Learning Categorial Grammars from Semantic Types[1]

Daniela Dudău-Sofronie, Isabelle Tellier, Marc Tommasi
LIFL-Grappa Team

Abstract

This paper investigates the inference of Categorial Grammars from a new perspective. To learn such grammars, Kanazawa's approach consists in providing, as input, information about the structure of derivation trees in the target grammar. But this information is hardly arguable as relevant data from a psycholinguistic point of view. We propose instead to provide information about the semantic type associated with the words used. These types are considered as general semantic knowledge and their availability is argued. A new learning algorithm from types is given and discussed.

1 Introduction

Learning a Categorial Grammar from sentences it generates consists in identifying the categories assigned to each word of its vocabulary, when the rewriting schemas (i.e. Forward Application, Backward Application, etc) are supposed to be known.

The formal learnability of AB-Categorial Grammars in Gold's model has been deeply studied in [Kan96], [Kan98]. Kanazawa has proved that the class G_k of Categorial Grammars assigning at most k different categories with any given word is learnable in Gold's model from positive Structural Examples. When $k = 1$, the grammars are called "rigid" and they can be efficiently learned. When $k > 1$, the learnability is NP-hard [C.01]. But structural examples are very informative and are difficult to justify from a psycholinguistic point of view as input of a learning algorithm. The learnability of G_k also holds from texts (i.e. positive sequences of words) but is also intractable. We believe that one cannot avoid giving more knowledge to a learning process to get practical learning algorithms.

In the meantime we should emphasize that Categorial Grammars are well known for their formalized connection with semantics [Mon74], [DWP81], providing a good compromise between formalism and linguistic expressivity [OBW88]. The idea of "semantic bootstrapping" [Bre96] (semantic information inserted to help syntax reconstruction) seems a path not yet exploited enough in formal learnability researches. Links between Kanazawa's learning strategy and semantic information have been shown in [Tel99]. This first approach is still not satisfactory as it does not avoid combinatorial explosion.

Our purpose in this article is also to learn Categorial Grammars, but from a new kind of input data, more informative than texts and more relevant than Structural Examples. These data will consist of sequences of types associated with words and can be interpreted as semantic information, allowing to distinguish facts from entities and from properties satisfied by entities.

Section 2 contains preliminaries introducing the nature of the information admitted as input to the learning process and its links with logical semantics. Section 3 illustrates our learning strategy with a detailed example. The conclusion criticizes its limitations and argues about the cognitive plausibility of the data provided to this algorithm, in comparison with the data usually used in other learning algorithms. Perspective issues are also evoked.

2 Semantic information

2.1 A typing system

We will not define here a full semantic language, as types can be associated with very different kinds of semantic representation [Mon74]. For us, the only semantic information needed will be a typing system making a basic distinction between entities and facts and allowing to express the types of functors, among which are the predicates. This typing system is a an-intensional version of Montague's intensional logic. The set Θ of possible types is defined by:

- elementary types : $e \in \Theta$ (type of entities) and $t \in \Theta$ (type of truth values) are the elementary types of Θ;
- Θ is the smallest set including every elementary type and satisfying : if $u \in \Theta$ and $v \in \Theta$ then $\langle u, v \rangle \in \Theta$ (the composed type $\langle u, v \rangle$ is the type of functors taking an argument of type u and providing a result of type v).

[1] this work was supported by the ARC INRIA project GRACQ

These types combine following rewriting rules similar with the ones for classical Categorial Grammars.

- **TF** (Type Forward) : $\langle u, v\rangle.u \rightarrow v$;
- **TB** (Type Backward) : $u.\langle u, v\rangle \rightarrow v$.

Example 1. *For example, the types associated with some words (representing respectively an entity, one-place predicates, and a predicate modifier) may be:*

- *John:* e ;
- *man, woman, runs, walks:* $\langle e, t\rangle$;
- *fast:* $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$.

