Inducing Constraint-based Grammars from a Small Semantic Treebank

Smaranda Muresan
Department of Computer Science
Columbia University
New York, NY
smara@cs.columbia.edu

Tudor Muresan
Department of Computer Science
Technical Univ. of Cluj-Napoca
Cluj-Napoca, Romania
tmuresan@cs.utcluj.ro

Judith L. Klavans
Department of Computer Science
Columbia University
New York, NY
klavans@cs.columbia.edu

Abstract

We present a relational learning framework for grammar induction that is able to learn meaning as well as syntax. We introduce a type of constraint-based grammar, *lexicalized well-founded grammar (lwfg)*, and we prove that it can always be learned from a small set of semantically annotated examples, given a set of assumptions. The semantic representation chosen allows us to learn the constraints together with the grammar rules, as well as an ontology-based semantic interpretation. We performed a set of experiments showing that several fragments of natural language can be covered by a *lwfg*, and that it is possible to choose the representative examples heuristically, based on linguistic knowledge.

Introduction

In this paper we introduce a theoretic type of grammar, called *lexicalized well-founded grammar*, which facilitates the learning of both meaning and syntax, starting with simple rules in a bottom-up fashion. This approach to learning follows the argument that language acquisition is an incremental process, in which simpler rules are acquired before complex ones (Pinker 1989). Moreover, using semantic information during acquisition can facilitate learning only from positive data for some classes of grammars (Oates *et al.* 2003). Grammar induction is achieved using a relational learning framework based on a set of *representative examples* (Muresan, Muresan, & Potolea 2002). There are two key features of this set: 1) the examples are ordered, allowing the bottom-up learning of the grammar; and 2) the size of the set is small, which is essential because large semantically annotated treebanks are hard to build.

Learning meaning as well as syntax from a small number of training examples is useful for rapid text-to-knowledge acquisition, replacing the process of rewriting grammars by hand for each specific domain. This is especially suited for domains that are not well-covered by existing syntactic and semantic parsers, such as the medical domain. The framework is currently applied to building a terminological knowledge base in that domain. Another application for this framework is a tool for linguists, who will be able to build and test their own models of language learning.

In order to model the lexicalized well-founded grammar, we propose a constraint-based grammar formalism which augments a context free grammar with: 1) a semantic representation, called *semantic molecule*, and 2) two constraints: one for semantic composition and one for ontological validation. This constraint-based grammar allows an interleaved treatment of syntax and semantics. The *semantic molecule* encodes both the semantic representation of a natural language expression (given as a flat representation) and the information necessary for semantic composition (given as a one level feature structure). This reduced representation, while expressive enough to model syntactic/semantic aspects of natural language, is simple enough to be effectively used in our relational learning framework. Given this, our grammar formalism allows both the grammar and the compositionality constraints to be learned. Another feature of our formalism is that ontological constraints are applied at the rule level, so that the semantic interpretation is limited not by the grammar, as in general linguistic frameworks, but by the complexity of the domain ontology. Thus, we have an ontology-based semantic interpretation of natural language expressions, which refrains from full logical analysis of meaning. Our framework is suitable for domain specific texts that have a domain ontology as semantic model. This is applicable for domains with large ontologies (e.g. the medical domain). The ontology will guide the grammar acquisition. This follows the argument that, during language acquisition, in order for syntax to be acquired properly, semantic distinctions and conceptual information should be known (Pinker 1989).

The paper is organized as follows. First, we introduce the theoretical concepts of *lexicalized well-founded grammar* and *representative examples*, as well as the assumptions we consider for the grammar induction problem. Second, we present the constraint-based grammar formalism that models the lexicalized well-founded grammar, describing the *semantic molecule* and the constraint rules. Next we describe our grammar induction algorithm together with a theorem stating that given the assumptions mentioned above, the induction of a lexicalized well-founded grammar, from a small unambiguous set of representative examples, can always be done. We conclude with a set of experiments and results of our grammar induction framework.

$A1 \Rightarrow Adj$
$N1 \Rightarrow Noun$
$N1 \Rightarrow A1\ \ N1$
$N2 \Rightarrow Det\ \ N1$
$N2 \Rightarrow N2\ \ Rc1$
$V1 \Rightarrow Tv$
$Rc1 \Rightarrow Rpro\, V1\, N2$
$Rc1 \Rightarrow \dots$

(a)

P_G	Level	Nonterminal sets	R_G
	1	$\{Adj, Noun, Tv, Det, Rpro\}$	
$A1 \Rightarrow Adj$ $N1 \Rightarrow Noun$ $V1 \Rightarrow Tv$	2	$\{A1, N1, V1\}$	$A1 \succeq Adj$ $N1 \succeq Noun$ $V1 \succeq Tv$
$N1 \Rightarrow A1\, N1$ $N2 \Rightarrow Det\, N1$	3	$\{N2\}$	$N1 \succeq A1, N1 \succeq N1$ $N2 \succeq Det, N2 \succeq N1$
$Rc1 \Rightarrow Rpro\, V1\, N_2$	4	$\{Rc1\}$	$Rc1 \succeq Rpro, Rc1 \succeq V1, Rc1 \succeq N2$
$N2 \Rightarrow N2Rc1$			$N2 \succeq N2$

(b)

Figure 1: (a) Grammar, G; (b) Iterative steps for the Well_Founded_Grammar(G) algorithm

