A tool for language learning based on categorial grammars and semantic information

Daniela Dudau Sofronie, Isabelle Tellier, Marc Tommasi
LIFL-Grappa, Université Charles de Gaulle-lille3
59653 Villeneuve d'Ascq Cedex, FRANCE

E-mail : dudau@lifl.fr, tellier@univ-lille3.fr, tommasi@univ-lille3.fr
`http://www.grappa.univ-lille3.fr`

1 Introduction

Natural language learning still remains an open problem, although there exist many approaches issued by actual researches. The large majority of approaches are based on statistical methods (see [Rot00]) but there exists also symbolic methods as theory refinement [Her00]. We also address ourselves this challenge of language learning and we provide here a prototype of a tool.

As language learning can take place at different levels: phonetic, syntactic, semantic, etc. we need to clarify that we center here on the syntactic level. We intend to find a (set of) grammar(s) that recognizes new correct sentences (in the sense of the correct order of the words) by means of some initial correct examples that are presented and of a strategy to deduce the corresponding grammar(s) consistent(s) with the examples at each step. In this model, languages are supported by grammars, so, the process of learning is, in essence, a process of grammatical inference.

Usually, in NLP approaches, natural language is represented by lexicalized grammars because the power of the language consists in the information provided by the words and their combination schemas. That's why we adopt here the formal model of a categorial grammar. A categorial grammar assigns every word a category and furnishes some general combination schema of categories.

But, in our model, the strings of words are not sufficient to infer a categorial grammar, so additional information is needed. Kanazawa's work [Kan98] is the most recent and complete reference on grammatical inference of categorial grammars. The additional information he furnishes is the internal structure of each sentence as a Structural Example. We try to provide instead an additional information more lexicalized, of semantic nature: the semantic type of words. Its provenance, as well as the psycho-linguistic motivation can be found in [DSTT01b] and [DSTT01a].

2 The data and the learning algorithm

The prototype proposed, conceived in Java, is built in order to be a test instrument of our formal hypothesis concerning the learning of the syntax helped by semantic types. In order to be more precise concerning the notions used, we give here the formal definitions of classical categorial grammars[1] and of semantic types.

Definition 1. *Let Σ be a finite alphabet and C be a finite set of elementary categories (where S is a distinguished category). C' is the closure of C under the operators "/" and "\", meaning that C' is the smallest set that verifies: (a) $C \subseteq C'$;(b) if $A \in C'$ and $B \in C'$ then $A/B \in C'$ and $B\backslash A \in C'$.*

A categorial grammar *G on Σ is every finite relation between Σ et C' ($G \subset \Sigma \times C'$ and G finite). In other words, every word is associated with a corresponding set of categories.*

[1]There exists different classes of categorial grammars depending on the set of combination schemas used: classical categorial grammars, combinatory grammars, Lambek grammars.

Definition 2. *A* classical categorial grammar *is a categorial grammar in which the only admissible combination schemas (reduction rules) for all A and B in C' are: (a) FA: A/B . $B \rightarrow A$ (forward application); (b) BA: B . $B\backslash A \rightarrow A$ (backward application).*

The language recognized by a classical categorial grammar G is the set of sentences (finite concatenations of words from the alphabet) for which there exists a category assignment that can be reduced to the distinguished category S.

We consider a simple *semantic typing system* (see [Mon74] for details), making distinction between *entities* and *facts* and allowing to express the types of *functors*, among which are the *predicates*. The set Θ of every type 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).

Between the semantic types and the categories there exists a similarity in construction, but the main difference consists in loosing the direction of application in types expressions, whereas this direction is explicit indicated by the operators (/ and \) in categories.

