Memoisation in Sentence Generation with Lexicalised Grammars

Nicolas Nicolov
Dept of Artificial Intelligence
Univ. of Edinburgh
Edinburgh EH1 1HN, UK
nicolas@dai.ed.ac.uk

Abstract

This paper discusses a sentence generation system PROTECTOR which uses: (i) a non-hierarchical semantic representation which allows for flexible lexical choice and uniform treatment of different languages, (ii) a lexicalised D-Tree Grammar which is very similar to Tree-Adjoining Grammar in spirit, and (iii) dynamic programming techniques to avoid doing redundant computations. We review the motivation for choosing such an organisation of the generation system and give an example of the generation of a sentence which involves a lexical gap. The generation of the example sentence requires a non-deterministic mode of computation (the lexical gap forcing backtracking). We show how dynamic programming techniques can be used to save re-generating structures using a top-down generation algorithm.

Keywords: natural language generation, non-hierarchical semantics, lexicalised d-tree grammars, dynamic programming.

1 Introduction

Natural language generation is the process of generating text from a set of abstract communicative goals. It attempts to model the human language production mechanisms in man-machine communication. As part of the overall generation process computer systems will need to consider how the communicative goals can be mapped onto conceptual representations and these in turn into sentences in a natural language. The latter process is known as sentence generation and this paper discusses a system for doing this task (realising sentences from meaning representations).

2 Conceptual input

Early work on sentence generation assumed input of the form: $\texttt{pred}(\texttt{arg}_1, \ldots \texttt{arg}_n)$ and the generation process was reduced to mapping $\texttt{pred} \rightarrow$ *verb*, $\texttt{arg}_1 \rightarrow$ first *complement*, etc. This approach, of course, makes the "semantic structures" be nothing more than disguised syntactic representations and reduces the sentence generation problem to finding out the ordering of the constituents. The tree-like semantic assumption does not allow for handling head switching examples (Nicolov, 1993), incorporation of modifiers in the syntactic head (*French blond* and *blond French girl* cannot be generated from `french(blond(girl))`) and cases like: *She smiled a welcome to the guests./ She welcomed the guests with a smile.*

Such phenomena can be addressed more elegantly using non-hierarchical semantic representations. In PROTECTOR conceptual graphs are used (Sowa, 1992). The same generation mechanisms can be used with underspecified discourse representation structures.

3 D-Tree Grammars

D-Tree Grammar (Rambow et al., 1995) is a grammar formalism which arises from work on Tree-Adjoining Grammars (TAG) (Joshi, 1987).[1] In the context of generation, TAGs have been used in a number of systems MUMBLE (McDonald and Pustejovsky, 1985), SPOKESMAN (Meteer, 1990), WIP (Wahlster et al., 1991), synchronous TAGs (Shieber and Schabes, 1991) the system reported by McCoy (McCoy et al., 1992), the first version of PROTECTOR (Nicolov et al., 1995), and SPUD (Stone and Doran, 1997). TAGs have been given a prominent place in the VERBMOBIL project — they have been chosen to be the framework for the generation module (Caspari and Schmid, 1994; Harbusch et al., 1994; Becker et al., 1998). In the area of grammar development TAG has been the basis of one of the largest grammars developed for English (Doran et al., 1994).

[1] DTG and TAG are very similar, yet they are not equivalent (Weir pc).

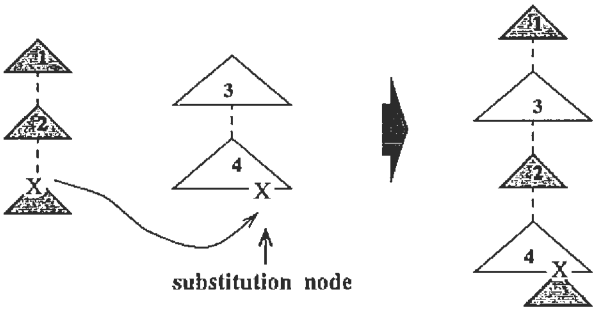

Figure 1: Subsertion

DTGs uses two operations to combine elementary structures — subsertion (Figure 1) and sister adjunction (Figure 2). The elementary structures are d-trees (descriptions of trees) which in addition to immediate dominance relation allow for stating dominance relationships between nodes in the d-tree.

