PARSING STRATEGY FOR SPOKEN LANGUAGE INTERFACES WITH A LEXICALIZED TREE GRAMMAR

Ariane Halber and David Roussel†*

Thomson-CSF
Corporate Research Laboratory
F-91404 Orsay Cedex, France
E-mail: {ariane, roussel}@thomson-lcr.fr

Abstract

Our work addresses the integration of speech recognition and natural language processing for spoken dialogue systems.

To deal with recognition errors, we investigate two repairing strategies integrated in a parsing based on a Lexicalized Tree Grammar. The first strategy takes its root in the recognition hypothesis, the other in the linguistic expectations.

We expose a formal framework to express the grammar, to describe the repairing strategies and to foresee further strategies.

1. Introduction

The coupling of speech recognition (ASR) and natural language processing (NLP) grows an unavoidable part of spoken dialogue systems. On the one hand, ASR modules can benefit from linguistic information, ([1] among others). On the other hand, the NLP modules can benefit from an extended output from the recognizer –since a single best hypothesis may not be accurate. To uncover the correct utterance from the recognition output, [2] and [3] parse the N-best hypotheses, [4] parses a word graph. We propose a compromize between exhaustivity and heavy computation with an integrated approach that anchors the parsing in the best hypothesis and calls needed repairing taking advantage of the concurrent recognition hypothesis.

In order to repair an ill-recognized sentence, a parser should present selective capacities on the concurrent acoustic hypotheses. We have based our work on a repairing strategy which exploits the complementarity between the information taken from N-best hypotheses and the predictive power of a lexicalized tree grammar.[1] In this paper, we first present the principles of the Lexicalized Tree Furcating Grammar (LTFG) which leads our parsing strategy. Second, we expose a granular formalization. Then, the repairing strategies are described autonomously in this framework. Finally, three symbolic/stochastic hybrid strategies relying on the same framework are reviewed.

2. LTFG basic principles

LTFG, like LTAG (Lexicalized Tree Adjoining Grammar), belongs to the family of Lexicalized Tree Grammar. It captures word dependencies thanks to a set of syntactic structures –called elementary trees– defined in its lexicon and factors recursion thanks to a subset of trees –called auxiliary trees– that can come and insert into any host tree structure. "Lexicalized" means that every elementary tree presents at least one lexical anchor on its frontier. In LTFG, trees can be combined through substitution and furcation. The substitution operation places a tree at an expected syntactic position. The furcation operation puts an auxiliary tree inside a host tree. As compared to TAG adjunction, it is a simplification: it does not introduce systematically a governing level between the auxiliary tree and the host node, furthermore the auxiliary tree frontier falls all on one side of the host subtree[2].

Figure 1 displays the result of several auxiliary tree furcations on a single node. The semantic structure displayed figure 1 is built synchronously to the parsing. This semantic representation serves as a substrate for higher level processings. Each syntactic operation activate a specific rule to assemble the target node features with the source node features. To eliminate early wrong syntactic analyzis, the nodes features are checked to be compatible for any syntactic operation [8]. It allows namely to reject spurious successive furcations on a single node, however distant. The node features are updated through the assembling process so that compatibility between all auxiliary trees is ensured while building the semantic representation.

---

* with Dept. Signal, ENST Paris

† with Lab. CLIPS IMAG, Grenoble

[1] In [5], we present examples of the two repairing strategies, and how to inform the parsing process with lexical information from the N-best hypotheses.

[2] Furcation borrows from [6] furcation operation. Assuming furcation, the synchronous insertion of elements on either side of a node cannot be captured while this phenomenon is captured in TAG through so-called "wrapping" auxiliary trees [7]. This grants TAG an extended context sensitivity which allows it to express a language $\{a^n b^m c^n d^m\}$, while TFG can only express $\{a^n b^m c^n d^p\}$. At the cost of this little loss in context sensitivity, TFG gains an $o(n^3)$ parsing instead of $o(n^6)$ [7].

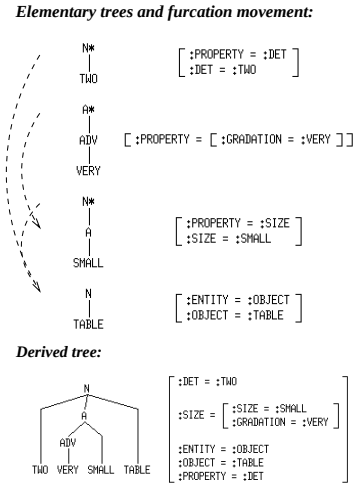

Figure 1: Example of furcation. A star symbol on the right of a dominating node indicates an auxiliary tree that will anchor (on its right) on a compatible node.

