Electronic Notes in Theoretical Computer Science 5 (2001) *to appear*

Stochastic Lambek Categorial Grammars

Guillaume Bonfante [1] Philippe de Groote [2]

LORIA BP 239
Campus Scientifique
Vandœuvre-lès-Nancy, France

Abstract

We present here a stochastic point of vue on Lambek Categorial Grammars which have shown their utility in Natural Language Processing. The key point comes from the fact that the parsing in the Lambek Calculus is based on an essentially non-deterministic algorithm. We claim that a stochasting treatment may be a good way to resolve that issue.

1 Introduction

Statistical methods have turned out to be quite successful in natural language processing. During the recent years, several models of stochastic grammars have been proposed, including models based on lexicalised context-free grammars [3], tree adjoining grammars [16], dependency grammars [2,5], or categorial AB grammars [13].

In this exploratory paper, we propose a new model of stochastic grammar, whose originality derives from being based on Lambek categorial grammars [8]. This model presents interesting properties:

- Probabilities are attached to syntactic dependencies and not to derivation rules. Moreover, they are expressed at the level of the lexicon. As a consequence, our model is fully lexicalised.
- A treatment of lexical ambiguities is provided. This treatment is based on unresolved dependencies. Consequently, long distance dependencies are taken into account.
- The probabilities attached to the lexical entries are not used to approximate correct parsings by highly probable parsings, but just act as guidelines

[1] Email: bonfante@loria.fr
[2] Email: degroote@loria.fr

during the categorial deduction. Consequently, the rigourous discipline of categorial type logics is not lost.

2 Lambek calculus and proofnets

We assume that the reader is familiar with Lambek categorial grammars as presented, for instance, in [1,9,10,11], and focus on the relation existing between the Lambek calculus [8] and Girard linear logic [6].

Remember that the formulas (or types) of the Lambek calculus are built upon a set of atomic types $\mathcal{A}$ by means of the connectives $\setminus$ (direct implication, or left division), / (retro-implication, or right division), $\bullet$ (conjunction, or product), and that the deduction relation is specified by an intuitionistic sequent calculus without any structural rules.

It is possible to translate the formulas of the Lambek calculus into formulas of multiplicative linear logic in such a way that a sequent of the Lambek calculus is valid if and only if its translation is a valid sequent of Girard-Yetter cyclic linear logic [17].

The formulas of multiplicative linear logic obey the following grammar:

$$\mathcal{F} \ ::= \ \mathcal{A} \mid \mathcal{A}^{\perp} \mid \mathcal{F} \otimes \mathcal{F} \mid \mathcal{F} \parr \mathcal{F}$$

where the unary connective $^{\perp}$ denotes the linear negation, and the binary connectives $\otimes$ (*tensor*) and $\parr$ (*par*) correspond to multiplicative conjunction and disjunction respectively.

The deduction relation of cyclic linear logic is specified by means of a classical one-sided sequent calculus, and the translation $[\![\cdot]\!]$ of a Lambek sequent into such a classical sequent is defined as follows:

$$[\![A_0, \ldots, A_n \vdash A]\!] \ = \ \vdash [\![A_0]\!]^{-}, \ldots, [\![A_n]\!]^{-}, [\![A]\!]^{+}$$

where:

(i) $[\![a]\!]^{-} = a^{\perp}$, for a an atomic type,
(ii) $[\![A \bullet B]\!]^{-} = [\![A]\!]^{-} \parr [\![B]\!]^{-}$
(iii) $[\![A \setminus B]\!]^{-} = [\![A]\!]^{+} \otimes [\![B]\!]^{-}$
(iv) $[\![A/B]\!]^{-} = [\![A]\!]^{-} \otimes [\![B]\!]^{+}$
(v) $[\![a]\!]^{+} = a$, for a an atomic type
(vi) $[\![A \bullet B]\!]^{+} = [\![B]\!]^{+} \otimes [\![A]\!]^{+}$
(vii) $[\![A \setminus B]\!]^{+} = [\![B]\!]^{+} \parr [\![A]\!]^{-}$
(viii) $[\![A/B]\!]^{+} = [\![B]\!]^{-} \parr [\![A]\!]^{+}$

The above translation allows proof-nets for the Lambek calculus to be defined [7,14,15].

