Learning languages from positive examples with dependencies

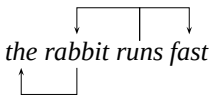
Jérôme Besombes, Jean-Yves Marion
Loria, INRIA Lorraine

Abstract

We investigate learning dependency grammar from positive data, in Gold's identification in the limit model. Examples are dependency trees. For this, we introduce reversible lexical dependency grammars which generate a significant class of languages. We have demonstrated that reversible dependency languages are learnable. We provide a $O(n^2)$-time, in the example size, algorithm. Our objective is to contribute to design and the understanding of formal process of language acquisition. For this, dependency trees play an important role because they naturally appear in every tree phrase structure.

1. An identification paradigm

From Tesniére (Tesnière, 1959) seminal study, and from ideas of Mel'čuk (Mel'čuk, 1988), we propose a two tier communication process between two speakers, see Figure 1. Jean transmits a sentence to Marie. At the first stage, Jean generates a structural sentence, like the following dependency tree



Then, Jean transforms it into a linear phrase, *the rabbit runs fast*, and send it to Marie. Now, Marie has to inverse the two tier process of Jean. For this, she has (i) to recover the structural sentence from the linear sentence (η^{-1}), (ii) to build/update/improve her grammar in order to understand the message, and to generate other messages (θ^{-1}). In the setting of natural language learning, parsers perform the first task of Marie. Roughly speaking, parsing corresponds to inverse η, that is to compute η^{-1}.

We are investigating identification, that is exact learning, of dependency tree languages. The leading idea is that data used to learn are dependency trees. Dependencies are semantic annotation by additional informations which facilitates the learning process. Such data are widely available and come from a lot of computational linguistic formal grammars such as LFG, TAG, categorial grammar and interaction grammar. The relations between those formalism are explained in the special issue of TAL (Kahane, 2000) on dependency grammars.

The data available are positive examples of a language, that is a sequence of dependency trees. Our hypothesis is that the computation of θ is reversible, that is the inputs can always be deduced from the outputs. So, identification corresponds to inverse θ. For this, we give a sufficient condition of reversibility on the grammars (G_0). We show that the class of reversible dependency tree languages is efficiently identifiable in Gold's model (Gold, 1967). That is, given a finite sequence of examples of a reversible language, we determine a reversible grammar that generates it. We refer to (Jain *et al.*, 1999) for further explanations. Our study leans on the work of Angluin (Angluin, 1982) on learning languages produced by deterministic and reversibles finite automaton. It is also closely related to Sakakibara (Sakakibara, 1992) work on reversible context free grammars and to Kanazawa (Kanazawa, 1998) work on rigid categorial grammars.

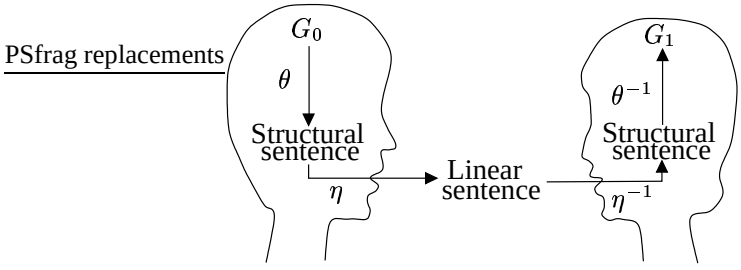


Figure 1: A two tier communication process

© 2000 . *Proceedings of the Sixth International Workshop on Tree Adjoining Grammar and Related Frameworks (TAG+6)*, pp. 25–29. Universitá di Venezia.

2. Lexical dependency grammar

Following Dikovsky and Modina (Dikovsky and Modina, 2000), we present a class of projective dependency grammars which was introduced by Hays (Hays, 1961) and Gaifman (Gaifman, 1965).

A *lexical dependency grammar* (LDG) Γ is a quadruplet $\langle \Sigma, N, P, S \rangle$, where Σ is the set of terminal symbols, N is the set of non-terminal symbols, $S \in N$ is the start symbol, and P is the set of productions. Each production is of the form $X \rightarrow U_1 \dots U_p\ a\ V_1 \dots V_q$, where $X \in N$, each U_i and V_j are in $\Sigma \cup N$. The terminal symbol a is called the *head* of the production. In other words, the head is the root of the flat tree formed by the production right handside. Actually, if we forget dependencies, we just deal with context free grammars.

Example 1. The grammar $\Gamma_0 = \langle \{a, b\}, \{S\}, P, S \rangle$ where P consists

$$S \rightarrow a\ S\ b \mid a\ b$$

Partial dependency trees are recursively defined as follows.

1. S is a partial dependency tree generated by Γ.

2. If $\dots X \dots \mathrm{b} \dots$ is a partial dependency tree generated by Γ, and if $X \rightarrow U_1 \dots U_p\ a\ V_1 \dots V_q$ is a production of Γ, then

$$\dots U_1 \dots U_p\ a\ V_1 \dots V_p \dots \mathrm{b} \dots$$

is a partial dependency tree generated by Γ.