Example 1. *The categories and the semantic types corresponding with some words of a vocabulary are:*

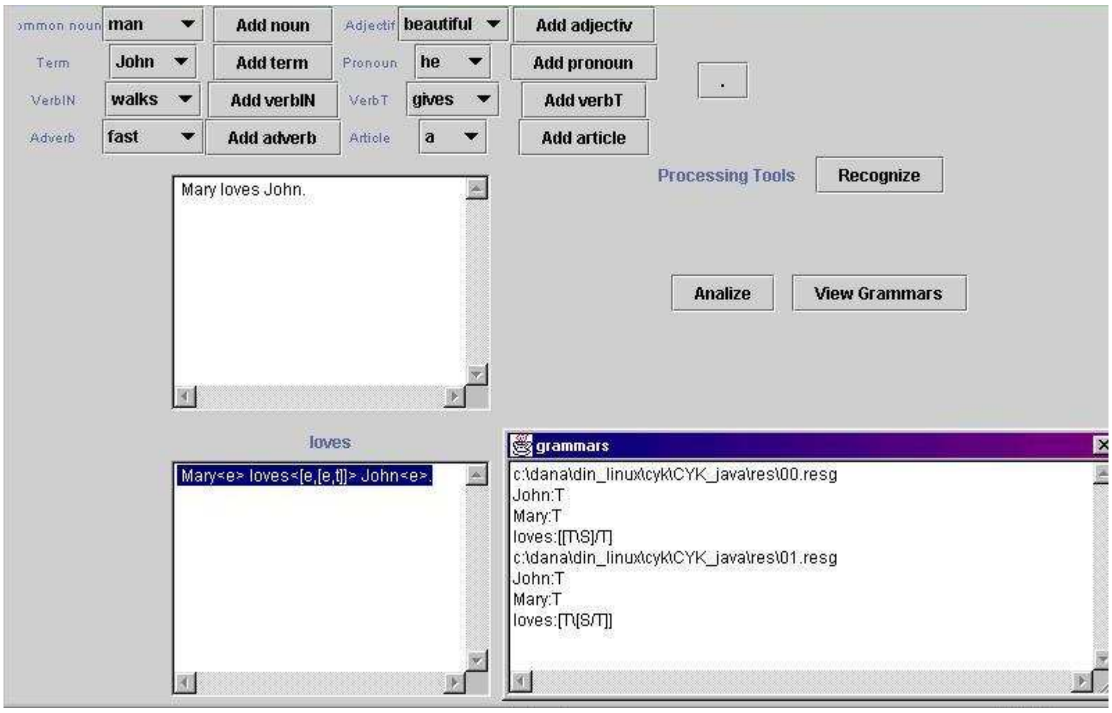
- *John: category - T; type - e*
- *man, woman: category - CN; type - $\langle e, t\rangle$*
- *runs, walks: category - $T\backslash S$; type - $\langle e, t\rangle$*
- *fast: category - $(T\backslash S)\backslash(T\backslash S)$; type - $\langle\langle e, t\rangle, \langle e, t\rangle\rangle$*
- *a: category - $(S/(T\backslash S))/CN$; type - $\langle\langle e, t\rangle, \langle\langle e, t\rangle, t\rangle\rangle$*

The significance of types is not so hard to guess: a proper noun must be an "entity"; a common noun is an "one-place predicate" over entities returning a truth value; an intransitive verb is also an "one-place predicate"; an adverb is an "one-place predicate modifier" etc.

The semantic types are considered as previously acquired knowledge, so they are integrated in the sample offered as input as labels of the words. Assuming that a sample of typed sentences is offered to our tool, it will build a set of categorial grammars compatibles with the sample. The categorial grammars we work on have the propriety that the relation between categories and types is an injective homomorphism.

The engine of the whole machinery represents a parse algorithm in the CYK-style [HU79] that works on the sequence of types attached to each sentence and deduces in the first step all possible parses and some constraints over types. The second step consists in a renaming phase to obtain a set of grammars, by applying the constraints on the expressions of types and thus inferring categories. So, for each sentence we have the set of grammars compatible with it. The entire process is incremental, in the sense that the constraints will be propagated from one sentence in the sample to another. As soon as the processing is finished for a sample the (set of) grammar(s) inferred will recognize all the sentences in the sample.

