_j they are built from (Figure 3).

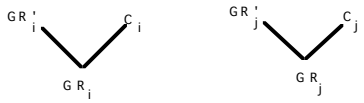

Fig. 3. APT's predicate invention

More formally,

Gr_i: verb(V_label_i,X) :-
comp($Gram\_rel_i$,Y,X),prep
($Prep\_label_i$,X),head(Z,Y),np_i(Z).

GR_j: verb(V_label_j,X) :-
comp($Gram\_rel_j$,Y,X),prep($Prep\_label_j$,X),head(Z,Y),np_j(Z).

a new predicate np is defined in DT as,

C_i: np(X) :- np_i(X). C_j: np(X) :- np_j(X).

Two new clauses GR_i' and GR_j' are built by intraconstruction, respectively substituting np for np_i and np_j in GR_i and GR_j by inverting two resolution steps.

GR_i': verb(V_label_i,X) :-
comp($Gram\_rel_i$,Y,X),prep($Prep\_label_i$,X),head(Z,Y),*np*(Z).

GR_j': verb(V_label_j,X) :-
comp($Gram\_rel_j$,Y,X),prep($Prep\_label_j$,X),head(Z,Y),*np*(Z).

The APT predicate invention operator applied here differs from the one described in Section 3.2 as it performs generalization instead of compression as Muggleton's and Buntine's (88) operator does. The parts of the two initial clause bodies which are not replaced by the new predicate differ, and thus two parent clauses are built instead of one and all that they have in common is the invented head term. They are thus more general than the ones they have been built from.

The new clauses GR_i' and GR_j' are added to the GR set of clauses, however GR_i and GR_j are not removed so that they can produce other generalizations than GR_i' and GR_j'. DT then forms an acyclic graph and not a hierarchy.

Example. Suppose G_1 and G_2 , given below, considered as similar.

```
G1: verb(pour,X):-                              comp(dobj,Y,X),prep
(none,Y),head(Z,Y),np1(Z).
G2: verb(drop,X):-comp(padj,Y,X),prep(in,Y),head(Z,Y),np2(Z).
```

np_1 and np_2 are defined in DT by

```
np1(X) :- milk(X).   np1(X) :- water(X).     np1(X) :- wine(X).
np2(X) :- milk(X).   np2(X) :- water(X).     np2(X)          :-
cream(X).
```

A new predicate np is invented and defined as

```
np(X) :- np1(X).     np(X) :- np2(X).
```

and two new clauses G_1' and G_2' are built which generalize G_1 and G_2:

```
G1': verb(pour,X):-
comp(dobj,Y,X),prep(none,Y),head(Z,Y),np(Z).
G2': verb(drop,X):-comp(padj,Y,X),prep(in,Y),head(Z,Y),np(Z).
```

□

3.3.2 Heuristics, Distance computing and Threshold

Predicate invention results in generalizing the grammatical relation between verbs and head terms: the set of valid head terms for one verb and grammatical function is enriched by the addition of the valid head terms for another verb and grammatical function. For instance, the set of liquid terms valid as direct object of "to pour" is enriched by the set of liquid terms valid as place adjunct introduced by the preposition "in", after the verb "to drop". The induction leap due to predicate invention is controlled by a distance-based heuristic. GR clause pairs are considered as similar if their distance is less than a given threshold set by the user.

The distance Dist (Table 3), between two clauses G_i and G_j of the GR set depends on the proportion of the number of occurrences of *similar* variabilized examples covered by both clauses, and of the total occurrence number of examples in V.

- **Similarity between examples**

Two examples of V are *similar* if their head terms are the same. For example, V_1 and V_2 are similar.

V_1: verb(pour,X) :-
comp(dobj,Y,X), prep(none,Y), head(Z,Y), *milk*(Z).

V_2: verb(drop,X) :-
comp(padj,Y,X), prep(in,Y), head(Z,Y), *milk*(Z).

The set of variabilized examples covered by a GR clause G is denoted Cov(G). The set $Sim_{Gj}(G_i)$ is the set of examples of $Cov(G_i)$ such that there exists a similar example in $Cov(G_j)$. Notice that $|Sim_{Gj}(G_i)| = |Sim_{Gi}(G_j)|$. occ(VE) denotes the number of occurrences for the example VE.

- **Definition of the Distance, Dist**

The distance between two GR clauses G_i and G_j is defined as follows (Table 3).