Theoretical Concepts

A context-free grammar is a 4-tuple $G = \langle N, \Sigma, P_G, S\rangle$ where N is an alphabet of nonterminal symbols, Σ is an alphabet of terminal symbols with $N \cap \Sigma = \emptyset$, P_G is a set of production rules, where every production is a pair (A, β) with $A \in N$ and $\beta \in \{N \cup \Sigma\}^*$ (also denoted $A \Rightarrow \beta$), and $S \in N$ is the start nonterminal symbol. Let's consider the ground derivation rule $\beta \Rightarrow^* w$ with $w \in \Sigma^*$ such that: $\frac{A \Rightarrow w}{A \Rightarrow^* w}$, $\frac{\beta = B_1 \dots B_n \qquad B_i \Rightarrow^* w_i,\ i=1,\dots,n}{\beta \Rightarrow^* w_1 \dots w_n}$, and $\frac{A \Rightarrow \beta \qquad \beta \Rightarrow^* w}{A \Rightarrow^* w}$. We have $A \Rightarrow \beta \Rightarrow^* w$. The language generated by the grammar G is $L(G) = \{w \in \Sigma^* \colon S \Rightarrow^* w\}$ and the set of substrings generated by G is $L_s(G) = \{w \in \Sigma^* \colon A \in N, A \Rightarrow^* w\}$. The set of substrings generated by the rule $A \Rightarrow \beta$ is $L_s(A \Rightarrow \beta) = \{w \in \Sigma^* \colon (A \Rightarrow \beta) \in P_G, A \Rightarrow \beta \Rightarrow^* w\}$. We consider that the set of nonterminals N is a well-founded set based on the partial ordering relation $\succeq$ among the nonterminals. Thus $\forall A \in N$ and $\forall t \in \Sigma$, we have $S \succeq A \succeq t$, where we used the same notation for the reflexive, transitive closure of $\succeq$.

Definition 1. A rule $(A \Rightarrow \beta) \in P_G$ is called an *ordered rule*, if $\forall (B \in N) \subseteq \beta \colon A \succeq B$.

Definition 2. A Context Free Grammar, G, is called a *well-founded grammar (wfg)* if:

1. The set of nonterminal symbols is well founded.
2. Every nonterminal symbol is a left-hand side in at least one ordered non-recursive rule.
3. The empty string ε cannot be derived from any nonterminal symbol.

In a well-founded grammar, there are three types of rules in which a nonterminal can appear: ordered non-recursive rules, ordered recursive rules or non-ordered rules. The ordered non-recursive rules ensure a termination condition for the ground derivation, essential to the inductive process.

Proposition 1. Every context-free grammar can be efficiently tested to see whether it is a well-founded grammar, by Algorithm 1. This algorithm assigns a level l to every nonterminal A, $A \in N_l$, and returns the set R_G of partial ordering relation among the nonterminals. The efficiency of the algorithm is O($|P_G|^2 * |\beta|$).

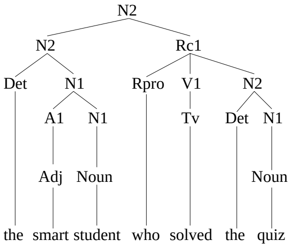
Lemma 1. *A context-free grammar $G = \langle N, \Sigma, P_G, S\rangle$ is a well-founded grammar $G = \langle N, \Sigma, P_G, S, R_G\rangle$ iff $R_G \neq \emptyset$ is returned by Algorithm 1.*

Algorithm 1: Well_Founded_Grammar(G)

```
$R_G \leftarrow \emptyset, N_0 \leftarrow \Sigma, P \leftarrow P_G, V \leftarrow \emptyset, l \leftarrow 0$
while $P \neq \emptyset$ and $N_l \neq \emptyset$ do
    $V \leftarrow V \cup N_l, l \leftarrow l + 1, N_l \leftarrow \emptyset$
    foreach $(A \Rightarrow \beta) \in P$ and $\beta \subseteq V$ do
        $P \leftarrow P - \{A \Rightarrow \beta\}$
        if $A \notin V$ then
            $N_l \leftarrow N_l \cup \{A\}$
            foreach $(B \in N) \subseteq \beta$ do
                $R_G \leftarrow R_G \cup \{A \succeq B\}$
        else
            foreach $(B \in N) \subseteq \beta$ and $\neg(A \succeq B$ or $B \succeq A)$ do
                if $A \in N_i$ and $B \in N_j$ and $i \geq j$ then
                    $R_G \leftarrow R_G \cup \{A \succeq B\}$
                else
                    $R_G \leftarrow R_G \cup \{B \succeq A\}$
if $P = \emptyset$ and $\epsilon \notin \Sigma$ then return $R_G$ else return $\emptyset$
```

Example. Figure 1 gives an example of a partial grammar for noun phrases with relative clauses and the iterative steps of Algorithm 1. As can be seen, in this grammar, $A1 \rightarrow Adj$, $N1 \rightarrow Noun$, $N2 \rightarrow Det\ \ N1$ are examples of ordered non-recursive rules; $N1 \rightarrow A1\ \ N1$ is an example of an ordered recursive rule, while $N2 \rightarrow N2\ \ Rc1$ is a non-ordered rule, since $Rc1$ is a bigger nonterminal than $N2$, i.e. $Rc1 \succeq N2$ (see Figure 1(b)). We use Rc to denote relative clauses. The last rule means that there can be other alternatives for relative clauses (e.g. reduced relative clauses).

A well-founded grammar, G, induces a partial ordering relation on any generated sublanguage $E \subseteq L(G)$. For each string $e \in E$ ($S \Rightarrow^* e$) and for each substring e_i of e, we choose (if it exists) the smallest nonterminal A_i, so that $A_i \Rightarrow^* e_i$. In other terms, if S is the root of the syntactic tree for string e, A_i is the root of the subtree of e_i. Let C_i be the biggest nonterminal of the subtree rooted at A_i (for ordered non-recursive rules: $C_i \prec A_i$, for ordered recursive rules: $C_i = A_i$, and $C_i \succeq A_i$ otherwise). The equivalence class of substring e_i is (A_i) for an ordered non-recursive rule, $({A_i}^r)$ for an ordered recursive rule, and (C_i, A_i) for a non-ordered rule, respectively. A lexicographic ordering



(a)

E_R	Eq. class	Min-Rule $\in P_{e_0}$
smart	(A_1)	$A_1 \Rightarrow Adj$
student	(N_1)	$N_1 \Rightarrow Noun$
solved	(V_1)	$V_1 \Rightarrow Tv$
smart student	$({N_1}^r)$	$N_1 \Rightarrow A_1\ N_1$
the quiz	(N_2)	$N_2 \Rightarrow Det\ N_1$
who solved the quiz	(Rc_1)	$Rc_1 \Rightarrow Rpro\ V_1\ N_2$
the student who solved the quiz	(Rc_1, N_2)	$N_2 \Rightarrow N_2\ Rc_1$

(b)

Figure 2: (a) Parse tree ; (b) Results of Algorithm 3

is assumed. Thus the equivalence classes introduce a partial ordering relation among the substrings of the sublanguage E: $e_i \succeq e_j$ iff $(C_i, A_i) \succeq_{lex} (C_j, A_j)$.

