2

Learning Dependency Languages from a Teacher

Jérôme Besombes and Jean-Yves Marion

We investigate learning dependency grammar from partial data and membership queries as a model of natural language acquisition. We define a learning paradigm based on a dialogue between the learner and a referent who knows the target language. This dialogue consists in a presentation of structured partial sentences and queries about the membership of original sentences constructed by the learner. We define an efficient algorithm corresponding to this paradigm and illustrate it on examples.

2.1 Introduction

For the definition of our model we consider several hypotheses which are largely inspired by the works of the linguist Chomsky Chomsky (1986) the psycholinguist Pinker Pinker (1994). First, the child learns the language of his parents or more specifically, the language he hears. Correct sentences are so presented to the learner, possibly partially understood and constitute the input data of the model. These data are not only linear sentences but pre-calculated structures. Indeed, semantic information or prosody are information included in the signal

The structures are commonly considered as tree in which nodes are labelled by the words of the sentence. Since the linear order of the words is conserved during the structuration process, we will consider dependency trees as a relevant model of the input data (Figure 1).

Proceedings of Formal Grammar 2004.
Paola Monachesi, Gerhard Jäger,
Gerald Penn and Shuly Winter(eds.).

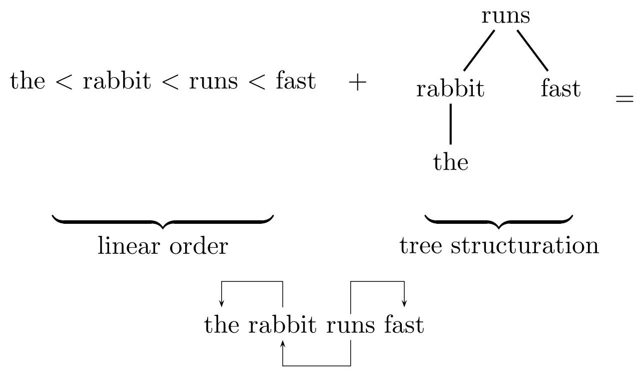

FIGURE 1 Dependency structure

Natural learning properties	Algorithmic model properties
Finite set of correct sentences are needed	Input data are a finite set of partial positive examples
Chomsky's universal grammar	Learning algorithm independent of a particular language
Structured data	Dependency tree languages
Communication with a referent	Membership queries

FIGURE 2 Correspondance between properties of the natural language aquisition and properties of the algorithmic model

Pinker underlines that the learner and the referent take part in a communication process; this communication is a key point of the learning process since it turns out that the language acquisition is not possible with no physical presence of the referent. If we suppose that a new sentence produced by a child and not understood by the referent can provide the conclusion that this sentence is not correct (doesn't belong to the language he learns), we will take into account membership queries: the algorithm submits sentences to an Oracle who replies yes or no whether they belongs to te target language or not.

The algorithm A learns a class of language if and only if, for any language L in the class, there is a finite set of partial data RS (representative sample) such that A determines L from RS with help of membership queries. Properties of A are summerized in Figure 2

Related works

Angluin first introduced the paradigm of learning with queries in Angluin (1987) for the case of regular languages and in Angluin (1988) is studied a paradim of learning from positive examples, membership queries and equivalence queries (the possibility to ask an Oracle whether a guess language corresponds to the target language or not). Obviously, for our motivation of modelling, this kind of queries are not relevant. Angluin's works have been extendend in particular by Sakakibara Sakakibara (1987b,a, 1990) for the inference of context-free grammars from structured data. The learnability of dependency languages has been studied in Besombes and Marion (2002); in this work, an algorithm for a sub-class of lexical dependency languages has been defined. As far as we know, the idea of learning from partial data and membership queries is original.

2.2 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,
- P is the set of productions.

Each production is of the form

$$X \rightarrow X_1 \ \ldots X_p \ a \ X_{p+1} \ \ldots X_q$$

or of the form

$$X \rightarrow a$$ [1]

where X and each X_i are in N and a in Σ. 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.

Given a grammar G, *partial dependency trees* t generated by a non-terminal X of G are recursively defined as follows.

- X is a partial dependency tree.

[1] This form is corresponding to the previous with $p = q = 0$.