Table 3. Asium distance.

$$if Sim_{G_j}(G_i) \neq \emptyset, Dist(G_i, G_j) = 1 - \frac{\log \sum_{v \in Sim_{G_i}(G_j)} Occ(v) + \log \sum_{v \in Sim_{G_j}(G_i)} Occ(v)}{\log \sum_{v \in Cov(G_i)} Occ(v) + \log \sum_{v \in Cov(G_j)} Occ(v)}$$

$$Dist(G_i, G_j) = 1, otherwise.$$

The only criterion used for choosing the clause pairs is distance. It is possible for members of a candidate pair to have been built at different generalization steps.

3.3.3 Example Generation

The distance between two GR clauses is computed on the basis of the occurrence number of the examples covered. Thus we want to easily associate each new GR clause G_i' with the variabilized examples it potentially covers and not only with the training examples it actually covers (the ones covered by G_i). We also want to associate a number of occurrences to the new examples. Asium generates all these new examples by a partial evaluation of G_i', (Van Harmelen and Bundy, 88), with respect to DT. New examples are then added to V extending $Cov(G_i')$. This amounts to generating one new variabilized example $V_i$new per example $V_j$old of $Cov(G_j)$ that is *not* in $Sim_{Gi}(G_j)$ so that $V_i$new is similar to $V_j$old and vice et versa.

Thus $Cov(G_i')$ and $Sim_{Gj,}(G_i')$ become equal. The number $Occ(V_i new)$ associated to the example $V_i$new is equal to $Occ(V_j old)$.

Example. Consider G_1' and G_2'.

```
G1': verb(pour,X) :-
       comp(dobj,Y,X), prep(none,Y),head(Z,Y),np(Z).

G2': verb(drop,X)  :-
       comp(padj,Y,X), prep(in,Y), head(Z,Y), np(Z).
```

np, np_1 and np_2 are defined in DT by, `np(X) :- np1(X). np(X) :- np2(X).`

```
np1(X) :- milk(X).   np1(X) :- water(X).   np1(X) :- wine(X).
np2(X) :- milk(X).   np2(X) :- water(X).   np2(X) :- cream(X).
```

For $Cov(G_1') = Sim_{G2}$, $Cov(G_1')$ the following new examples have to be generated:

```
verb(drop,X) :-
  comp(padj,Y,X), prep(in,Y), head(Z,Y), wine(Z).

verb(pour,X) :-
  comp(dobj,Y,X), prep(none,Y), head(Z,Y), cream(Z).
```

since "wine" is the only predicate that is less general than np1 and not less general than np_2 and "cream" is the only predicate that is less general than np_2 and not less general than np_1. □

3.3.4 Learning Result

Generalization ends when no GR clause can be generalized further: when there is no pair that is similar enough with respect to the threshold.

The learned ontology is the set of DT clauses that *are not* basic clauses. The subcategorization frame $SubCat_{Verb\_Id}$ of a given verb verb_Id, is the set of the most general GR clauses concluding with verb(Verb_Id,X). There is one such clause per valid concept for a given verb - grammatical relation. As DT is not hierarchical, learning can results in more than one most general clause for a given verb - grammatical relation.

4. Applications

To illustrate the potential applications of the proposed approach we present how learning results contribute to solving the semantic tagging, parsing disambiguation and language control problems. These components have been implemented in Asium for measuring performance with respect to the three tasks.

- Semantic tagging tags verb complement heads in the test corpus by the ontology concepts according to the verb subcategorization frames. Semantic tagging is a way to extend documents or user queries for Information Retrieval by enriching texts by synonyms or more abstract concepts than those actually occurring.
- The ontology and verb subcategorization frames help disambiguate parsing in two ways: by determining if a given noun phrase should be attached to a verb or to a

noun; and by determining the type of attachment to a verb (argument, adjunct and the type of the argument or adjunct).

- Language control checks the semantic validity of the heads of verb complements in the corpus according to the ontology and verb subcategorization frame.

These three tasks are based on the same logical operation: given a parsed clause, show that the verb subcategorization frame covers the parsed clause according to generalized subsumption with DT. Examples to be handled (disambiguated, controlled or tagged) should be pre-processed as described in Section 3.1, that is, parsed and split. Given an example E,

```
E:  verb_e(v_label_e,clause_id)  :-
      comp(gram_rel_e,phrase_id,clause_id),
      prep(prep_label,phrase_id),
      head(head_id,phrase_id), head_term_e(head_id).
```

The subcategorization frame Fr of the verb v_label, *covers* E iff there exists a clause G of Fr,

```
G : Verb_G(v_label_G,X) :-
      comp(gram_rel_G,Y,X), prep(prep_label_G,Y), head(Z,Y),
      head_term_G(Z).
```

that is more general than E according to SLD resolution with DT: G and E must have the same verb, they also must have the same grammatical relation (preposition plus type of argument), and the head term of G, $head\_term_G$, must be more general than $head\_term_e$ with respect to DT.