The main difference between syntactic categories of a Categorial Grammar and types in Θ is that the *direction* of a functor-argument application in categories is indicated by the operator used (/ or \), whereas this direction is lost in types (as both are replaced by a ","). The functor-argument application of types is commutative, whereas it is not for categories. Note that for a given type containing n elementary types, there are 2^n possible ways to replace each of its $n-1$ "," by either / or \.

Note also that, although we consider only two elementary types, Categorial Grammars include a non bounded number of elementary syntactic categories.

2.2 From logic to types

Before describing the learning process, let us slightly deepen the links between semantics and types. As a matter of fact it seems possible, under conditions, to deduce the type corresponding with a word from its meaning representation in a Montague-like system.

The typing system we use is well adapted to the language of first order predicate logic extended with typed-lambda calculus (corresponding with a simplified un-intensional version of Montague's intensional logic, without the temporal and modal operators).

Two classical syntactic rules that contribute to the construction of the set of meaningful well-formed expressions of type $\tau \in \Theta$, ME_τ are the following:

1. λ-*abstraction*: if $\alpha \in ME_\tau$ and u is a variable of type $b \in \Theta$ then $\lambda u.\alpha \in ME_{\langle b,\tau\rangle}$;
2. *application*: if $\alpha \in ME_{\langle a,b\rangle}$ and $\beta \in ME_a$ then $\alpha(\beta) \in ME_b$.

Example 2. *The semantic translations of some words in this logical language may be:*

- *John:* $John'$ *(the prime symbol distinguishes local constants from words)*
- *man:* $\lambda x.man'(x)$*; woman:* $\lambda x.woman'(x)$
- *runs:* $\lambda x.runs'(x)$*; walks:* $\lambda x.walk'(x)$
- *a:* $\lambda P.\lambda Q.\exists x[P(x) \wedge Q(x)]$
- *fast:* $\lambda P.\lambda x.(fast'(P))(x)$

Nevertheless, the reader should note that we need to impose some syntactic conditions in order to derive types from logical formulas. For the language evoked they include the following:

- symbols denoting predicates must be used *in extension*, i.e. displaying all their arguments: for example, a two-place predicate $loves'$ will appear as $\lambda x.\lambda y.loves(x)(y)$;
- formulas associated with words must be closed, to avoid free variables and all lambdas and all quantifiers must bind variables that have at least a free occurrence in the sub-formula that follows. For example we forbid formulas like $\lambda P.\lambda y.\lambda x.\neg P(x)$ and also formulas like $\lambda x.\exists y[man'(x) \wedge runs'(x)]$ (where the variable y is not in the sub-formula that follows its binding by a lambda or a quantifier).

These two constraints refer in a certain manner to the "structure of the type"(the organization of couples of $\langle\rangle$). The number of λ symbols ahead a formula (in fact the number of application of a λ- *abstraction* rule) determinates the inner parenthesized description of the final type.

Example 3. *One* λ *symbol determines:* $\langle\square, \square\rangle$*; two* λ *symbols determines:* $\langle\square, \langle\square, \square\rangle\rangle$*, etc.*

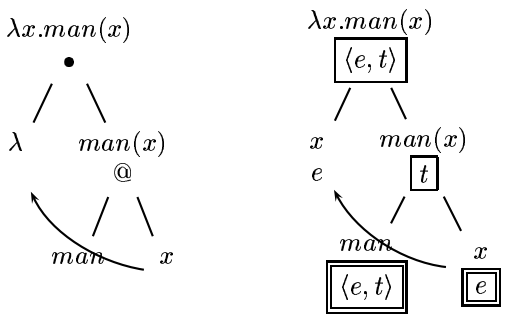

Figure 1: Type deduction example

As soon as the structure is discovered it remains to decorate it with types. Types of variables and constants are supposed to be known and thus one can label the leaves of the tree structure that represents the formula. Then the whole type of a word is evaluated by propagating contraints over types.