A *dependency tree* generated by a LDG Γ is a partial dependency tree of Γ in which all nodes are terminal symbols. The language $\mathcal{D}(\Gamma)$ is the set of all dependency trees generated by Γ.

Example 2. The language generated by Γ_0 of Example 1 is

$$\mathcal{D}(\Gamma_0) = \{a\ b, a\ a\ b\ b, a\ a\ a\ b\ b\ b, \dots\}$$

Without dependencies, we recognize the context free language $\{a^n b^n / n > 0\}$.

3. Reversible LDG

A LDG grammar Γ is *reversible* if the conditions $\mathbf{R1}$, $\mathbf{R2}$ and $\mathbf{R3}$ of Figure 2 are satisfied. The class of *reversible dependency tree languages* is the class of languages generated by reversible LDG.

4. The learning algorithm

The learning algorithm works as follows. The input is a finite set H of positive examples which are dependency trees. Define $\mathtt{TG}(H)$ as the grammar constructed from all productions outputed by $\mathtt{TG}(w, S)$, for each $w \in H$. The function $\mathtt{TG}$ is described in Figure 3.

Stage 0 $G_0 = \mathtt{TG}(H)$.

Stage n+1 The grammar G_{n+1} is defined by applying one of the rule of Figure 2.

R1 If $X \rightarrow \mathbf{U}\ a\ \mathbf{V}$ and if $Y \rightarrow \mathbf{U}\ a\ \mathbf{V}$, then $X = Y$.

R2 If $X \rightarrow \boldsymbol{\alpha}\ Y \boldsymbol{\beta}\ a\ \boldsymbol{\gamma}$ and if $X \rightarrow \boldsymbol{\alpha}\ Z \boldsymbol{\beta}\ a\ \boldsymbol{\gamma}$, then $Y = Z$, where $Y, Z \in N$.

R3 If $X \rightarrow \boldsymbol{\alpha}\ a\ \boldsymbol{\beta}\ Y\ \boldsymbol{\gamma}$ and if $X \rightarrow \boldsymbol{\alpha}\ a\ \boldsymbol{\beta}\ Z\ \boldsymbol{\gamma}$, then $Y = Z$, where $Y, Z \in N$.

We write $\boldsymbol{\alpha}\ a\ \boldsymbol{\beta}$ for $\alpha_1 \ldots \alpha_p\ a\ \beta_1 \ldots \beta_q$.

Figure 2: Reversibility conditions and reduction rules.

Function $\mathtt{TG}(w, X)$
Inputs : w is a dependency tree,
X is a non-terminal symbol.
if $w \in \Sigma$
then Output $X \rightarrow w$

if $w = u_1 \ldots u_p\ a\ v_1 \ldots v_q$
then Take new $U_1, \ldots, U_p$ and $V_1, \ldots, V_q$

Output $X \rightarrow U_1 \ldots U_p\ a\ V_1 \ldots V_q$
for $i = 1$ **to** p **do** $\mathtt{TG}(u_i, U_i)$
for $i = 1$ **to** q **do** $\mathtt{TG}(v_i, V_i)$

Figure 3: $\mathtt{TG}(w, X)$ recognizes exactly w, i.e. $\mathcal{D}(\mathtt{TG}(w, X)) = \{w\}$

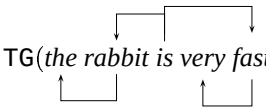
The process terminates at some stage m because the number of grammar productions decreases at each stage. So, put $\phi(H) = G_m$.

Theorem 3. *ϕ learns the class of reversible dependency tree languages.*

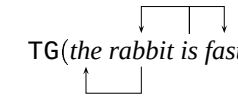
The learning algorithm is incremental and runs in quadratic time in the size of the examples. Our algorithm is implemented as a Java prototype which is accessible from `http://www.loria.fr/~ besombes`.

Example 4. $H = \{$ *the rabbit is very fast, the rabbit is fast, the rabbit is very very fast*$\}$,

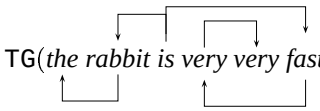
The productions of $G_0 = \mathtt{TG}(H)$ are

$\mathsf{TG}($  $, S) =$

$S \rightarrow X_1$ *is* X_2,
$X_1 \rightarrow X_3$ *rabbit*,
$X_2 \rightarrow X_4$ *fast*,
$X_3 \rightarrow$ *the*,
$X_4 \rightarrow$ *very*

$\mathsf{TG}($  $, S) =$

$S \rightarrow X_5$ *is* X_6,
$X_5 \rightarrow X_7$ *rabbit*,
$X_6 \rightarrow$ *fast*,
$X_7 \rightarrow$ *the*.