Algorithm 2, given below, returns the topologically sorted set E_s of substrings e_i based on the partial ordering relation and the substring length $|e_i|$: $e_m \geq ... \geq e_i \geq ... \geq e_0$. The algorithm is polynomial in $|E|$ and $|e|$. $Find\_Min\_Rule$ and $Max\_Nonterminal$ are efficiently performed by a bottom-up active chart parser (Kay 1973). This parser also computes the set of rules, P_{e_0}, from which the substring e_0 can be ground derived (see Algorithm 3).

Algorithm 2: Substring_Equivalence_Classes(E, G)

```
foreach $j$ do
    Eq_Class(j) ← $\emptyset$
foreach $e \in E$ do
    foreach $e_i \subseteq e$ do
        Find_Min_Rule $A_i \Rightarrow \beta \Rightarrow^* e_i$
        if $(A_i \Rightarrow \beta) \neq \emptyset$ then
            Max_Nonterminal($A_i \Rightarrow \beta$, $C_i$)
            if $A_i \succeq C_i$ and $C_i \not\succeq A_i$ then $j \leftarrow (A_i)$
            else if $A_i \succeq C_i$ then $j \leftarrow ({A_i}^r)$
                else $j \leftarrow (C_i, A_i)$
            Eq_Class($j$) ← Eq_Class($j$) ∪ $\{e_i\}$
    Topological_Sort(Eq_Class($j$), $E_s$)
    return $E_s$
```

Definition 3. We call *representative example set*, the set E_R obtained through Algorithm 3.

Algorithm 3: Find_Representative_Example(E, G)

```
$E_s \leftarrow$ Substring_Equivalence_Classes($E, G$)
$E_R \leftarrow \emptyset$ , $P_{G_r} \leftarrow \emptyset$
repeat
    $e_0 \leftarrow$ Extract_Min($E_s$)
    $P_{e_0} \leftarrow \{(A \Rightarrow \beta) \in P_G \colon e_0 \in L_s(A \Rightarrow \beta)\}$
    $P_{G_r} \leftarrow P_{G_r} \cup P_{e_0}$
    $E_s \leftarrow E_s - L_s(G_r)$
    $E_R \leftarrow E_R \cup \{e_0\}$
until $E_s = \emptyset$
return ($E_R$, $P_{G_r}$)
```

Theorem 1 (Representative Examples Theorem). *Given a well-founded grammar G, a subset E of $L(G)$, and a topologically sorted set E_s (generated using Algorithm 2), Algorithm 3 generates in polynomial time the representative example set, E_R, together with the associated grammar G_r, such that $E \subseteq L(G_r) \subseteq L(G)$ and $|E_R| \leq |P_{G_r}|$.*

Proof. Given the grammar G, $\forall e \in E$, we have $S \Rightarrow^* e$. Algorithm 2 assures that for every substring e_i of e, which has a syntactic subtree rooted in A_i belonging to the syntactic tree of e rooted in S, we have $e_i \in E_s$. But Algorithm 3 assures that every time a substring e_i is removed from the set E_s, the associated rules $A_i \Rightarrow \beta$ are added to the grammar set P_{G_r}. Thus G_r contains all the rules used to obtain the syntactic tree rooted in S. In other words we have $S \Rightarrow^* e$ in grammar G_r, for each $S \Rightarrow^* e$ in grammar G. Thus $E \subseteq L(G_r) \subseteq L(G)$. It is straightforward that the algorithm is polynomial in $|E|$ and $|e|$, and $|E_R| \leq |P_{G_r}|$. □

The above theorem states that if a well-founded grammar G covers a set of representative examples E_R, the grammar covers the sublanguage E, and the size of E_R is small.

Example. Figure 2(b) shows the results of Algorithm 3 given the sublanguage of the noun phrase “the smart student who solved the quiz” and the grammar in Figure 1(a), while Figure 2(a) shows the corresponding parse tree.

Both in formal and linguistic theories of grammars, lexicalization is an important factor. Lexicalized grammars are finitely ambiguous and thus decidable (Joshi & Schabes 1997). In our framework, we consider *lexicalized well-founded grammars*, where the string w is paired with a semantic representation, called *semantic molecule* and denoted w'. The semantic molecule encodes both the semantic representation of the string, given as a flat representation, and the information connected to the semantic head, necessary for composition.

Definition 4. Let G be a well-founded grammar. We call G a *lexicalized well-founded grammar* if all substrings w, derived from a nonterminal A have the same category of their semantic molecules w'. That is, there exists a mapping from the nonterminal set N to the category set C, $c\colon N \rightarrow C$, such that $(w_1, w_1'), (w_2, w_2') \in L_s(A) \rightarrow h_1.cat = h_2.cat = c(A)$ (see the next section for the definition of semantic molecule).

For these grammars, the ground derivation rule becomes $\beta \Rightarrow^* (w, w')$, such that:

$\frac{A \Rightarrow (w,w')}{A \Rightarrow^* (w,w')}$, $\frac{\beta = B_1 \ldots B_n \qquad B_i \Rightarrow^* (w_i, w_i'),\ i=1,\ldots,n}{\beta \Rightarrow^* (w_1 \ldots w_n, w_1' \circ \cdots \circ w_n')}$, and $\frac{A \Rightarrow \beta \qquad \beta \Rightarrow^* (w,w')}{A \Rightarrow^* (w,w')}$, where $w = w_1 \ldots w_n$ is the concatenation of strings, and $w' = w_1' \circ \cdots \circ w_n'$ is the composition of semantic molecules. The semantic molecule composition requires the enhancement of grammar rules with compositional semantic constraints.

Grammar Induction Problem. *Given $E \subseteq L(G)$ of an unknown lexicalized well-founded grammar G, together with its set of representative examples E_R, $E_R \subseteq E_s$, learn a grammar G_r such that $E \subseteq L(G_r) \cap L(G)$.*

Considering the following three assumptions, we prove that the induction can always be done (see Theorem 2).