Unlike TAGs, DTGs provide a uniform treatment of complementation and modification at the syntactic level. DTGs are seen as attractive for generation because a close match between semantic and syntactic operations leads to simplifications in the overall generation architecture. DTGs try to overcome the problems associated with TAGs while remaining faithful to what is seen as the key advantages of TAGs (Joshi, 1987):

1. the extended domain of locality over which syntactic dependencies are stated; and
2. function argument structure is captured within a single initial construction in the grammar.

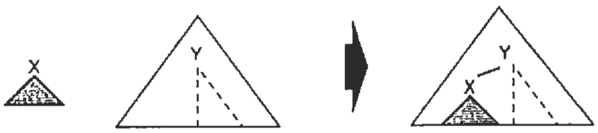

Figure 2: Sister adjunction

We use a lexicalised (every elementary structure contains a terminal node (anchor) which 'justifies' the construction), feature-based (nonterminals are feature structures) DTG.

4 Generation strategy

PROTECTOR uses declarative specification of the relation between semantics and syntax encoded as mapping rules. The mapping rules are elementary d-trees (i.e., tree descriptions) annotated with applicability semantics a match with which will licence the applicability of the mapping rule. In addition if the d-tree has nonterminal leaf nodes relevant parts of the applicability semantics are related to these nodes so that we know how the semantics is decomposed. PROTECTOR employs a top-down (recursive descent) strategy for generating the complements once an initial top-level mapping rule has been chosen (this stage is called generation of skeletal structure). PROTECTOR keeps track how much of the input semantics it has consumed. Then in a consequent stage the remaining semantics is consumed which involves the use of modification and sister-adjunction.

5 Example

In this section we discusss the generation of a sentence which involves a lexical gap:

> **Alexander attacked the town 'full-scalely'.*
> *Alexander launched a full-scale attack on the town.*

The input semantics and the search space are are shown in Figure 3 (see next page). At the onset of generation there are at least two top-level mapping rules that can be chosen (*attack* and *launch an attack*) and the default one (*attack*) leads to a dead end. The reason is the lack of a mapping rule (not only in the linguistic knowledge base of the generator but worse of all in the English language) that would allow us to express the concept FULL-SCALE as a structure that we can intergrate to the existing skeletal syntactic structure (*Alexander attacked the town*). Such is the nature of lexical gaps and this forces backtracking. The generator would need to reconsider its previous decisions, it would have to undo (forget) about all the structures it had built all the way up to the point when it chose the wrong mapping rule. This was the first choice that was made so practically every computation is lost. All the work that went into building the subject and object NPs has to be duplicated. Choosing the alternative (*launch an attack*) mapping rule