- If $\ldots X' \ldots \mathrm{b} \ldots$ is a partial dependency tree generated by X, and if $X' \to X_1 \ldots X_p\ a\ X_{p+1} \ldots X_q$ is a production of G, then $\ldots X_1 \ldots X_p\ a\ X_{p+1} \ldots X_p \ldots \mathrm{b} \ldots$ [2] is a partial dependency tree generated by X.

We note $X \xrightarrow{*} t$ to express that t is generated by X.

A *dependency tree* generated by a non-terminal X is a partial dependency tree generated by X in which all nodes are terminal symbols. A *dependency tree* is a dependency sub-tree generated by S. The language $\mathcal{DL}(G)$ is the set of all dependency trees ($\mathcal{DL}(G) = \{d\ :$ dependency tree and $S \xrightarrow{*} d\}$).

(1) Example. Consider the grammar G defined by:

$$G = \langle \Sigma, N, P, S \rangle$$

where

- $\Sigma = \{a, b, c\}$,
- $N = \{S, X_1, X_2, X_3, X_4\}$,
- P is the following set of productions.

$$S \to X_2\ a\ X_3$$

$$\begin{array}{lll} X_2 \to X_1\ b & \qquad & X_3 \to c\ X_4 \\ X_1 \to X_2\ b & & X_3 \to c \\ X_1 \to b & & X_4 \to c\ X_3 \end{array}$$

The language $\mathcal{DL}(G)$ is the set of dependency trees

$$\{\underbrace{b\ \ldots\ b}_{n \text{ even}}\ a\ \underbrace{c \ldots\ c}_{m \text{ odd}}\}$$

[2] Dependencies can be drawn either over or under the word line for a reason of clearity.

A *subtree* of a dependency tree is inductively defined as follows:

- of $d = a$ for a terminal symbol a, then d is the only subtree of d,
- if $d = d_1 \ldots d_p \; a \; d_{p+1} \ldots d_q$ is a dependency tree then:
  - d is a *subtree* of d,
  - any subtree of d_i is a subtree of d.

If d is a dependency tree, $S(d)$ is the set of subtrees of d and if D is a set of dependency trees, $S(D)$ is the set of all subtrees of the elements of D. A *context* $c[\sharp]$ of a dependency tree d is obtained by replacing exactly one occurence of a subtree of d by a special symbol $\sharp$. In particular $\sharp$ is a context of all dependency trees. If d is a dependency tree, $C(d)$ is the set of contexts of d and if D is a set of dependency trees, $C(D)$ is the set of all contexts of the elements of D.

We will also use the notation $d = c[d']$ to express that d' is a subtree of d.

A grammar homomorphism ϕ between two grammars $G = \langle \Sigma, N, P, S \rangle$ and $G' = \langle \Sigma, N', P', S' \rangle$ is defined from a surjective mapping from N to N' which satisfies the following properties:

- $\phi(S) = S'$
- P' is the set of productions $\phi(X) \rightarrow \phi(X_1) \ldots \phi(X_p) \; a \; \phi(X_{p+1}) \ldots \phi(X_q)$

  for every production $X \rightarrow X_1 \ldots X_p \; a \; X_{p+1} \ldots X_q$ of P.

We note $G' = \phi(G)$ and in this case we have $\mathcal{DL}(G) \subseteq \mathcal{DL}(G')$.

2.3 Observation table

Following Angluin Angluin (1988), information obtained from the membership queries is stored in a table. Let $\mathcal{DL}$ be a dependency language, D a finite set of subtrees and C a finite set of contexts. The *observation table* $T = T_{\mathcal{DL}}(S(D), C)$ is the table defined by:

- rows are labelled by the subtrees of D,
- columns are labelled by elements of C,
- cells $T_{\mathcal{DL}}(d, c[\sharp])$, where $d \in S(D)$ and $c[\sharp] \in C$, are labelled with `1` and `0` in such a way that:

$$T_{\mathcal{DL}}(d, c[\sharp]) = \begin{cases} \texttt{1} \text{ if } c[d] \in \mathcal{DL} \\ \texttt{0} \text{ otherwise} \end{cases}$$

For any $d \in S(D)$, we denote by $row_T(d)$ the binary word corresponding to the reading from left to right of the row labelled by d in T.