4.1 Semantic Tagging

Semantic tagging consists of listing all intermediate goals when proving $head\_term_G(Z)$, that is to say, listing the concepts of the ontology, the definition of which is needed for proving $head\_term_G(Z)$ given $head\_term_e(head\_id)$.

Example. G covers the example E,

```
E:  verb(drop,c1) :-
      comp(padj,p11,c1),prep(in,p11),head(h11,p11),wine(h11).

G:  verb(drop,X) :-
      comp(padj,Y,X), prep(in,Y),head(Z,Y), liquid(Z).
```

since DT says,

```
liquid(X) :- alcoholic_beverages(X).
alcoholic_beverages(X) :- wine(X).
```

and E is tagged as,

```
E:  verb(drop,c1) :-
      comp(padj,p11,c1), prep(in,p11), head(h11,p11),
      wine(h11), Alcoholic_beverages(h11), liquid(h11).  □
```

Notice that tagging an example differs from saturation: we do not want to add *all* concepts that are more general than *wine*, but only the ones that are relevant here. Relevancy depends on the *syntactic and semantic context* given by the subcategorization frame as highlighted in (Riloff, 93). Learning simple classes from co-occurrences in text-windows cannot provide a way to disambiguate the role of a term, but learning subcategorization frames can.

4.2 Parsing Disambiguation

Parsing disambiguation simply selects the parsing interpretation that is covered by the subcategorization frames and removes the others. When no interpretation is left, a possible parsing can be suggested by abduction as in (Duval, 91).

Example. Of the two possible interpretations

```
E:  verb(cook,c1) :-
     comp(dobj,p11,c1),prep(none,p11),head(h11,p11),3_minutes(h
11).
E': verb(cook,c1) :
     comp(dadj,p11,c1),prep(none,p11),head((h11,p11),3_minutes(h
11).
```

the second one is correct, according to `C` and DT, saying that `3_minutes` is a `duration`, where `C` is given below.

```
C:  verb(cook,X) :-
      comp(dadj,Y,X),  prep(none,Y),  head(Z,Y),  duration(Z).
```

□

If the parser would not have built the second and correct interpretation, but only the first one, the Asium disambiguating component would have suggested it by abducing `comp(dadj,p11,c1).` When only one literal lacks among the four needed, it is abduced in order to complete the proof.

4.3 Language Control

Language control checks the syntactic validity of the verbal grammatical relations, and the semantic validity of the heads. If there is no clause in the subcategorization frame covering the example to be tested, the example is considered as invalid. In particular, it allows one to detect metonymies. A possible replacement of the invalid head can be suggested by abduction in a similar way as when disambiguating.

For example, `C` does not cover the example `E`, as "glass" is not defined as a "liquid" in DT. "liquid" can be suggested for replacing "glass".

```
E:  verb(drink,c1) :-
      comp(dobj,p11,c1),prep(none,p11),head(h11,p11),glass(h11
).
C:  verb(drink,X) :-
      comp(dobj,Y,X), prep(none,Y), head(Z,Y),liquid(Z).
```

5. Future work

The training example set ranges from large to very large. Asium, like other similar methods, learns grammatical relations for a given verb independently of one another for reasons of efficiency. As a consequence, concepts filling the selection restrictions can be overgeneral for some tasks like query extension in information retrieval where computational efficiency is crucial.

For instance, the learned subcategorization frame of "to cook" will be,

```
C1: verb(cook,X) :-
      comp(dobj,Y,X), prep(none,Y), head(Z,Y), cake(Z).

C2: verb(cook,X) :-
      comp(dobj,Y,X), prep(none,Y), head(Z,Y), eggs(Z).

C3: verb(cook,X) :-
      comp(tadj,Y,X), prep(for,Y), head(Z,Y), duration(Z).
```

It says that cakes and eggs can be cooked in any duration, although eggs should not be cooked more than 12 minutes. A user query "how long should eggs be cooked? " would trigger a search through the cooking recipe base for all combinations of "eggs" and "duration" defined in the ontology instead of only the relevant ones. Learning grammatical relations independently has another consequence: the properties of the grammatical relations of a given verb such as mutual exclusion, optionality or requirement are not learned. For instance in the cooking recipe corpus, the time adjuncts of "to cook", "for - duration" and "duration" are mutually exclusive and the preposition "for" is omitted when the direct object is present. We are developing a post-processing method based on the method HAIKU (Nedellec et al., 96) and the language CARIN (Levy & Rousset, 98) in order to learn such dependencies. It will both specialize the overgeneral selection restrictions and learn dependencies between verb complements.