$\mathsf{TG}($  $\}, S) =$

$S \rightarrow X_8$ *is* X_9,
$X_8 \rightarrow X_{10}$ *rabbit*,
$X_9 \rightarrow X_{11}$ *fast*,
$X_{10} \rightarrow$ *the*,
$X_{11} \rightarrow$ *very* X_{12},
$X_{12} \rightarrow$ *very*

Second, we apply $\mathbf{R1}$ to identify $X_4 = X_{12}$, $X_3 = X_7 = X_{10}$.

$S \rightarrow X_1$ *is* X_2 $\quad X_3 \rightarrow$ *the* $\quad X_5 \rightarrow X_3$ *rabbit* $\quad X_8 \rightarrow X_3$ *rabbit*

$X_1 \rightarrow X_3$ *rabbit* $\quad X_4 \rightarrow$ *very* $\quad X_6 \rightarrow$ *fast* $\quad X_9 \rightarrow X_{11}$ *fast*

$X_2 \rightarrow X_4$ *fast* $\quad S \rightarrow X_5$ *is* X_6 $\quad S \rightarrow X_8$ *is* X_9 $\quad X_{11} \rightarrow$ *very* X_4

We apply $\mathbf{R1}$ to identify $X_1 = X_8$ and $X_5 = X_1$.

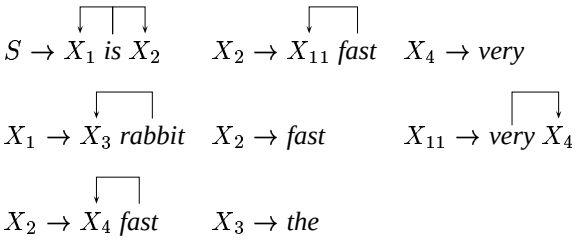
$S \rightarrow X_1$ *is* X_2 $\quad X_4 \rightarrow$ *very* $\quad S \rightarrow X_1$ *is* X_9

$X_1 \rightarrow X_3$ *rabbit* $\quad S \rightarrow X_1$ *is* X_6 $\quad X_9 \rightarrow X_{11}$ *fast*

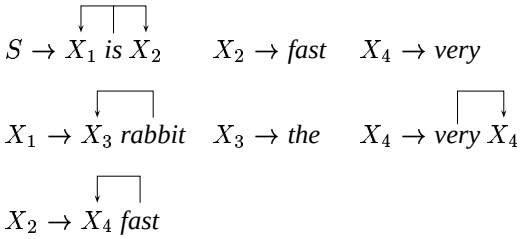
$X_2 \rightarrow X_4$ *fast* $\quad X_6 \rightarrow$ *fast* $\quad X_{11} \rightarrow$ *very* X_4

$X_3 \rightarrow$ *the*

Now, we merge $X_2 = X_6 = X_9$ by applying **R3** on S rules.



Lastly, we apply **R2** on X_2 rules by merging $X_4 = X_{11}$. We obtain the final grammar:



5. Related works

- Sakakibara (Sakakibara, 1992) gives a learning algorithm to infer reversible context free languages from skeleton parse trees. The definition of reversible grammar is very similar to ours. However, we distinguish between both productions $X \to Y\ a\ b\ Z$ and $X \to Y\ a\ b\ Z$, unlike (Sakakibara, 1992) in which they are considered identical.
- Kanazawa (Kanazawa, 1998) studies inference of several classes of categorial grammars from functor structures, based on counting the number of categories associated to a terminal symbol. It is not difficult to faithfully translate rigid grammar in reversible dependency grammars.

The defect of learning from structures is that examples usually depend on the implicit grammar that we have to guess. It appears that it is not the case in our approach because we deal with tree languages, and so is seemingly more natural.

References

Angluin, Dana. 1982. Inference of reversible languages. *Journal of the ACM*, 29:741–765.
Dikovsky, A. and L. Modina. 2000. Dependencies on the other side of the curtain. *Traitement automatique des langues*, 41(1):67–96.
Gaifman, H. 1965. Dependency systems and phrase structure systems. *Information and Control*, 8(3):304–337.
Gold, M.E. 1967. Language identification in the limit. *Information and Control*, 10:447–474.
Hays, D.G. 1961. Grouping and dependency theories. In *National symp. on machine translation*.
Jain, J., D. Osherson, J. Royer and A. Sharma. 1999. *Systems that learn*. MIT press.
Kahane, S. 2000. *Les grammaires de dépendance*, volume 41. Hermes.
Kanazawa, M. 1998. *Learnable classes of Categorial Grammars*. CSLI.
Mel'čuk, I. 1988. *Dependency Syntax: Theory and Practice*. The SUNY Press.
Sakakibara, Y. 1992. Efficient learning of context free grammars from positive structural examples. *Information and Computation*, 97:23–60.
Tesnière, L. 1959. *Eléments de syntaxe structurale*. Klincksieck.