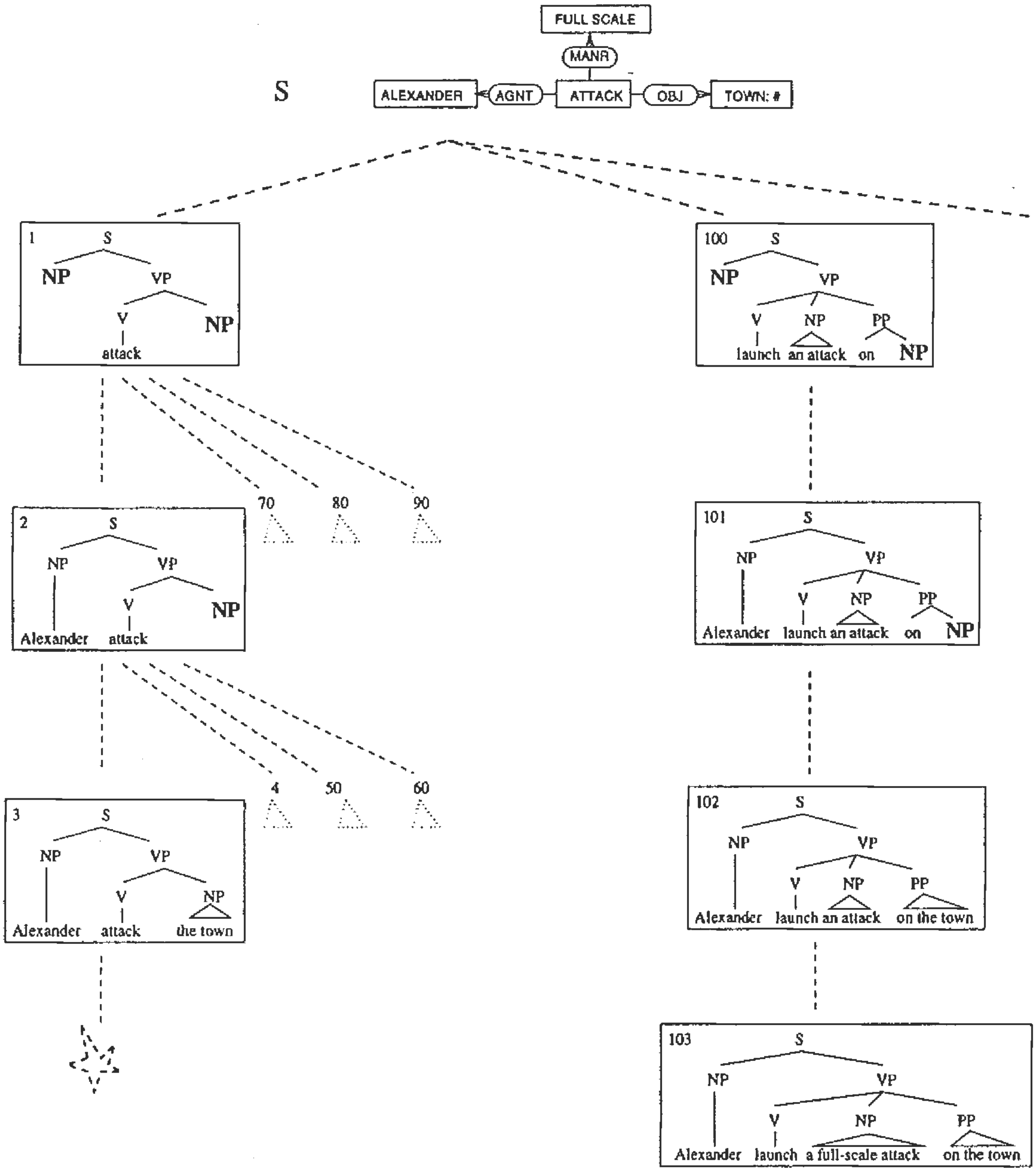

Figure 3: The search space for the example

and generating its required complements will result in re-computation of the subject and object NPs. These NPs can be arbitrarily large and in order to avoid doing redundant computations we store the results of previous generation goals and reuse them if needed again. Such dynamic programming techniques have been exploited heavily in parsing and PROTECTOR's declarative mapping rules and flexibility of incorporating alternative generation strategies allows us to take advangates of that work. This approach is gaining popularity in generation (Shieber, 1988; Haruno et al., 1993; Pianesi, 1993; Gerdemann and Hinrichs, 1995; Kay, 1996; Nicolov et al., 1997). The other approaches to chart generation are based on CFGs and in a bottom-up strategy one has to make sure that in moving from an $\bar{\mathrm{N}}$ to NP all modifiers have been expressed. This causes serious overhead in backtracking. Our use of DTGs and flexible way of adding modifiers using precedence constraints between semantic classes of modifiers does not suffer from this problem.

PROTECTOR does not assume that lexical choice is performed prior to surface realisation. It chunks the input semantics appropriately on the basis of the mapping rules.

6 Conclusions

We have described a sentence generator which takes non-hierarchical input, uses mapping rules to relate parts of the semantics to elementary d-trees, combines the syntactic structures in a manner that closely mirrors the semantic decomposition and employs dynamic programming to avoid re-generation of structures on backtracking which cannot always be predicted in advance as is the case for lexical gaps. Our architecture allows for easy encoding of alternative generation strategies (e.g., bottom-up, best-first, etc.) which other systems have not considered and in fact find rather difficult to do. Thus, PROTECTOR can be seen as a test bed for experimenting and evaluating alternatives methods for generation.

References

Tilman Becker, Wolfgang Finkler, Anne Kilger, and Peter Poller. 1998. An efficient kernel for multilingual generation in speech-to-speech dialogue translation. In *Proceedings of COLING-ACL'98)*, Montreal, Canada.

Rudolf Caspari and Ludwig Schmid. 1994. Parsing und generierung in trug. Technical Report Verbmobil-Report 40, Siemens AG, December.

Christine Doran, Dania Egedi, Beth Ann Hockey, Bangalore Srinivas, and Martin Zaidel. 1994. XTAG — A Wide Coverage Grammar for English. In *Proceedings of the 15th Int. Conference on Computational Linguistics (COLING-94)*, pages 922–928, Kyoto, Japan, 5-9 August.