Assumption 1. We assume that the grammar G is not redundant, i.e. it does not contain equivalent nonterminals or rules: $A_i \neq A_j \quad iff \quad L_s(A_i) \neq L_s(A_j)$, and $A \Rightarrow \beta_i \neq A \Rightarrow \beta_j \quad iff \quad L_s(A \Rightarrow \beta_i) \neq L_s(A \Rightarrow \beta_j)$ respectively.

Assumption 2. We assume that the subset E is unambiguous [1]. In this case, in Algorithm 3, we have $P_{e_0} = \{A_0 \Rightarrow \beta\} \cup chr(A_0)$, where $A_0 \Rightarrow \beta$ is a minimal rule and $chr(A_0) = \{A_k \Rightarrow A_{k-1}, \ldots, A_1 \Rightarrow A_0\}$, is a chain of rules ended in A_0 from which the string e_0 can be ground derived ($A_k \succeq A_{k-1} \succeq \cdots \succeq A_0$). We call $chs(A_0) = \{A_0, \ldots, A_k\}$ the chain set. From Assumption 1 we have $L_s(A_0) \subset L_s(A_1) \subset \cdots \subset L_s(A_k)$.

Lemma 2. *Let G be an unambiguous, nonredundant lexicalized well-founded grammar, G, and $A_0 \Rightarrow \beta$ a minimal rule resulted from Algorithm 3. If β' is formed from β by substituting a nonterminal B_j with a nonterminal B'_j of the same chain, $B_j, B'_j \in chs(j)$, and $B'_j \succeq B_j$ then $L_s(A_0 \Rightarrow \beta') \supseteq L_s(A_0 \Rightarrow \beta)$.*

Assumption 3. We assume that the subset E is rich enough, so that the ground derivation of E_s covers all grammar rules. Thus for each equivalence class (C_i, A_i), classes (A_i) and $({A_i}^r)$ are also generated. Since $(A_i) \preceq ({A_i}^r) \preceq (C_i, A_i)$, for each nonterminal A the learning algorithm will first learn the ordered non-recursive rules, then the ordered recursive rules and last the non-ordered rules, unless the rules belong to the chain of rules $chr(A)$ (see Assumption 2 and the proof of Theorem 2). In the absence of this assumption, the learning machinery might need theory revision steps.

Constraint-Based Grammar

The grammar is based on the Definite Clause Grammar (DCG) formalism (Pereira & Warren 1980). In our model we augment the nonterminals with a semantic representation, called *semantic molecule*, allowing an interleaved treatment of syntax and semantics, and we add two constraints to the grammar rule: one for semantic compositionality and one for ontological validation.

Semantic Molecule

The semantic representation we adopt in this work bares some similarities with Minimal Recursion Semantics (Copestake, Lascarides, & Flickinger 2001). We denote by $w' = h \bowtie b$ the semantic molecule of a string w, where h is the head acting as a valence for molecule composition, and b is the body acting as a flat semantic representation of the string w.

The valence, h, of the molecule includes all the necessary information for molecule linking (i.e., semantic composition) in the form of a one level feature structure (i.e., feature values are atomic). In Figure 3 we present the semantic molecules for one adjective and one noun. The heads of these two molecules are given as attribute-value matrices (AVMs). Let $\mathcal{A}_h$ be the set of attributes used in the head, h, of the semantic molecule $w' = h \bowtie b$. An element of h has the form $h.a = val$, where $a \in \mathcal{A}_h$ and val is either a constant or a logical variable. The set of logical variables of the head, h, is denoted by $var(h)$. For example, Figure (3a) shows the semantic molecule for the adjective "smart" whose head, h, has three attributes: cat, which is the syntactic category, $head$, which is the index of the head, and mod which is the index of the modified noun.

The body, b, of the molecule is a Canonical Logical Form (CLF) using as semantic primitives a set of atomic predicates (APs) based on the traditional concept of attribute-value pair:

$$
\begin{array}{llll}
(1) & \langle\text{CLF}\rangle & \longrightarrow & \langle\text{AP}\rangle \\
 & & & \mid \langle\text{CLF}\rangle\,\langle\text{lop}\rangle\,\langle\text{CLF}\rangle \\
 & \langle\text{AP}\rangle & \longrightarrow & \langle\text{concept}\rangle\,\textbf{.}\,\langle\text{attr}\rangle = \langle\text{concept}\rangle
\end{array}
$$

The *lop* is the general logical operator, while *concept* correspond to a frame in an ontology and *attr* is a slot of the frame, encoding either a property or a relation. This frame-based representation is motivated by our ontology-based semantic interpretation approach, which has recently been considered, due especially to the growing interest for the Semantic Web (Jensen & Nilsson 2003). For example, Figure (3a) shows the body of the semantic molecule for the adjective "smart". It has two predicates: $X_1.isa = smart$, encoding the lexical sign and $X_2.Has\_prop = X_1$, meaning that the adjective X_1 is a value of a slot in a frame corresponding to the noun denoted by X_2. The variable *Has_prop* will be instantiated after the interpretation on the ontology.

For the composition of several semantic molecules, if $\circ$ is the semantic composition operator, we have:

$$
\begin{aligned}
(2) \qquad w' = h \bowtie b &= (w_1 \ldots w_n)' = w_1' \circ \cdots \circ w_n' \\
&= (h_1 \bowtie b_1) \circ \cdots \circ (h_n \bowtie b_n) \\
&= h_1 \circ \cdots \circ h_n \bowtie b_1, \ldots, b_n
\end{aligned}
$$

Thus the composition affects only the molecule head, while the body parts are connected through the conjunctive connector. We denote by $\Phi_{comp}(h, h_1, ..., h_n)$ the compositional semantic constraints, which are added at the grammar rule level. They are encoded as a system of equations (Eq (4)) given below.

$$
(4a) \qquad \left\{ \begin{array}{l} h.a = constant \\ h.a = h_i.a_i \end{array} \right. \quad \text{where} \quad \begin{array}{c} 1 \leq i \leq n \\ a \in \mathcal{A}_h, a_i \in \mathcal{A}_{h_i} \end{array}
$$

[1] Unambiguity is provided by the lwfg constraints (even if syntactically the sublanguage might be ambiguous). For learning, the given annotated examples are disambiguated.