The first tree in the Figure 1 illustrates the decomposition of the semantic translation of the word *man* where • is a λ-*abstraction* rule and @ is a *application* rule. The type of the leaves in the double boxes are supposed to be known. We infer that $man(x)$ has the type t and the entire formula has the type $\langle e, t\rangle$.

3 A new learning algorithm from types

We adopt here *a semantic-based theory of syntax learning*, i.e. we consider that the capacity of acquiring a grammar is conditioned by the ability to build a representation of the situation described. In previous semantic-based methods of learning [HW75], [Hil83], [Fel98] word meanings are supposed to be already known when the grammatical inference mechanism starts. We make here the smoother hypothesis that the crucial information to be extracted from the environment is the semantic type of words. These types may be inferred from semantic information (as shown previously) or directly extracted from the environment.

The input of our learning algorithm is a sample of couples composed of a syntactically correct sentence and a corresponding sequence of types. The purpose is to build a rigid Categorial Grammar able to generate this sample. The missing information is the direction of the functor/argument applications, as each type of the form $\langle u, v\rangle$ can combine with a type u placed either on its left (with a **TB** rule), or on its right (with a **TF** rule). The identification of the operators / and \ will be processed in two steps: **a parsing step** and a step to **get categories from types**.

Parsing types We assume that every sentence given as input is syntactically correct and thus that the associated sequence of types can be reduced using the TB and TF rules to the type of truth values **t**. The first step of the algorithm is to find such a reduction.

Example 4.

a	*man*	*runs*
$\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$	$\langle e, t\rangle$	$\langle e, t\rangle$

This reduction can be viewed as a parse in a context free grammar. **TF** and **TB** are interpreted as schemes of rules instantiated by types in the input sequence. As every instantiated rule fitting one of these schemes is in Chomsky Normal Form, it is possible to adapt a standard parsing algorithm to find the parse on types. In this paper, we modify the Cocke-Younger-Kasami (CYK) algorithm. The whole process is the search for replacing the "," in type expressions to get categories. Possible values for "," are / , \ or "left unchanged" (a "," is left unchanged when the corresponding type is never in a functor place). To this aim, we identify every "," with a distinct variable. A reduction via **TF** or **TB** implies some equalities between categories and sub-categories that must be propagated.

Hence, type reductions translate into variable equalities and a given parse leads to a system of equalities.

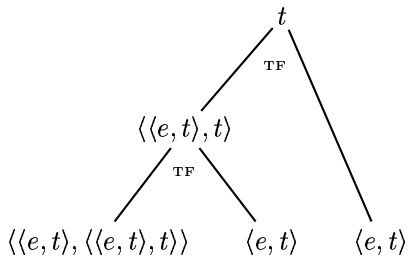

Figure 2: A parse tree for types

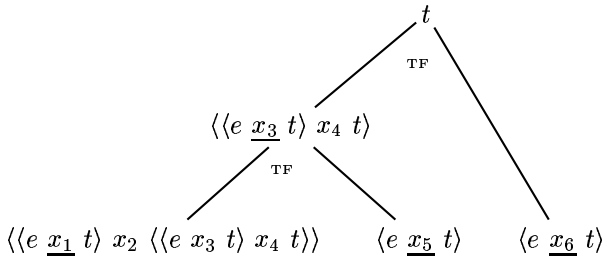

Figure 3: A parse tree underling equality between variables

Note that there could be more than one parse for a given input. The outcome of the parser is thus a set of parses each of which being described by a set of variable equalities.

Getting categories Now given such constraints on types, we easily deduce categories and hence a Categorial Grammar that accepts the sequence of words as input. To this aim, we consider that **TB** and **TF** have the classical rules in the categorial grammar formalism as a counterpart. The association between types and categories is done in the following way:

- **t** is associated with S.
- any distinct type that never occurs at a functor place in the parse is associated with a new variable category.
- If **TF**: $\langle ux_iv\rangle.u \rightarrow v$ and u (resp. v) is associated with the category A (resp. B) then $x_i = /$ and $\langle u\ x_i\ v\rangle$ is associated with the category B/A.
- If **TB**: $u.\langle ux_iv\rangle \rightarrow v$ and u (resp. v) is associated with the category A (resp. B) then $x_i = \backslash$ and $\langle u\ x_i\ v\rangle$ is associated with the category $A\backslash B$.