3. Formalization

TFG can be formalized in a granular way in Linear Indexed Grammar (LIG) formalism as far as syntactic operations are concerned. This is a means to both compare formally TFG with different grammars and to host different parsing strategies. Moreover, the LIG compilation is appropriate to bear a stochastic version of the grammar. Finally having a unified representation for regular parsing and stochastic parsing will be helpful to design collaboration schemes between the two in a modular and capitalizing way.

We adapt the LIG compilation proposed by [9] for TAG. The aim is to compile a tree grammar into a grammar based on rewriting string of non terminal and terminal symbols[3]. It comes down to individualizing every node of every elementary tree and to describing transformation rules, which follow a path from node to node. Actually, each node is split into a top and a bottom node. The nodes can be rewritten according to five types of rules gathered below. [10] shows how to obtain a stochastic version of Tree Grammar based on its LIG compilation, it consists in granting a probability p to each rule, with $p = 1$ for rules which express a simple path in an elementary tree (here type (1) rules).

Types 1-2: Simple path in the tree –moving from a node η to its dominated nodes $\eta_1 ... \eta_n$

$$b[\eta] \rightarrow t[\eta_1]t[\eta_2]...t[\eta_n] \quad (1)$$

In the right member of the equation, there might be lexical anchors together with the $t[\eta_i]$ (then the rule is called "terminal"). The above domination equation applies after the moving from the top to the bottom of a node, which acts as a no–more–furcation rule since furcation only applies on $t[\eta]$:

$$t[\eta] \rightarrow b[\eta] \quad (2)$$

Types 3-4: Furcation –moving from a node η that receives a furcation to the auxiliary tree root node η_r.

For a right-auxiliary tree:

$$t[\eta] \rightarrow t[\eta_r]t[\eta] \quad (3)$$

For a left-auxiliary tree:

$$t[\eta] \rightarrow t[\eta]t[\eta_r] \quad (4)$$

The subtrees dominated by the two nodes are ran over to their frontiers, the frontiers fit into place, given definition of rules (3) and (4).

The furcation rules can be seen as type (1) domination rules above the host node and the auxiliary root node. Here successive furcation on a single node is allowed; the disallowing of further furcation on η, after the furcation of a given auxiliary tree for instance, is realized by alternate rules:

$$t[\eta] \rightarrow t[\eta_r]b[\eta] \quad (3\text{bis})$$
$$t[\eta] \rightarrow b[\eta]t[\eta_r] \quad (4\text{bis})$$

Type (5): Substitution –moving from a substitution site η to the substituted root node η_r:

$$t[\eta] \rightarrow t[\eta_r] \quad (5)$$

4. Repairing Strategies

The contribution of LTFG for recognition errors repairing stems from two properties : validation and prediction. Validation enables a bottom–up repairing strategy, while prediction enables a top–down repairing strategy. The repairing articulates around a bottom–up parsing. The parsing process decomposes into a lexical analyzis phase which generates possibly concurrent sequences of trees, and a parsing phase which processes the given sequence of trees in parallel branches of parsing. From LIG point of view the lexical analyzis works as an activating of terminal rules according to each lexical instance w:

$$b[\eta] \rightarrow t[\eta_1]..w..t[\eta_n] \quad (1a)$$

Besides, when a rule is activated its left member node η is made to refer to the tree α to which it belongs, then triggering each node "left rules", i.e. rules in which the considered node is the left member.

$$\eta \overset{refers}{\rightarrow} \alpha \overset{triggers}{\rightarrow} \mathcal{R}_l(\alpha) \quad (6)$$

$$\mathcal{R}_l(\alpha) = \bigcup_{\eta_i \in \alpha} (pos_{\eta_i}[\eta_i] \rightarrow pos_{\eta_k}[\eta_k]..pos_{\eta_l}[\eta_l]) \quad (7)$$

where $pos_{\eta_i} \in \{t; b\}$. Let $\mathcal{R}_0$ be the whole set of left rules triggered:

$$\mathcal{R}_0 = \bigcup_{i \in [1..m]} \mathcal{R}_l(\alpha_i) \quad (8)$$

The search space $\mathcal{R}$ used by the parsing is a subset of $\mathcal{R}$. Are being considered only rules which allow at

[3] It should be noted that contrarily to TAG compilation, TFG compilation doesn't use any stack propagation in the transformation rules. This is due to its restriction in auxiliary trees types assuming furcation.

least one step of parsing, in the sense that all right member nodes match at least one left rule compatible with the trees order. This assures in particular that only nodes from referred trees are involved. It is noted:

$$\mathcal{R} = \bar{\mathcal{R}}_0 \tag{9}$$

The repairing strategies consist in modifying the search space $\mathcal{R}$.