(3a)

$$(smart/adj)' = h_1 \bowtie b_1 \qquad \text{where} \quad h_1.cat = adj, h_1.head = X_1, h_1.mod = X_2$$

$$= \begin{bmatrix} cat & adj \\ head & X_1 \\ mod & X_2 \end{bmatrix} \bowtie [X_1.isa = smart, X_2.Has\_prop = X_1] \qquad \begin{array}{c} \mathcal{A}_{h_1} = \{cat, head, mod\}, \\ var(h_1) = \{X_1, X_2\}, \\ var(b_1) = \{X_1, X_2, Has\_prop\} \end{array}$$

(3b)

$$(student/n)' = h_2 \bowtie b_2 \qquad \text{where} \quad h_2.cat = n, h_2.head = X_3$$

$$= \begin{bmatrix} cat & n \\ head & X_3 \end{bmatrix} \bowtie [X_3.isa = student] \qquad \begin{array}{c} \mathcal{A}_{h_2} = \{cat, head\} \\ var(h_2) = \{X_3\}, var(b_2) = \{X_3\} \end{array}$$

(3c)

$$(smart\ \ student)' = h \bowtie b = (smart)' \circ (student)' = h_1 \bowtie b_1 \circ h_2 \bowtie b_2 = h_1 \circ h_2 \bowtie b_1, b_2 \quad \%(from\ \text{(2)})$$

$$= \begin{bmatrix} cat & n \\ head & X_2 \end{bmatrix} \bowtie [X_1.isa = smart, X_2.Has\_prop = X_1, X_2.isa = student] \quad \%(\text{(3a), (3b)}, \Phi_{comp}(h, h_1, h_2))$$

$$\Phi_{comp}(h, h_1, h_2) = \{h.cat = h_2.cat, h.head = h_1.mod, h_1.cat = adj, h_1.mod = h_2.head, h_2.cat = n\} \ \%(from\ \text{(4a)},\ and\ \text{(4b)})$$

Figure 3: Examples of semantic molecules and their composition

$$\text{(4b)} \quad \begin{cases} h_i.a_i = constant \\ h_i.a_i = h_j.a_j \end{cases} \quad \text{where} \quad \begin{array}{c} 1 \le i, j \le n, i \ne j \\ a_i \in \mathcal{A}_{h_i}, a_j \in \mathcal{A}_{h_j} \end{array}$$

In Figure (3c), we give an example of semantic composition for the noun phrase "smart student", obtained by composing the semantic molecules of the adjective "smart" (3a) and the noun "student" (3b). The compositional semantic constraints, $\Phi_{comp}(h, h_1, h_2)$, are given as well. As a consequence of variable bindings due to compositional constraints at the head level, some variables from the bodies of the semantic molecules (i.e. $var(b)$, $var(b_i)$) become bound as well, with $var(h) \subseteq var(b)$ for each semantic molecule.

Grammar Rules

A grammar rule $A \Rightarrow B_1, \ldots, B_n \Rightarrow^* (w, w')$ is encoded as a constraint rule, as shown below, where the nonterminals are augmented with an extra-argument, representing the semantic molecule, and two constraints are added: one for semantic composition, Φ_{comp}, and one for ontological validation, Φ_{onto}. In our extended DCG formalism both the substrings w_i and the canonical logical forms, b_i are represented as difference lists.

$$\begin{array}{l} A(w, h \bowtie b) \Rightarrow B_1(w_1, h_1 \bowtie b_1), ..., B_n(w_n, h_n \bowtie b_n)\ : \\ \qquad\qquad w = w_1...w_n,\ b = b_1, ..., b_n, \\ \qquad\qquad \Phi_{comp}(h, h_1, ..., h_n),\ \Phi_{onto}(b) \end{array}$$

The ontological constraints, Φ_{onto} present at the grammar rule level, render only the ontology responsible for the semantic interpretation, not the grammar. Φ_{onto} is based on a meta-interpreter with *freeze* (Saraswat 1989; Muresan, Potolea, & Muresan 1998) and is applied only to the body of the semantic molecule. The meta-interpreter assures that the atomic predicates, APs, (see Eq (1)), of the molecule body are not evaluated (i.e., they are postponed) till at least one variable becomes instantiated. This technique will allow a nondeterministic efficient search in the ontology. Moreover, the meta-interpreter search strategy is independent of the actual representation of the ontology, and therefore behaves as an interface to any ontology at the level of atomic predicates. The ontology-based interpretation is not done during the composition operation, but afterwards. Thus, for example, the head of the noun phrase "smart student" ((3c)) does not need to store the slot *Has_prop*, which allows us to use flat feature structures for representing the head of the semantic molecule.

From the description of both Φ_{comp} and Φ_{onto}, it can be seen that the metaphor of *semantic molecule* is essential to model and efficiently compute these two constraints associated with the grammar rules. The head of the semantic molecule is used during semantic composition, while the body is evaluated on the ontology.

In association with this constraint-based grammar, a reversible robust parser is built, based on the bottom-up active chart parsing method (Kay 1973). The parser is used both during parsing/generation of natural language expressions after the grammar and compositional constraints are learned, and during grammar and constraints learning.

In the first case, the grammar G_r and the compositional semantic constraints, Φ_{comp}, are known. For direct parsing, the string w is given. Its substrings, w_i and thus their semantic molecules, $h_i \bowtie b_i$ result through robust parsing, while the string semantic molecule $h \bowtie b$ results through Φ_{comp}. In reverse parsing (generation) the semantics of the string w is given, i.e b. By robust parsing b_i, and thus h_i and w_i result, while h is obtain by Φ_{comp} at the same time with w.

In the second case (i.e., during learning), the string w and its semantic molecule $h \bowtie b$ are given. After robust parsing the substrings w_i and thus their semantic molecules $h_i \bowtie b_i$ are obtained, while Φ_{comp} is learned based on them. If syntactic information for agreement is present, more than one positive example and also negative examples would be needed, to control the generalization/specialization process, for each Φ_{comp} learned .