Clustering based on FOL distances (such as the ones of (Esposito et al., 91), (Bisson, 92), (Kirsten & Wrobel, 98)) instead of the Asium distance could help to control the generalization of dependent selection restrictions. They are not applicable here for reasons of complexity. For instance, the cooking recipe corpus contains 90,000 examples. Up to 800 concepts and 1000 verb subcategorization frames have to be learned in parallel. However such distances could be successfully applied to learning predicate schemata from verb subcategorization frames and noun frames.

6. Related work

In this paper, we have presented the ILP method Asium which learns ontologies and verb subcategorization frames from a parsed corpus in an unsupervised way.

As proposed by the work reported in (Hindle, 90), (Pereira et al., 93), (Grishman & Sterling, 94) and (Grefenstette, 92), among others, Asium clusters terms on the basis of syntactic regularities observed in a parsed corpus. The clustered terms are heads of

verb complements, arguments and adjuncts. Asium differs from both Hindle's (90) and Grefenstette's (92) methods where adjuncts are not considered for learning. Instead, Hindle's method only considers arguments while Grefenstette's method considers arguments and noun relations (adjectival and prepositional). Experiments with the cooking recipe corpus and the Pascal corpus of INIST[2] have shown that considering not only arguments but also adjuncts yields better results in terms of precision and recall. Further experiments are performed with the Mo'K system (Bisson et al.) for comparing the results when learning from noun relations as proposed by Grefenstette (92), and Grishman and Sterling (94).

The way Asium clusters terms for building hierarchies of concepts fundamentally differs from the clustering methods described in (Pereira et al., 93), (Hogenhout & Matsumoto, 97) and more generally, from those applied in conceptual clustering. As the goal is to build classes of terms, terms are viewed as the examples, i.e. the objects to cluster. The examples are described by their attributes; that is to say, their syntactic context (verb plus grammatical relation) in the learning corpus. Notice that verbs are viewed as the objects when learning verb classes as in (Basili & Pazienza, 97). Bottom-up clustering usually computes the distances between pairs of objects according to the attributes they have in common. The best pair is selected, the two objects clustered, and clustering goes on until a tree is built with a single class containing all objects at its top. This strategy builds deep trees with many intermediate useless concepts and the concepts at the lowest levels contain very few terms. The novel strategy proposed here is to compute distances between *all pairs of attributes* and to cluster the two sets of objects which are described by the closest *pair* of attributes. Thus the number of terms in the classes is much larger and the tree much shallower. This improves the readability of the tree and the efficiency of its use. One effect could be a lack of precision; however, preliminary experiments on the two corpora cited above did not show major differences in precision but a notable reduction of tree size. Further experiments would be needed in order to characterize the properties of the corpora for which this strategy would be preferable.

The ILP approach proposed here remains applicable in all the four cases, clustering terms versus clustering verbs, and clustering objects as usual, versus clustering attributes as in Asium. It could thus be usefully used for modeling previous work on clustering terms in an ILP framework.

Acknowledgement

This work has been partially supported by the CEC through the ESPRIT contract LTR 20237 (ILP 2).

References

1. Basili R. & Pazienza M. T., "Lexical acquisition for information extraction" in *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, M. T. Pazienza (Ed.), (pp.14-18), Lecture Notes in Artificial Intelligence Tutorial, Springer Verlag (Pub.), Frascati, Italy, July 1997,
2. Bisson G., "Learning in FOL with a similarity measure", in *Proceedings of the Tenth National Conference en Artificial Intelligence*, (pp. 82-87), San Jose, AAAI Press / The MIT Press (Pub.), July, 1992.

[2]Pascal is a base of scientific paper abstracts on agriculture, maintained by INIST.