The bottom–up repairing puts new elementary trees into the lexical analyzis back–end according to new lexical candidates. The lexical candidates are taken from the recognizer output, they are selected either from acoustic observation [5] or from linguistic prediction on word dependencies (LTFG lexicon predicts namely missing anchors for multi–anchored trees). The concurrent recognition candidates are factored under one or several "over-specified" joker trees. The latter are created on-line to share the syntactic tree structure of their subsumed instances, and to cumulate a factorization of their semantic features. From LIG point of view, a joker sees its nodes inherit the rewriting rules involving corresponding nodes of subsumed instances, at the only difference, for the terminal rule, that an interrogation mark stands for the lexical anchor:

$$b[\eta] \rightarrow t[\eta_1]..t[\eta_i] \ ?\ t[\eta_{i+2}]..t[\eta_n] \tag{1a}$$

The terminal rule (1a) is incorporated in the lexical analyzis result, the search space is triggered according to equations (6) to (9).

The top–down repairing infers from the state of parsing when failing, whether an under-specified elementary tree can be inserted in order to resume the combining process. Those special elementary trees are either defined in the lexicon or generated on-line, they present minimal semantic information. From LIG point of view, rules with nodes from the joker tree are introduced to push the rule reduction further, they induce a new search space $\mathcal{R}'$, derived from the cumulation of left rules sets.

$$\mathcal{R}'_0 = \mathcal{R}_0 \cup \mathcal{R}'_l \tag{8bis}$$

$$\mathcal{R}' = \bar{\mathcal{R}}'_0 \tag{9}$$

5. Evaluation

We have tested the parsing strategy on errors produced by two existing ASR systems from SRI and Cambridge University. The former, Nuance Communication recognizer system is constrained by a Context Free Grammar. Linguistic anomalies detection (ie: the top-down strategy) is normally needless. The second system, Abbot, uses an n-gram model (backed off trigram model)[4]. The application domain is taken from the COVEN project [11]

[4]The training corpus for the trigram was generated artificially by the context free grammar of the first recognizer mentioned. 15% of the testset is slightly out of the Nuance CFG.

[4]The joker tree inserted by the top-down repairing generalizes a large number of elementary trees in the lexicon. The N-best hypotheses intersect at most a small subset or none.

PARSING USED	% relevant	% irrelevant	average nb of hyps with joker → wholly lexicalized
NUANCE recognition output			
standard	32 %	17 %	1.02 → 1.02
bottom–up repairing	47 %	4 %	1.15 → 3.68
ABBOT recognition output			
standard	62 %	6.5 %	1.02 → 1.02
bottom–up repairing	14 %	6.5%	1.02 → 2.25
top–down repairing	11%	0%	36 (lexicon)[4] 1 → 2.1 (N-best)

Table 1: results of repairing strategy

The parser has been tested on a 200 words application keeping at most seven N-best hypotheses from the recognition output. The robust parsing runs in real time on an SGI Indigo-2 Impact (R4400 250 MHZ). First results on the repairing capacities are presented in table 1.

The top-down repairing is always used relevantly with our experiment data, but higher level have generally to consider the different instances that are generalized by underspecified jokers –introduced by this pass. To be exhaustive, the hypotheses number amounts to 36 in average, so that some sorting is needed.

The bottom-up repairing relevance relies on the available acoustic candidates. When the right word is present in none of the N-best hypotheses, the parsing may produce irrelevant analyzis. Increasing dramatically the number of N-best, to make sure to get the right word among them, would go along with an increase of parsing computation, and possibly an increase of produced analysis hypotheses. The dialogue module cannot afford to handle too many concurrent analyzis, even if they are factored into an overspecified joker. The dialogue runs the risk of confusing the user with irrelevant interactions. Here again an improved sorting of lexicalized hypotheses would be desirable.

6. Toward Hybrid Strategies

Our experiment, as well as other experiments like [12], suggests to cross-check the parsing strategy with other knowledge sources, like statistical cues derived from text corpora or from recognition errors corpora. An algorithm to build a stochastic LTAG (SLTAG) is proposed by [10] based on a LIG compilation similar to that proposed for LTFG in section 3. That way, there are three possible collaboration modalities between symbolic and statistic approaches. The first is at the fore–end of the parsing, the second is embedded in the core of the parsing and the third deals with the back–end of the parsing.

First, lexical analysis works as a blind process while its result conditions a great part of the parsing complexity. [13] investigates several statistic models to help lexical analysis task in case of Lexicalized Tree Grammar –which he calls "Supertagging". The most promising approach turns out to be the one based on syntactic links probabilities. This is exactly the probabilities that scored LIG rules would capture in a stochastic version of LTFG.