Thus, while getting categories, we add equality constraints of the form $x_i = /$ or $x_i = \backslash$ for some i . The process is always guaranteed to give a set of categories because of the properties of the sequences in input.

Learning process An on-line and incremental learning algorithm can be designed using the strategy describe above. It consists in inferring equality constraints between variables and equality constraints fixing the value of variables from each input couple (in the example below, both are inferred at the same time). We also take into account that the target Categorial Grammar is rigid, that is every word is associated with at most one category.

Example 5. *A first input is provided ...*

a	*man*	*runs*
$\langle\langle e\ x_1\ t\rangle x_2\langle\langle e\ x_3\ t\rangle x_4 t\rangle\rangle$	$\langle e\ x_5\ t\rangle$	$\langle e\ x_6\ t\rangle$

$x_1 = x_5$
$x_2 = /$
$x_3 = x_6$
$x_4 = /$

At this point, the Categorial Grammar inferred consists in the associations: a: (S/B)/A ; man: A; runs: B.

Then the second input is provided and the set of variable equalities is enriched. Since every word has at most one category we consider the same variables in the type of words already encountered in a previous sentence.

a	$woman$	$walks$
$\langle\langle e\ x_1\ t\rangle x_2\langle\langle e\ x_3\ t\rangle x_4 t\rangle\rangle$	$\langle e\ x_7\ t\rangle$	$\langle e\ x_8\ t\rangle$

$$x_1 = x_7 = x_5$$
$$x_2 = /$$
$$x_3 = x_8 = x_6$$
$$x_4 = /$$

Consider a third input. The set of variable equalities becomes:

a	$woman$	$walks$	$fast$
$\langle\langle e\ x_1\ t\rangle x_2\langle\langle e\ x_3\ t\rangle x_4 t\rangle\rangle$	$\langle e x_7 t\rangle$	$\langle e x_8 t\rangle$	$\langle\langle e\ x_9\ t\rangle x_{10}\langle e\ x_{11}\ t\rangle\rangle$

$$x_1 = x_7 = x_5$$
$$x_2 = /$$
$$x_3 = x_8 = x_6 = x_{11} = x_9$$
$$x_4 = /$$
$$x_{10} = \backslash$$

Finally, if the last sentence is "John walks", equality constraints propagate x_8 to several categories.

$John$	$walks$
e	$\langle e\ x_8\ t\rangle$

$\ldots\ x_3 = x_8 = x_6 = x_{11} = x_9 = \backslash\ \ldots$

To summarize, after renaming e in T and $\langle e x_1 t\rangle$ in CN, the Categorial Grammar obtained consists in $John : T$; $man, woman : CN$; $runs, walks : T\backslash S$; $a : (S/(T\backslash S))/CN$; $fast : (T\backslash S)\backslash(S\backslash T)$. This grammar is able to recognize sentences like "John runs fast" and some generalization has thus occurred. In more complicated cases where more than one parse is possible on a given input, the algorithm must maintain a set of sets of constraints representing a set of candidate grammars.

4 Conclusion

We will discuss here the complexity of our approach and after that we will try to evidence our point in comparison with other works. In the end we will furnish the future directions of our research.

A parse in a context free grammar in Chomsky Normal Form can be computed in polynomial time in the size of the input. This complexity result is stated for a given context free grammar. In our setting, we do not have a context free grammar but a "set of possible ones" defined by schemes of rules. This changes a lot the complexity issues as illustrated by the following example:

Example 6. *Let us consider the following input:*

$a \ldots a$	b	$a \ldots a$
$e \ldots e$	$\langle e, \langle e, \ldots, \langle e, t\rangle\rangle$	$e \ldots e$

One can parse such a string in a context free grammar in polynomial time with respect to the size of the input sequence. But there exist C_n^{2n} different context free grammars compatible with this input. As a matter of fact the type associated with b expects $2n$ arguments among which n are on its right and n are on its left. Since our algorithm builds all possible grammars, it will run in exponential time with respect to n. Nonetheless, note that this n does not depend on the length of the input. It is related to the lexicon and the size of types associated with words.