3. Bisson G., Nedellec C. & Canamero L., "Clustering methods for ontology learning: The Mo'K workbench", in *Proceedings of the European Conference on Artificial Intelligence Workshop on Ontology Learning*, Staab S. et al. (Eds), Berlin, 2000 (in press).
4. Buntine W., "Generalized subsumption and its application to induction and redundancy", in *Artificial Intelligence* 36, (pp. 375-399), 1988.
5. Cohen W. W., "Cryptographic limitations on learning one-clause logic program" in *Proceedings of the Tenth National Conference on Artificial Intelligence*, Washington D.C., 1993.
6. Constant P., "L'analyseur linguistique SYLEX", Fifth CNET summer school, 1995.
7. Dagan I., Lee L., & Pereira F., "Similarity-based methods for word-sense disambiguation", in *Proceedings of the Annual Meeting of the Association for Computational Linguistics,* 1996.
8. Duval B., "Abduction for explanation-based learning", in Proceedings of *the European Working Session on Learning*, (pp. 348-360), Lecture Notes in Artificial Intelligence, Y. Kodratoff (Ed.), Springer Verlag (Pub.), March 1991.
9. Esposito F., Malerba D. & Semeraro G., "Flexible matching for noisy structural descriptions.", in *Proceedings of Twelfth International Joint Conference on Artificial Intelligence*, (pp. 658-664), Sydney, August, 1991.
10. Faure D. & Nedellec C.,"Knowledge acquisition of predicate-argument structures from technichal texts using machine learning", in *Proceedings of Current Developments in Knowledge Acquisition*, D. Fensel & R. Studer (Ed.), Springer Verlag (Pub.), Karlsruhe, Germany, May 1999.
11. Hindle D., "Noun classification from predicate-argument structures", in *Proceedings of the 28st annual meeting of the Association for Computational Linguistics*, (pp. 1268-1275), Pittsburgh, PA, 1990.
12. Grefenstette G., "SEXTANT: exploring unexplored contexts for semantic extraction from syntactic analysis", in *Proceedings of the Thirtieth Annual Meeting of the Association of Computational Linguistics*, (pp. 14-18), 1992.
13. Grishman R. & Sterling J., "Generalizing automatically generated selectional patterns", in *Proceedings of the Sixteenth International Conference on Computational Linguistics*, 1994.
14. Harris Z., Gottfried M., Ryckman T., Mattick Jr P., Daladier A., Harris T. & Harris S., *The form of information in science, analysis of immunology sublanguages*, Kluwer Academic (Pub.), Dordrecht, 1989.
15. Hogenhout W. R. & Matsumoto Y., "A preliminary study of word clustering based on syntactic behavior", *Proceedings of Thirty-fifth Annual Meeting of the Association of Computational Linguistics*, 1997.
16. Kirsten M. & Wrobel S., "Relational distance-based clustering", in *Proceedings of the Eighth workshop on Inductive Logic Programing,* Page D. (ed.), (pp. 261-270(, Springer Verlag (Pub.), Madison, 1998.
17. Levy A. & Rousset M. C. "Combining Horn rules and description Logics in CARIN", in *Artificial Intelligence Journal*, vol 104, 165-210, September 1998.
18. Muggleton S. & Buntine W., "Machine invention of first order predicates by inverting resolution", in *Proceedings of the Fifth International Machine Learning Worksho*, Morgan Kaufman (Pub.), (pp. 339-352), 1988.
19. Nedellec C., "How to specialize by theory refinement", in *Proceedings of the Tenth European Conference on Artificial Intelligence,* (pp. 474-478), Neuman B. (Ed.), John Wiley & sons (Pub.), Vienna, August, 1992.
20. Nedellec C., Rouveirol C., Ade H., Bergadano F. & Tausend B.,"Declarative bias in inductive logic programming" in *Advances in Inductive Logic Programming*, 82-103, de Raedt L. (Ed.), IOS Press (Pub.), 1996.

21. Pereira F., Tishby N. & Lee L., "Distributional clustering of English words" in *Proceedings of the 31st annual meeting of the Association for Computational Linguistics,* (pp. 183-190), 1993.
22. Resnik P. & Hearst M. A. "Structural ambiguity and conceptual relations", in *Proceedings of Workshop on Very Large Corpora: Academic and Industrial Perspectives*, (pp. 58-64), Ohio State University, 1993.
23. Resnik P., "Using information content to evaluate semantic similarity in a taxonomy.", in *Proceedings of the International Joint Conference on Artificial Intelligence,* Montreal, 1995.
24. Rouveirol C., "Flattening and saturation: two representation changes for generalization", in *Machine Learning, 14*, 219-232, Kluwer Academic (Pub.), Boston, 1994.
25. Riloff H., "Automatically constructing a dictionary for information extraction tasks", in *Proceedings of the 11th National Conference on Artificial Intelligence, (*pp. 811-816), AAAI Press / MIT Press (Pub.), 1993.
26. Sekine S., Caroll J. J., Ananiadou S. et Tsujii J., "Automatic learning for semantic collocation" in *Proceedings of the Third Conference on Applied Natural Language Processing,* (pp. 104-110), Trente, Italy, 1992.
27. Van Harmelen F. & Bundy A., «Explanation based generalization = partial evaluation», in *Artificial Intelligence* 36, 401-412, 1988.
28. Yarowsky D., "Word-Sense disambiguation using statistical models of Roget's categories trained on large corpora", in *Proceedings of the International Conference on Computational Linguistics*, (pp. 454-460), Nantes, 1992.