Example 2. *Let's consider a simple input sample formed by two typed sentences: a⟨[[e,t],[[e,t],t]] ⟩ man⟨[e,t]⟩ walks⟨[e,t] ⟩. a⟨[[e,t],[[e,t],t]]⟩ woman⟨[e,t]⟩ runs⟨[e,t]⟩ fast⟨[[e,t],[e,t]]⟩. The output is one grammar compatible with the input sample: a=[[S/A0]/A1], walks=A0, fast=[A2\A0], woman=A1, runs=A2, man=A1. A generalization occurs here, for example, this output grammar recognizes also the sentence: "a woman walks."*



```
Mary loves John.
```

```
Mary<e> loves<[e,[e,t]]> John<e>.
```

```
c:\dana\din_linux\cyk\CYK_java\res\00.resg
John:T
Mary:T
loves:[[T\S]/T]
c:\dana\din_linux\cyk\CYK_java\res\01.resg
John:T
Mary:T
loves:[T\[S/T]]
```

At this moment, we can consider that a training phase was accomplished, so we can test the inferred grammars to see if they recognize a new built sample with the same words as in the training sample. Until now, we didn't consider the case when new words can appear in the test sample, but this is an issue of the future work.

The graphical interface of our tool (see figure above) proposes two different manners of constructing the input sample of training:

- a user-sensitive and interactive interface. The user has the possibility to construct himself different sentences with words provided by a vocabulary. The vocabulary is stocked in a file that can be changed at any time, the system restarting with the new added words. When a new word is used in the construction its type is automatically added, if it is a single-type word. If not, a list of possible types appears and the user has to make a choice. More or less the user must possess a certain knowledge about the semantics of the words (an a-priori semantic acquisition is supposed). As soon as a sentences will be finished by a "." the typed sample is ready.
- an already built sample to be chosen from a file.

The sample is analyzed with "Analyze" command and the results can be visualized with "View Grammars" that puts the output grammars in files, one grammar in each file and offers the user the possibility to visualize them. Another functionality of this tool concerns recognizing. With the grammar(s) obtained in the previous operation we can check whether a test sample is or not recognized. The test sample is charged from a test file, or is built as previously with the words in the same vocabulary.

3 Conclusion

The tool presented is a prototype, it has not been tested yet on large corpora, but the functionalities that it provides are very flexible. So we are allowed to do a lot of variations in the vocabulary, by adding or deleting words, also by choosing the grammar and the test sample. Concerning the actual work to extend our tool, that needs to be able to process large amounts of data, we provide a module that is able to transform a simple corpora labeled in XML format with part-of-speech tags to a typed-corpora. A simple XML parser will be used and all part-of-speech tags will be replaced by type tags, using a simple table of correspondences. For example: a part-of-speech "intransitive verb" has the type label "[e,t]". These kind of labelled data are easier to obtain than structural examples needed by Kanazawa, and thus justify our approach.

References

[DSTT01a] D.Dudau-Sofronie, I. Tellier, and M. Tommasi. From logic to grammars via types. In *Proceedings of the 3rd Learning Language in Logic Workshop*, pages 35–46, 2001.

[DSTT01b] D.Dudau-Sofronie, I. Tellier, and M. Tommasi. Learning categorial grammars from semantic types. In *Proceedings of the 13th Amsterdam Colloquium*, pages 79–84. ILLC/Department of Philosophy, 2001.

[Her00] Déjean Hervé. Allis: a symbolic learning system for natural language learning. In *Proceedings of the 4th Conference on Computational Natural Language Learning and of the 2nd Learning Language in Logic Workshop*, pages 95–98, Lisbon, Portugal, September 2000. ACL.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[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.

[Rot00] Dan Roth. Learning in natural language: Theory and algorithmic approches. In *Proceedings of the 4th Conference on Computational Natural Language Learning and of the 2nd Learning Language in Logic Workshop*, pages 1–6, Lisbon, Portugal, September 2000. ACL.