A first idea to circumvent this combinatorial explosion is to try to build only one grammar among the set of possible ones. But something has to be done to get an incremental learning procedure and handle mind changes that may arise with new inputs. A second way of research is to find an economic and efficient manner to handle the representation of a set of grammars compatible with a given set of input sentences.

Learning a Categorial Grammar means associating categories with words. Categories are built from basic categories and operators. Learning categories from strings of words seems impossible in reasonable time. So, richer input data need to be provided.

The approach developed so far by Buskowsky & Penn and Kanazawa consisted in providing Structural Examples, i.e. giving the nature of the operators / and \ and letting the basic categories to be learnt. Our approach is dual, as it consists in providing the nature of the basic categories (under the form of types) and letting the operators to be learnt. But, we

must emphasize that the mapping between categories and types is not perfect. For example, there exist some words like common nouns and intransitive verbs that initially receive the same type $\langle e, t \rangle$ corresponding with a one place predicate and that won't receive the same syntactic category at the end of the learning process because they have different combinatorial properties in English.

The algorithm we propose is able to identify in the limit any rigid Categorial Grammar. It still needs to be extended to k-valued Categorial Grammars. In fact, two sources of polymorphism should be distinguished: the case where a word must be associated with different semantic types (for example words like "and"), and the case where a word must be associated with different syntactic categories corresponding with the same semantic type (which must be the case for words like "a"). The first case should be handled as a natural extension of our learning strategy, where each semantically distinct instance of the word is treated as a new word, but the second case should be harder to deal with. Another extension to be explored is the case when only some of the types associated with words are known (for example, those associated with lexical words), whereas some others (those associated with grammatical words) remain unknown.

References

[Bre96] Michael R. Brent. *Computational Approaches to Language Acquisition.* MIT Press, 1996.

[C.01] Costa Florncio C. Consistent identification in the limit of any of the classes k-valued is np-hard. In *Logical Aspects of Computational Linguistics*, volume 2099 of *Lecture Notes in Artificial Intelligence*, pages 125–134. Springer Verlag, 2001.

[DWP81] D. R. Dowty, R. E. Wall, and S. Peters. *Introduction to Montague Semantics.* Linguistics and Philosophy. Reidel, 1981.

[Fel98] J. A. Feldman. Real language learning. In *ICGI'98, 4th International Colloquium in Grammatical Inference*, pages 114–125, 1998.

[Hil83] J. C. Hill. A computational model of language acquisition in the two-year-old. *Cognition and Brain Theory*, 3(6):287–317, 1983.

[HW75] H. Hamburger and K. Wexler. A mathematical theory of learning transformational grammar. *Journal of Mathematical Psychology*, 12:137–177, 1975.

[Kan96] Makoto Kanazawa. Identification in the limit of categorial grammars. *Journal of Logic, Language, and Information*, 5(2):115–155, 1996.

[Kan98] M. Kanazawa. *Learnable Classes of Categorial Grammars.* The European Association for Logic, Language and Information. CLSI Publications, 1998.

[Mon74] R. Montague. *Formal Philosophy; Selected papers of Richard Montague.* Yale University Press, 1974.

[OBW88] Richard T. Oehrle, Emmon Bach, and Deirdre Wheeler, editors. *Categorial Grammars and Natural Language Structures.* D. Reidel Publishing Company, Dordrecht, 1988.

[Tel99] I. Tellier. Towards a semantic-based theory of language learning. In *12th Amsterdam Colloquium*, pages 217–222, 1999.