(2) Example. Let be $\mathcal{DL} = \mathcal{DL}(G)$ the dependency language defined in Example 1, D the singleton $\{b\ b\ a\ c\ c\ c\}$ and C the set of contexts $\{\sharp, \sharp\ b\ a\ c\ c\ c,\ \sharp\ a\ c\ c\ c, b\ b\ a\ c\ c\ \sharp, b\ b\ a\ c\ \sharp, b\ b\ a\ \sharp\}$. The corresponding observation table $T = T_{\mathcal{DL}}(S(D), C)$ is the table of figure 2.

An observation table $T = T_{DL}(S(D), C)$ is *coherent* if and only if for any pair (d, d') of trees in $D \times D$, $row_T(d) = row_T(d')$. A coherent observation table $T = T_{DL}(S(D), C)$ defines a grammar G_T:

$$G_T = \langle \Sigma, N, P, S \rangle$$

where:

- Σ is set of symbols occuring in D,
- $N = \{row_T(d)\ :\ d \in S(D)\}$
- $S = row_T(d)$ for any dependency tree $d \in D$
- P is the set of productions of the form $row_T(d_1\ \ldots d_p\ a\ d_{p+1}\ \ldots d_q)$ $\rightarrow row_T(d_1)\ \ldots row_T(d_p)\ a\ row_T(d_{p+1})\ \ldots row_T(d_q)$ for all $d_1\ \ldots d_p\ a\ d_{p+1}\ \ldots d_q$ in $S(D)$.

(3) Example. The table of Example 2 is coherent and the corresponding grammar is $\phi(G)$, where G is the grammar given in Example 1 and ϕ the homomorphism defined by $\phi(S) = \texttt{100000}$, $\phi(X_1) = \texttt{010000}$, $\phi(X_2) = \texttt{001000}$, $\phi(X_3) = \texttt{000101}$, $\phi(X_4) = \texttt{000010}$.

A coherent table $T = T_{DL}(S(D), C)$ is *consistent* if and only if for every dependency trees $d = d_1\ \ldots d_p\ a\ d_{p+1}\ \ldots d_q$ and $d' = d'_1\ \ldots d'_p\ a\ d'_{p+1}\ \ldots d'_q$

	$\sharp$	$\sharp\ b\ a\ c\ c\ c$	$\sharp\ a\ c\ c\ c$	$b\ b\ a\ c\ c\ \sharp$	$b\ b\ a\ c\ \sharp$	$b\ b\ a\ \sharp$
$b\ b\ a\ c\ c\ c$	1	0	0	0	0	0
$b\ b$	0	0	1	0	0	0
b	0	1	0	0	0	0
$c\ c\ c$	0	0	0	1	0	1
$c\ c$	0	0	0	0	1	0
c	0	0	0	1	0	1

FIGURE 3 An observation table

in $S(D)$, for all i, $row_T(d_i) = row_T(d'_i)$ implies that $row_T(d) = row_T(d')$.

2.4 Representative sample

We now define the property, for a finite set of subtrees of a language, to contain the minimum information necessary to explicitely identify this language. This constitutes a minimal hypothesis to conclude in the learnability of the language. Let $\mathcal{DL}$ be a dependency language generated by a grammar G. Any finite subset RS of $S(\mathcal{DL})$ is said to be *representative* for $\mathcal{DL}$ if and only if for any transition $X \to X_1 \ldots X_p\ a\ X_{p+1} \ldots X_q$ of G, there is an element $d = d_1 \ldots d_p\ a\ d_{p+1} \ldots d_q$ in $S(RS)$ such that for all i, $X_i \xrightarrow{*} d_i$. Informally, a finite set RS is a representative sample for G if and only if each production of G has been used at least once to produce the elements of RS.

Lemma 1 *Let G be a dependency grammar, RS a representative sample for $\mathcal{DL}(G)$ and C a finite set of contexts containing $C(RS)$, if $T_{\mathcal{DL}(G)}(S(RS), C)$ is consistent then $\mathcal{DL}(G_T) = \mathcal{DL}(G)$.*

Theorem 2 *The algorithm defined in Figure 4 learns the class of dependency languages from representative samples and membership queries.*

The algorithm works as follows: it take a finite set of dependency trees as input and this set is decomposed in a finite set of subtrees and a finite set of contexts. With help of membership queries, a first observation table is constructed and the consistence is checked. If the table is not consistent, new contexts are calculated and added in the table which is then completed. The process stops as the table is consistent and a grammar is then ouput.