Dale Gerdemann and Erhard Hinrichs. 1995. Some open problems in head-driven generation. In Jerry Morgan, Georgia Green, and Jennifer Cole, editors, *Linguistics & Computation*, pages 65–197. CSLI Publications.

K. Harbusch, G. Kikui, and A. Kilger. 1994. Default handling in incremental generation. In *Proceedings of the 15th International Conference on Computational Linguistics (COLING-94)*.

Masahiko Haruno, Yasuharu Den, Yuji Matsumoto, and Makoto Nagao. 1993. Bidirectional chart generation of natural language texts. In *Proceedings of the 11th National Conference on Artificial Intelligence (AAAI'93)*, pages 350–356, Menlo Park, CA. American Association for Artificial Intelligence, The MIT Press.

Aravind Joshi. 1987. The Relevance of Tree Adjoining Grammar to Generation. In Gerard Kempen, editor, *Natural Language Generation*, pages 233–252. Kluwer Academic, Dordrecht, The Netherlands.

Martin Kay. 1996. Chart generation. In *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics (ACL'96)*, pages 200–204, Santa Cruz, California.

Kathleen F. McCoy, K. Vijay-Shanker, and Gijoo Yang. 1992. A functional approach to generation with tag. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics(ACL'92)*, pages 48–55, University of Delaware, U.S. Association of Computational Linguistics.

David McDonald and James Pustejovsky. 1985. TAGs as a grammatical formalism for generation. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 94–103.

Marie Wenzel Meteer. 1990. *The "Generation Gap": The Problem of Expressibility in Text Planning*. Ph.D. thesis, Computer and Information Science Department, University of Massachusetts, February. COINS Technical Report 90-04.

Nicolas Nicolov, Chris Mellish, and Graeme Ritchie. 1995. Sentence Generation from Conceptual Graphs. In Gerard Ellis, Robert Levinson, William Rich, and John Sowa, editors, *LNAI 954, Conceptual Structures: Applications, Implementation and Theory*, pages 74–88. Springer, Berlin, 14–18 August. Proceedings of the 3rd Int. Conf. on Conceptual Structures (ICCS'95), Santa Cruz, CA, USA.

Nicolas Nicolov, Chris Mellish, and Graeme Ritchie. 1997. Approximate Chart Generation from Non-Hierarchical Representations. In Ruslan Mitkov & Nicolas Nicolov, editor, *Recent Advances in Natural Language Processing: Selected papers from RANLP'95*, Current Issues in Linguistic Theory (CILT), 136, pages 273–294. John Benjamins, Amsterdam & Philadelphia.

Nicolas Nicolov. 1993. *Head Selection in NLG*. DAI discussion paper 140, Dept. of Artificial Itelligence, Univ. of Edinburgh.

Fabio Pianesi. 1993. Head-driven bottom-up generation and Government and Binding: a unified perspective. In Helmut Horacek and Michael Zock, editors, *New Concepts in Natural Language Generation*, chapter 3, pages 187–214. Pinter, London.

Owen Rambow, K. Vijay-Shanker, and David Weir. 1995. D-Tree Grammars. In *Proceedings of the 33rd Meeting of the Association for Computational Linguistics (ACL'95)*, pages 151–158.

Stuart Shieber and Yves Schabes. 1991. Generation and synchronous tree-adjoing grammars. *Computational Intelligence*, 7(4):220–228. Special issue on Natural Language Generation.

Stuart M. Shieber. 1988. A uniform architecture for parsing and generation. In *COLING-88*, Budapest.

John F. Sowa. 1992. Conceptual graphs summary. In Timothy E. Nagle, Janice A. Nagle, Laurie L. Gerholz, and Peter W. Eklund, editors, *Conceptual Structures: Current Research and Practice*, Ellis Horwood Series in Workshops, pages 3–51. Ellis Horwood Limited, London, England.

Matthew Stone and Christine Doran. 1997. Sentence Planning as Description Using Tree Adjoining Grammar. In *Proceedings of the 35th Annual Meeting of the Association of Computational Linguistics(ACL'97)*, pages 198–205, Madrid, Spain.

Wolfgang Wahlster, Elisabeth André Son Bandyopadhyay, Winfried Graf, and Thomas Rist. 1991. WIP: the coordinated generation of multimodal presentations from a common representation. RR 91-08, DFKI.