Induction Algorithm

The learning algorithm for grammar induction is based on our previous work (Muresan, Muresan, & Potolea 2002) and belongs to the class of Inductive Logic Programming (ILP) methods based on Inverse Entailment (Muggleton 1995). Unlike existing relational learning methods that use randomly-selected examples and for which the class of efficiently learnable rules is limited (Cohen 1995), our algorithm learns from an ordered set of representative examples, allowing a polynomial efficiency for more complex rules. The size of this set is small and thus our algorithm is able to learn when no large annotated treebanks can be easily built. In this cases, statistical methods, even if they are able to learn from structured data (Gartner, Lloyd, & Flach 2002), needed for semantics, are of reduced applicability due to the large number of training examples required for learning.

ILP methods have the ability to use background knowledge during learning. For our task, we use background knowledge, K, that contains: 1) the previously learned grammar, 2) the previously learned semantic compositionality constraints, 3) the ontology, and 4) the lexicon that specifies for each word the specific syntactic knowledge, including its preterminal (POS) as well as the semantic information given as semantic molecule.

Algorithm 4 describes the constraint-based grammar induction.

Algorithm 4: Constraint_Grammar_Induction(E_R, E, K); where MSCR=Most Specific Constraint Rule

```
begin
    $P_{G_r} \leftarrow \emptyset$
    repeat
        $e_i \leftarrow$ Extract_Min($E_R$)
1       $MSCR(e_i) \leftarrow$ Robust_Parse($G_r, e_i, K$)
2       $A_i \Rightarrow \beta \leftarrow$ Generalize($MSCR(e_i), E, K$)
        $P_{G_r} \leftarrow P_{G_r} \cup \{A_i \Rightarrow \beta\}$
    until $E_R = \emptyset$
    return $P_{G_r}$
end
```

For each representative example $e_i \in E_R$, a cover set algorithm performs two steps: 1) the most specific constraint rule generation, $MSCR(e_i)$, and 2) the generation of the final hypothesis rule, $A_i \Rightarrow \beta$. The process continues iteratively until all the representative examples are covered.

The learning engine uses both annotated and unannotated sets of examples at different stages. First, the cover set algorithm is based only on the representative set, E_R, that is semantically annotated (pairs of strings and their semantic molecules). During the generation of the final hypothesis, weakly annotated (only chunked), and optionally unannotated examples are used for the performance criteria in choosing the best rule. We denote these positive examples as E^+=E_s. Also negative examples, E^- are used, if needed, given as weakly annotated data.

For the most specific rule generation (step 1), a reversible bottom-up active chart parser is used to derive all triples $\langle$substring, semantic molecule, grammar nonterminal$\rangle$ from the current positive example, e_i, and the current grammar, G_r, present in the background knowledge. This set of triples will allow the generation of the most specific constraint rule of the following form:

$$\begin{aligned} A_e(w, h \bowtie b) \Rightarrow B_1(w_1, h_1 \bowtie b_1), \ldots, B_n(w_n, h_n \bowtie b_n) &: \\ w = w_1 \ldots w_n,\ b = b_1, \ldots, b_n, & \\ \Phi_{comp_e}(h, h_1, \ldots, h_n),\ \Phi_{onto}(b) & \end{aligned}$$

where:

- A_e is a new nonterminal, having the syntactic category specified in the semantic representation of the positive example, e_i ($c(A_e) = h.cat$).
- B_j are nonterminals that belong to the already existing grammar given in the background knowledge, K, and that are parsed by the robust parser.
- $\Phi_{comp_e}(h, h_1, \ldots, h_n)$ is the semantic compositional constraint learned also in step 1.
- $\Phi_{onto}(b)$ is the constraint which validates the canonical logical form on the ontology.

This most specific constraint rule, $MSCR(e_i)$, added to the existing grammar covers the current representative example. $MSCR(e_i)$ has the following properties:

p1 The semantic molecule $h_j \bowtie b_j$ is generalized such that for each maximum nonterminal $B\prime_j \in chs(j)$, no negative example E^- is verified by $MSCR\prime(e_i)$.

p2 For each $h_j \bowtie b_j$ verifying $p1$, B_j is the minimum nonterminal of the chain set $chs(j)$, so that $MSCR(e_i)$ covers the same number of positive examples E^+ as does $MSCR\prime(e_i)$ at $p1$.

The most specific rule generalization (step 2) is an iterative process. A set of candidate hypotheses is generated based on: $MSCR(e_i)$, background knowledge K, and a set of heuristics. From this set, the best candidate hypothesis is chosen as the final hypothesis, $A_i \Rightarrow \beta$, using the following performance criteria: it verifies the maximum number of positive examples, E^+; it does not verify any of the negative examples, E^-; and for the same performance the most specific hypothesis is chosen. For the latter, the following priorities are considered: non-recursive rule (ordered or nonordered) ; recursive rule; or a pair of one recursive and one non-recursive rules. For the same priority, the rule with minimum head A_j is considered as the most specific. The set of heuristics is:

h1 The new hypothesis is just $MSCR(e_i)$: $A_i \Rightarrow B_1, \ldots, B_n$, where the new nonterminal $A_e = A_i$ and $\forall A_j, c(A_j) = c(Ai) : j < i$.

h2 The new hypothesis is generated from $MSCR(e_i)$ by substituting A_e with A_j: $A_j \Rightarrow B_1, \ldots, B_n$, $\forall A_j, j < i, c(A_j) = c(A_i)$.

h3 The new hypothesis is generated as a pair $A_i \Rightarrow B_j; A_i \Rightarrow B_1, \ldots, B_{j-1}, A_i, B_{j+1}, \ldots, B_n$, $\forall B_j, c(B_j) = c(A_e)$.