2.5 Examples

(4) Example. The singleton $\{b\ b\ a\ c\ c\ c\}$ is a representative sample for the dependency tree language defined in Example 1. The observation table of Figure 3 is constructed from this input with help of membership queries; this table is consistent that implies

INPUT: a finite set of dependency trees D
INITIALIZATION: $C = C(D)$; construct $T = T_{\mathcal{DL}(G)}(S(D), C)$ with help of queries
WHILE T not consistent DO

find two dependency trees $d \; = d_1 \ldots d_p \; a \; d_{p+1} \ldots d_q$ and $d' = d'_1 \ldots d'_p \; a \; d'_{p+1} \ldots d'_q$
in $S(D)$ such that forall i, $row_T(d_i) = row_T(d'_i)$ and $row_T(d) \neq row_T(d')$

add every contexts $d_1 \ldots \sharp \ldots d_p \; a \; d_{p+1} \ldots d_q$ and $d_1 \ldots d_p \; a \; d_{p+1} \ldots \sharp \ldots d_q$ in C
complete $T = T_{\mathcal{DL}(G)}(S(D), C)$ with help of queries
ENDWHILE
RETURN G_T

FIGURE 4 The learning algorithm

that the corresponding dependency grammar given in Example 3 is computed by the algorithm and the language is learnt immediately (the loop is not processed).

The following example illustrates the iterative behavior of the algorithm.

(5) Example. Let G be the following grammar:

$$S \to aX_1, aX_2, bX_2$$

$$X_1 \;\to\; dX_3, c \qquad X_3 \;\to\; e$$

$$X_2 \;\to\; dX_4 \qquad X_4 \;\to\; f$$

We have: $\mathcal{DL} = \{b \;\; c, a \;\; c, a \;\; d \;\; e, b \;\; d \;\; e, a \;\; d \;\; f\}$.

Let now consider the following representative sample:

$$RS = \{b \quad c, a \quad d \quad e, d \quad f\}$$

From it, the algorithm constructs a first table that is not consistent (Figure 5). Indeed we have:

$$row_T(e) = row_T(f)$$

but

$$row_T(d \quad e) \neq row_T(d \quad f)$$

The new context $b \quad d \quad \sharp$ is computed and added to the table that is completed with queries. The table obtained is then consistent and the process stops with the construction of de grammar $\phi(G)$, where ϕ is defined by $\phi(S) = \mathtt{10000}$, $\phi(X_1) = \mathtt{01010}$, $\phi(X_2) = \mathtt{00010}$, $\phi(X_3) = \mathtt{00101}$, $\phi(X_4) = \mathtt{00100}$

References

Angluin, D. 1987. Learning regular sets from queries and counter examples. *Information and Control* 75:87–106.

Angluin, D. 1988. Queries and concept learning. *Machine learning* 2:319–342.

Besombes, J. and J.Y. Marion. 2002. Apprentissage des langages réguliers d'arbres et applications. *Conférence d'Apprentissage, Orléans 17, 18 et 19 juin 2002* pages 55–70.

Chomsky, N. 1986. *Knowlege of Language*. Praeger, New York.

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.

Hays, D.G. 1961. Grouping and dependency theories. In *National symp. on machine translation*.

Pinker, S. 1994. *The language instinct*. Harper.

Sakakibara, Y. 1987a. Inductive inference of logic programs based on algebraic semantics. Tech. Rep. ICOT, 79.

Sakakibara, Y. 1987b. Inferring parsers of context-free languages from structural examples. Tech. Rep. ICOT, 81.

Sakakibara, Y. 1990. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science* 76:223–242.

	$\sharp$	$b\ \sharp$	$a\ d\ \sharp$	$a\ \sharp$
$b\ c$	1	0	0	0
c	0	1	0	1
$a\ d\ e$	1	0	0	0
$d\ e$	0	1	0	1
$d\ f$	0	0	0	1
e	0	0	1	0
f	0	0	1	0

$$\Downarrow$$

	$\sharp$	$b\ \sharp$	$a\ d\ \sharp$	$a\ \sharp$	$b\ d\ \sharp$
$b\ c$	1	0	0	0	0
c	0	1	0	1	0
$a\ d\ e$	1	0	0	0	0
$d\ e$	0	1	0	1	0
$d\ f$	0	0	0	1	0
e	0	0	1	0	1
f	0	0	1	0	0

FIGURE 5 The learning algorithm processing