Let $A_0, \ldots, A_n \vdash A$ be a sequent of the Lambek calculus. The proof-frame

corresponding to this sequent is defined to be the sequence of formula trees $[\![A_0]\!]^-, \ldots, [\![A_n]\!]^-, [\![A]\!]^+$.

Given a proof-frame, an axiom link is an edge between two leaves of the proof-frame such that the literals decorating these leaves are dual of each other (i.e., A and $A^\perp$). The two leaves connected by such an axiom link are called the conclusions of the axiom link. A proof-frame being a sequence of trees (rather than a multiset), one distinguishes between the left and the right conclusion of an axiom link.

A proof-structure consist of a proof-frame and a set of axiom links such that:

(i) each leaf of the proof-frame is the conclusion of exactly one axiom link;

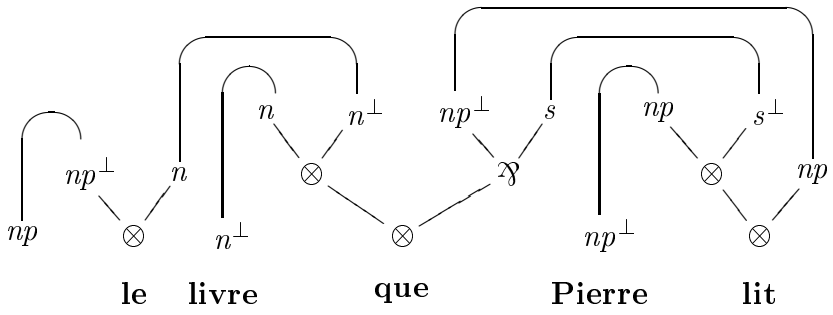
(ii) the axiom links do not cross; more precisely, for any two axiom links A and B, with left and right conclusions A_l, A_r and B_l, B_r respectively, it is not the case that $A_l < B_l < A_r < B_r$, where $<$ is the order in which the literals occur in the yield of the proof-frame.

It is not the case that any proof-structure may be interpreted as a proof of the sequent corresponding to the proof-frame. A proof-structure that corresponds to an actual proof is called a proof-net. There are several correctness conditions, in the literature, that allow the proof-nets to be distinguished from the other proof-structures. Girard's long trip condition [6] and Danos-Regnier criterion [4] are such correctness conditions.

Proof-nets may be considered as the categorial equivalent of parse trees. For instance, according to the following type assignment:

le: sn/n, **lit**: $(np \setminus s)/np$, **livre**: n, **Pierre**: np, **que**: $(n \setminus n)/(s/np)$,

parsing the french noun phrase *le livre que Pierre lit (the book that Pierre reads)* yields the following proof-net:



3 Stochastic Categorial Grammars

Consider a lexical entry ω whose assigned type is A/B. The rough intuition behind such a type assignment is that ω is a word that takes a constituent of type B on its right in order to form a constituent of type A. It does not mean, however, that in any grammatical sentence involving ω there is necessarily

a constituent of type B on the right side of ω. At the proof-net level, it means that an atomic type B occurring in a type A/B is not necessarily the left conclusion of an axiom link. As a counterexample, consider the proof-net given in the previous section. While the word *lit* is assigned the type $(np \setminus s)/np$, the second occurrence of np in this type is the right conclusion of an axiom link.

The quite simple idea behind the stochastic model we propose is to assign to each leaf of a proof-frame the probability that this leaf be the left conclusion of an axiom link. Clearly this probability does not depend only of the literal that is attached to the leaf, but on the whole context. In order to obtain a lexicalized model, we make the simplifying hypothesis that the probability assigned to a leaf depends only on the word whose type contains this leaf. In spite of its simplicity, this hypothesis is sufficient to allow interesting word order constraints to be expressed. For instance, the verb which serves as a head for the relative clause introduced by a relative pronoun such as *que* occurs necessarily after the pronoun. On the other hand, the noun or the noun phrase that is modified by the relative occurs necessarily before the pronoun. Consequently, in the type assigned to *que*, which is $(n \setminus n)/(s/np)$, the first occurrence of n will be assigned probability 0 and s and np will be assigned probability 1.