(smart, [cat=a, head=A, mod=N] ⋈ [A.isa=smart, N.Pn=A])
(student, [cat=n, head=N] ⋈ [N.isa=student])
(solved, [cat=v, head=V, agt=Ag, obj=Ob] ⋈ [V.isa=solve, V.agt=Ag, V.obj=Ob])
(smart # student, [cat=n, head=N] ⋈ [A.isa=smart, N.Pn=A, N.isa=student])
(the # quiz, [cat=n, head=N] ⋈ [N.det=the, N.isa=quiz])
(who # solved # the quiz, [cat=rc, head=Ag] ⋈ [Ag.isa=who, V.isa=solve, V.agt=Ag, V.obj=Ob, Ob.det=the, Ob.isa=quiz])
(the student # who solved the quiz, [cat=n,head=N] ⋈ [N.det=the, N.isa=student,
N.isa=who, V.isa=solve, V.agt=N, V.obj=Ob, Ob.det=the, Ob.isa=quiz])

(a) Representative Examples

A1(h ⋈ b) ⇒ Adj(h1 ⋈ b1) : Phi_comp(1,h,h1), Phi_onto(b)
N1(h ⋈ b) ⇒ Noun(h1 ⋈ b1) : Phi_comp(2,h,h1), Phi_onto(b)
V1(h ⋈ b) ⇒ Tv(h1 ⋈ b1) : Phi_comp(3,h,h1), Phi_onto(b)
N1(h ⋈ b) ⇒ Adj(h1 ⋈ b1), N1(h2 ⋈ b2) : Phi_comp(4,h,h1,h2), Phi_onto(b)
N2(h ⋈ b) ⇒ Det(h1 ⋈ b1), N1(h2 ⋈ b2) : Phi_comp(5,h,h1,h2), Phi_onto(b)
Rc1(h ⋈ b) ⇒ Rpro(h1 ⋈ b1), Tv(h2 ⋈ b2), N2(h3 ⋈ b3) : Phi_comp(6,h,h1,h2,h3), Phi_onto(b)
N2(h ⋈ b) ⇒ N2(h1 ⋈ b1), Rc1(h2 ⋈ b2) : Phi_comp(7,h,h1,h2), Phi_onto(b)

(b) Learned Grammar

Figure 4: Learning Example for noun phrases with determiners, adjectival modifiers and relative clauses

The final hypothesis, $A_i \Rightarrow \beta$ and the semantic constraint are added to the background knowledge, K, and the cover set algorithm continues iteratively until all the representative examples are considered. The overgeneralization can be controlled by E^-, but further probabilistic refinement can reduce the need of negative examples.

The algorithm is linear on the length of the learned hypothesis and has the complexity O($|E_R| * |\beta| * |E| * |e|^3$).

Theorem 2 (The lwfg Induction Theorem). *Given a lexicalized well-founded grammar, G, an unambiguous sublanguage $E \subseteq L(G)$, and a semantic annotated set $E_R \subseteq E_s$ of representative examples, together with the associated grammar G_r, computed with Algorithm 3, the Constraint_Grammar_Induction algorithm generates a lwfg, G'_r such that $E \subseteq L(G'_r) \cap L(G_r)$.*

Proof. We prove by induction that $L_s(G'_r) \supseteq L_s(G_r)$, given that $E^- \cap L_s(G_r) = \emptyset$ and the Assumptions 1, 2, and 3 hold true. We assume that the above property holds for the first i representative examples, $e_0, \ldots e_i$, that is $L_s(G'_i) \supseteq L_s(G_i)$, where G'_i is the learned partial grammar, and G_i is the partial grammar associated with e_i by Algorithm 3. The next representative example e_{i+1} has associated in G_r, the Min-rule $A_{i+1} \Rightarrow B_1, \ldots, B_n$ and the chain $chr(A_{i+1})$. This chain contains rules with heads greater than A_{i+1} that are still not learned, while the rules with the head B_j are already learned. If this rule is a non-recursive one, or it is not a first recursive rule for the nonterminal A_{i+1}, the robust parser can compute $MSCR(e_i)$: $A_e \Rightarrow B'_1, \ldots, B'_n$, with $B'_j \succeq B_j$ and $c(A_e) = c(A_{i+1})$ satisfying properties $p1$ and $p2$. If Min-rule is non-recursive, $MSCR$ can be generalized by heuristics $h1$ or $h2$, while if it is recursive, $MSCR$ can be generalized by the heuristic $h2$. If the above Min-rule is the first rule for the nonterminal A_{i+1} and it is recursive then : $A_{i+1} \Rightarrow B_1, \ldots, B_{j-1}, A_{i+1}, B_{j+1}, \ldots, B_n$ and G_r contains also the rule $A_{i+1} \Rightarrow B_j$. In this case the robust parser computes $MSCR(e_i)$: $A_e \Rightarrow B'_1, \ldots, B'_{j-1}, B_j, B'_{j+1}, \ldots, B'_n$, which can be generalized by the heuristics $h3$, so that the rule $A_{i+1} \Rightarrow B_j$, belonging to the chain $chr(B_j)$ is also learned. From the performance criteria and Lemma 2 it can be proved that the property of two monotonically growing grammars is preserved in the step $i+1$: $L_s(G'_{i+1}) \supseteq L_s(G_{i+1})$, and thus $L_s(G'_r) \supseteq L_s(G_r)$ holds by induction. But $L_s(G_r) \supseteq E_s$ and thus $E_s \subseteq L_s(G'_r) \cap L_s(G_r)$ and the theorem is proved as consequence. □

Proposition 2. If Assumptions 1, 2 and 3 hold and:

$$(E_R, P_{G_r}) \leftarrow Find\_Representative\_Examples(E, G)$$
$$P_{G'_r} \leftarrow Constraint\_Grammar\_Induction(E_R, E, K)$$
$$(E''_R, P_{G''_r}) \leftarrow Find\_Representative\_Examples(E, G'_r)$$

we have : $E''_R \equiv E_R$ and thus $L_s(G'_r) = L_s(G''_r)$. This means that under the given assumptions, the learned grammar preserves the representative examples, E_R. This property is useful when the acquired grammar is unknown, and the representative examples were chosen heuristically, based on linguistic knowledge. So we can verify if the sublanguage E is rich enough and if the set E_R was well chosen.

Experiments

We conducted a set of experiments to validate different aspects of the induction of our *lexicalized well-founded grammar*, as applied to a small fragment of natural language.