In a more embedded way, statistic can complement the constraints expressed in LTFG. For one thing, since errors recognition and homophony often disrupt agreement rules, those constraints have to be captured in a non exclusive mode. As a possible solution, a mixing of symbolic and stochastic parsing is offered by LIG framework. Moreover statistical regularities between words can be captured from a corpus on syntactical grounds instead of simple adjacency [14]. For example, cooccurences of modifier and modified trees can be tuned on a corpus. It should be helpful especially to filter out ill-recognized sentence that may falsely appear well formed.

Finally, a stochastic model allows to rank multiple analysis hypotheses provided by the parser. A probability estimate of a parsing is given straightforwardly [15] given the path of LIG rules (it is equivalent to the path of tree operations). A rule is put at the front of the path to express the initiating probability of the top rule (i.e. the initiating probability of the dominating tree). The parsing probability is estimated by the product of individual rules probabilities[5]. No further search or decoding is needed to get this probability as it stems directly from the rules path, which is already at hand[6]. This probability takes the lexical frontier into account so that hypotheses spanning different sentences can be compared. This is especially useful for identifying the most probable lexical instance of an analysis which contains a joker element.

7. Conclusion

To integrate ASR and NLP we proposed a parser which is able, in case of word errors, to process concurrent candidates stemming from the recognizer in a factored way or to directly predict underspecified candidates. This linguistic component is independent from a given recognition system. While defined to output a high level representation for the dialogue module, it make the recognition decoding progress to some extent. To exploit this feature further, it seems promising, thanks to a Linear Indexed Grammar compilation, to use collaboratively our symbolic repairing strategy –based on a Lexicalized Tree Grammar– and stochastic approaches –based on a stochastic version of the grammar.

[5] With more computation, the likelihood of the sequence of words, independently from a given parsing, can also be provided by an Inside probability as defined in [10].

[6] Pure stochastic parsing with a Tree Grammar has been proved NP–hard by [16].

References

[1] D. Goddeau and V. Zue. Integrating probabilistic LR-parsing into speech understanding systems. In *ICASSP'92, vol. 1*, pages 181–184. IEEE, 1992.

[2] J. Dowding, Moore R., Andry F., Gawron J.M., and Moran D. Combining linguistic and statistical knowledge sources in natural-language processing for ATIS. In *ARPA Spoken Language Technology Workshop*, 1995.

[3] Hirschman et al. Integrating syntax and semantics into spoken language understanding. In *SNLW'91*, pages 366–371, 1991.

[4] G. Hanrieder and G. Görz. Robust parsing of spoken dialogue using contextual knowledge and recognition probabilities. In *ESCA Tutorial and Research Workshop on Spoken Dialogue Systems*, Denmark, 1995.

[5] D. Roussel and A. Halber. Filtering errors and repairing linguistic anomalies for spoken dialogue systems. In *ACL/EACL Workshop on Spoken Dialogue Systems*, 1997.

[6] K de Smedt and G. Kempen. Segment grammar : a formalism for incremental generation. In C. Paris et al., editor, *Natural language generation and computational linguistics*. Dodrecht, Kluwer., 1990.

[7] Y. Schabes and R. Waters. Tree insertion grammar: a cubic-time parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4):480–515, 1995.

[8] D. Roussel. A lexicalized tree grammar with morphological component for spoken language processing *: in french*. In *Colloque Représentation et Outils pour les Bases Lexicales*, Grenoble, 1996.

[9] Y. Schabes and S. Shieber. An alternative conception of tree-adjoining derivation. *Computational Linguistics*, 20(1):91–124, 1994.

[10] Y. Schabes. Stochastic Lexicalized Tree-Adjoining Grammars. In *COLING*, 1992.

[11] V. Normand and J. Tromp. Collaborative Virtual Environments : the COVEN project. In *FIVE'96*, http://chinon.thomson-csf.fr/coven/, 1996.

[12] J. Bear, J. Dowding, and E. Shriberg. Integrating multiple knowledge sources for detection and correction of repairs in human-computer dialog. In *ACL'92*, pages 56–63, Newark,DE, 1992.

[13] A Joshi and B. Srinivas. Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing. In *(COLING)*, 1994.

[14] A. Halber. Capturing long distance dependencies from parsed corpora. Technical report, ATR, ITL Dept, Kyoto, December 1994.

[15] P. Resnick. Probabilistic Tree-Adjoining Grammar as a framework for statistical natural language processing. In *COLING*, 1992.

[16] K. Sima'an. Computational complexity of probabilistic disambiguation by means of tree-grammars. In *COLING*, 1996.