4 Operational interpretation of the parameters

As observed by Morrill [12], the structure of the proof-nets suggests a left-to-right incremental parsing procedure. This procedure may be implemented by a non-deterministic pushdown automata that takes a proof-frame as input, and scans the yield of this proof-frame from left to right. Two moves are possible:

(i) push the current literal and read the next literal from the yield;

(ii) erase the top of the stack provided it is the dual of the current literal, and read the next literal from the yield.

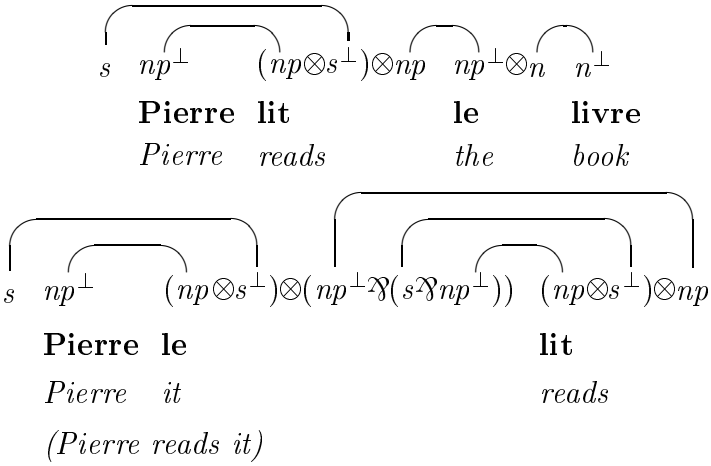
The second type of move, which amounts to creating an axiom link, is submitted to a correctness condition, akin to Danos-Regnier, that ensures that the possible proof-structures resulting from machine runs are indeed proof-nets.

At the begining of the computation, one give to the automata a stack containing a unique element $[s]$, that is the expected type of the sentence. The computation success if one branch ends with an empty stack.

Now, consider some machine run that successfully produces a proof-net. It is easy to see that this resulting proof-net is such that a leaf of its proof-frame is the left conclusion of an axiom link if and only if the corresponding literal was pushed on the stack during the machine run. Consequently, the probability we attach to a literal may be interpreted as the probability that a move of the first type acting on this literal leads to a successful parsing.

5 Taking into account lexical ambiguity

It is often assumed, in the literature, that a parser takes a tagged sentence as input and, consequently, that lexical ambiguity is handled by a part of speech tagger. We do not make this assumption and incorporate a stochastic treatment of lexical ambiguity within our model. Consider the two following sentences in which the french word *le* occurs with two different types. In the first sentence, it is used as a determiner, and is assigned the type np/n. In the second sentence it is used as a clitic pronoun, and is assigned the type $(np \setminus s)/((np \setminus s)/np)$.



Now consider the content of the stack of the automaton described in Section 4, just before scanning the word *le*. In the first case, the stack contains $[np]$ while, in the second case, it contains $[s\ np^{\perp}]$. This information, which corresponds to the unresolved dependencies, may be used to predict the type that must be chosen for *le*. Note that this proposal allows long distance dependencies to be taken into account, which is not the case with taggers based on n-grams or on hidden Markov models.

We are now in a position to give a formal definition of our model of stochastic categorial grammar. Let $\mathcal{A}$ be a set of atomic types. The set $WT(\mathcal{A})$ of weighed types is inductively defined as follows:

(i) $(A, \rho) \in WT(\mathcal{A})$ whenever $A \in \mathcal{A}$ and $\rho \in [0, 1]$;

(ii) $A \bullet B, A \setminus B, A/B \in WT(\mathcal{A})$ whenever $A, B \in WT(\mathcal{A})$.