In a first experiment, we considered the syntactic grammar, G, given in Figure 1(a). As a result of Algorithm 3, the representative examples E_R and the grammar G_r were automatically generated (see Figure 2). We then manually annotated these examples with their semantic molecules as shown in Figure 4(a). In addition with this set, we used a rich enough set of examples E^+ (see Assumption 3) that was only weakly annotated. This set contains more complex examples which are used for the generalization process. The learned constraint-based grammar is presented in Figure 4(b). As can be seen, this grammar is equivalent to the initial grammar. But, for the rule $N1$ the nonterminal Adj

appears instead of $A1$. This is because $A1$ is a redundant nonterminal (see Assumption 1), that is $L_s(A1) = L_s(Adj)$. This is also the case for rule $Rc1$, where nonterminal Tv appears instead of $V1$. The compositional semantic constraints (Phi_comp predicates) were learned as well. For example, $Phi\_comp(4, h, h1, h2)$ corresponds to $\Phi\_comp(h, h1, h2)$ shown in Figure 3. The first argument of the predicate represents its index. At each iteration step, the learned constraint is stored in the background knowledge, given it is distinct from the already stored constraints. This experiment was important since it shows the perspective of learning a syntactico-semantic grammar when we have an appropriate syntactic grammar, G, available.

A second set of experiments was done to test the grammar induction framework when the grammar is not available. We started with a fragment of natural language that partially covers complex noun phrases (adjectival premodifiers, prepositional phrases, relative clauses), coordination, simple clauses with active/passive form of verbs, and simple wh-questions. Based on linguistic knowledge, we manually annotated fifty representative examples. In addition a set of less than three hundred weakly annotated examples and a set of less than ten negative examples were used. The set of weakly annotated examples can be partially obtained from existing syntactic treebanks, or by using the output of existing syntactic parsers on examples of interest with appropriate manual correction. We used a reduced grounded lexicon derived from Extended WordNet (Harabagiu, Miller, & Moldovan 1999), since as discussed in (Wintner 1999) the lexicon does not influence the grammar semantics. The syntactic categories covered are: adv, adj, det, pro, n, v, prep, coord, rc, and cl (for simple sentences and wh-questions). The size of the learned grammar is comparable with the size of the representative example set. For this learned grammar Proposition 2 was validated, meaning that the representative examples were appropriately chosen. Since only a small fragment of language was used, a coverage test is not applicable at this point. However, the framework shows potential for building a grammar with a small annotation effort (the size of the representative example set is small, while the weakly annotated examples can be semi-automatically derived). The learned grammar was applied to a sample text, for which the semantic representation was obtain, in a small question-answering experiment. The type of questions were “who did what to whom?”.

Conclusion and Future Work

In this paper we focused on the theoretical aspects of learning a constraint-based grammar from a small set of examples. We have introduced the concepts of *lexicalized well-founded grammar (lwfg)* G, *semantic molecule*, and *representative examples* E_R of the sublanguage $E \subseteq L(G)$, and we proved the theorem of inducing such grammars from representative examples, which constitute a small semantic treebank (Theorem 2). Thus, if a fragment of natural language, E, can be covered by a lwfg G, and the semantically annotated representative examples can be provided based on linguistic knowledge, then an equivalent lwfg G_r covering E can be induced. The grammar learning framework is currently applied to building a terminological knowledge base in the medical domain and we plan to develop the framework to allow bootstrapping of both the grammar and the ontology. Another future direction will include adding probabilities to the grammar rules, after the learning process.

References

Cohen, W. 1995. Pac-learning Recursive Logic Programs: Negative Results. *Journal of Artificial Intelligence Research* 2:541–573.

Copestake, A.; Lascarides, A.; and Flickinger, D. 2001. An Algebra for Semantic Construction in Constraint-based Grammars. In *Proceedings of the Association for Computational Linguistics, ACL-2001*.

Gartner, T.; Lloyd, J.; and Flach, P. 2002. Kernels for Structured Data. In *Proceedings of the 12th International Conference on Inductive Logic Programming, ILP2002*.

Harabagiu, S.; Miller, G.; and Moldovan, D. 1999. Wordnet2 - A Morphologically and Semantically Enhanced Resource. In *Procedings of the ACL-SIGLEX Workshop: Standardizing Lexical Resources*.

Jensen, P. A., and Nilsson, J. F. 2003. Ontology-based Semantics of Prepositions. In *Proceedings of ACL-SIGSEM Workshop: The Linguistic Dimensions of Prepositions and their Use in Computational Linguistics Formalisms and Applications*.

Joshi, A., and Schabes, Y. 1997. Tree-Adjoining Grammars. In Rozenberg, G., and Salomaa, A., eds., *Handbook of Formal Languages*, volume 3. Springer, Berlin,New York. chapter 2, 69–124.

Kay, M. 1973. The MIND System. In Rustin, R., ed., *Natural Language Processing*. Algorithmics Press, New York. 155–188.

Muggleton, S. 1995. Inverse Entailment and Progol. *New Generation Computing, Special Issue on Inductive Logic Programming* 13(3-4):245–286.

Muresan, S.; Muresan, T.; and Potolea, R. 2002. Data Flow Coherence Constraints for Pruning the Search Space in ILP Tools. *International Journal of Artificial Intelligence Tools* 11(2).

Muresan, T.; Potolea, R.; and Muresan, S. 1998. Amalgamating CCP with Prolog. *Scientific Journal of Politechnics University, Timisoara* 43(4).

Oates, T.; Armstrong, T.; Harris, J.; and Nejman, M. 2003. Leveraging Lexical Semantics to Infer Context-Free Grammars. In *Proceedings of Workshop at ECML/PKDD 2003: Learning Context-Free Grammars*.

Pereira, F. C., and Warren, D. 1980. Definite Clause Grammars for Language Analysis. *Artificial Intelligence* 13:231–278.

Pinker, S. 1989. *Learnability and Cognition: The Acquisition of Argument Structure*. MIT Press.

Saraswat, V. 1989. *Concurrent Contraint Programming Languages*. Ph.D. Dissertation, Dept. of Computer Science, Carnegie Mellon University.

Wintner, S. 1999. Compositional Semantics for Linguistic Formalisms. In *Proceedings of the Association for Computational Linguistics, ACL’99*.