A stochastic Lambek grammar is then defined to be 5-tuple $\langle \mathcal{A}, \Sigma, \mathcal{L}, (\alpha_\sigma)_{\sigma\in\Sigma}, S\rangle$ where:

(i) $\mathcal{A}$ is a finite set of atomic types;

(ii) Σ is a finite set of terminal symbols.

(iii) $\mathcal{L} : \Sigma \to 2^{WT(\mathcal{A})}$ is a function that assigns to each terminal symbol a non-empty finite set of weighed types.

(iv) $(\alpha_\sigma)_{\sigma\in\Sigma}$ is an indexed family of functions $\alpha_\sigma : (\mathcal{A} \cup \mathcal{A}^{\perp}) \times \mathcal{L}(\sigma) \to [0, 1]$

such that for any $\sigma \in \Sigma$ and any $A \in (\mathcal{A} \cup \mathcal{A}^{\perp})$

$$\sum_{T \in \mathcal{L}(\sigma)} \alpha_\sigma(A, T) = 1.$$

The interpretation of these probability functions is as follows: $\alpha_\sigma(A, T)$ gives the probability that σ has type T when the top of the stack is A.

(v) $S \in \mathcal{A}$ is the distinguished type of the grammar.

6 Conclusions

This paper reports first steps towards the definition of stochastic categorial grammars. The simple model we have presented is based on the associative Lambek calculus, which is non commutative. As well known (see for instance [10]), there are several phenomena (crossed dependencies, medial extraction,...) that cannot be accomodated in a purely non commutative setting. On might adapt the model in order to allow for partial commutativity. The idea is to provide the abstract machine described in Section 4 with new types of moves. For instance, one could define a move that would consist in poping a non-top element of the stack. Several variations are possible, the tuning of which requires real size experimentation. To this end, we are currently developping parsing procedures and parameter estimation techniques. These algorithms run in polynomial time provided that the depth of the stack of the abstract machine is bounded, which is a sound hypothesis after Morrill's analysis of acceptability [12].

References

[1] B. Carpenter. *Type-Logical Semantics*. MIT Press, Cambridge, Massachussetts and London England, 1997.

[2] M. J. Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annual Meeting of the ACL and the 8th Conference of the EACL*, 1996.

[3] M. J. Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the ACL*, 1997.

[4] V. Danos and L. Regnier. The structure of multiplicatives. *Archive for Mathematical Logic*, 28:181–203, 1989.

[5] J. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistic*, pages pp. 340–345, 1996.

[6] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[7] F. Lamarche and C. Retoré. Proof nets for the Lambek calculus. In M. Abrusci and C. Casadio, editors, *Proofs and Linguistic Categories, Proceedings 1996 Roma Workshop*. Cooperativa Libraria Universitaria Editrice Bologna, 1996.

[8] J. Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65:154–170, 1958.

[9] M. Moortgat. *Categorial Investigations: logical and linguistic aspects of the Lambek calculus*. Foris Publications, 1988.

[10] M. Moortgat. Categorial type logic. In J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, chapter 2. Elsevier, 1997.

[11] G. Morrill. *Type Logical Grammar: Categorial Logic of Signs*. Kluwer Academic Publishers, Dordrecht, 1994.

[12] G. Morrill. Incremental processing and acceptability. *Computational Linguistics*, 26(3):319–338, 2000.

[13] Miles Osborne and Ted Briscoe. Learning stochastic categorial grammars. In T. Mark Ellison, editor, *CoNLL97: Computational Natural Language Learning*, pages 80–87. Association for Computational Linguistics, Somerset, New Jersey, 1998.

[14] C. Retoré. Calcul de Lambek et logique linéaire. *Traitement Automatique des Langues*, 37(2):39–70, 1997.

[15] D. Roorda. *Resource Logics: proof-theoretical investigations*. PhD thesis, University of Amsterdam, 1991.

[16] Y. Schabes. Stochastic lexicalised tree-adjoining grammars. In *Proceedings of the 14th International Conference on Computational Linguistic*, pages pp. 426–432, 1992.

[17] D. N. Yetter. Quantales and (non-commutative) linear logic. *Journal of Symbolic Logic*, 55:41–64, 1990.