Generation in the LOLITA system : An engineering approach.

Mark H. Smith, Roberto Garigliano, Richard G. Morgan
Laboratory for Natural Language Engineering,
University of Durham
E-mail: m.h.smith@durham.ac.uk

1 Introduction

This short paper will outline NLG work in the LOLITA (Large-scale Object-based Linguistic Interactor Translator and Analyser) system. Section 2 will provide background into the natural language engineering principles employed in the development of the system together with a brief overview of LOLITA itself. Section 3 will then provide an overview of the generation process and outline two specific stages in this process: abstract transformations and realisation.

2 Background

2.1 Natural Language Engineering

NL research at Durham University is concerned with natural language engineering (NLE) rather than the more traditional computational linguistics (CL). A lot of CL orientated NLP (including NLG) has concentrated on either trying to formulate universal theories that cover all aspects of language or developing very restricted theories which model small areas. The utilisation or expansion of these ideas to realistic systems which are not highly restricted by their task or domain has proved a great problem. Problems associated with other engineering disciplines which have to be considered in NL are:

Scale: The size of systems (e.g., grammar coverage, vocabulary size, word senses) must be sufficient for realistic large-scale applications.

Integration: Components of a system must not make unreasonable assumptions about other parts. This is often the case when specific NLP problems are tackled in isolation. Components should be designed and implemented so that they assist other components.

Flexibility: The ability to modify systems for different tasks in different domains.

Feasibility: For example, hardware requirements must not be too great and execution speeds must be acceptable. This process incorporates making the system and its components efficient.

Maintainability: How useful the system is over a long period of time. The maintenance of a large system has proved to be an important aspect of the software life-cycle [6].

Usability: The system must be able to support the applications end users want and be user-friendly.

Robustness: This is a critical aspect of large-scale systems. To quote [1] "while it [robustness] may not be a serious problem for any individual application, it has to be faced up to in general". This aspect concerns not only the linguistic scope of the system but how it deals with input which falls outside of this scope.

The fact that there are a large number of systems and projects with very restrictive aims and few that can claim to successfully address these issues suggest that they have associated **intrinsic research problems of their own**.

The NLE method has foundations in the belief that it is not necessary to wait for complete linguistic theories covering all the problems associated with NL (which do not exist at present) before large, realistic and useful NL systems can be built. Instead a full array of AI techniques is employed ranging from using well-developed linguistic and logic global theories (where they exist) to more localised theories, corpora, knowledge based heuristics, adaptive techniques and at the lowest level ad-hoc rules. Incorporating this wide range of methods means that the development of the system does not get stuck due to the difficulty in following a particular logical or linguistic theory while the benefits of such well established theories can still be enjoyed. The result is a practical, working solution.

2.2 The LOLITA system

The LOLITA system has been developed over the last eight years at Durham University. It belongs to only a small group of systems which can claim to have addressed most of the properties required of large engineered systems as described above. The rarity of systems such as LOLITA is exemplified by the fact that NL system terminology defined in [1] has to be extended to define LOLITA's status; it is more than a generic system as it is not restricted to a single task type, but it is not, as it stands, a general purpose machine which can be used for any task in any domain. We extend the terminology by defining LOLITA as a *general purpose base*. Although demonstration prototypes have been built using LOLITA

for various tasks and domains (e.g., template building, dialogue analysis and generation, interactive query, machine translation, chinese tutoring [2]), no polished final application has yet been developed. This is because our research resources have been concentrated on the 'base' of the system and thus the task-dependent development has not resulted in such systems.

LOLITA is built around an original kind of conceptual graph (a logically precise form of semantic network, see for example [12]) which is accessed, modified and manipulated by the other system components (including the generator). This representation, which holds world information and data as well as some of its linguistic data, currently comprises over 30,000 nodes (capable of more than 100,000 inflected word forms). It is presently being integrated with Princeton's WordNet [11] which will increase its size to more than 70,000 nodes. LOLITA can parse text, semantically and pragmatically analyse its meaning and add relevant information to the semantic network. It has been built to cope with *full* and *serious* text [1] (e.g. newspaper articles), as well as text containing errors.

LOLITA is implemented in the functional language Haskell. It comprises a total of approximately 32000 lines of Haskell code (corresponding to about 300,000 lines of, e.g., 'C').

3 Generation

A lot of work in the area of NLG has suffered from the problem of poor *integration*. Some generators have been built for specific tasks in specific domains and cannot easily be transported to others (that is they have poor *flexibility*). Other generators have been used as an interface to a variety of applications (e.g, Penman [3] [8], Spokesman [10]) but have been designed and built in isolation as separate components. This approach has led to researchers making some unlikely assumptions as to the input to their generators. For example, some generators assume the existence of a set of clause-size predicates from which the generator must choose, organise and realise [3].

The LOLITA generator has been developed as a part of a complete NL system. It has been built in tandem with the semantic network representation from which it generates and each component has influenced the development of the other (e.g. the recent improvement in the semantic representation of time and location was implemented with the requirements of the generator module in mind). This highly integrated approach prevents the problem of lack of expressibility found in other systems [10]. Workers have found that the input representation they use or assume is not isomorphic with the linguistic realisation resources they employ. This 'generation gap' problem does not arise in LOLITA as the semantic network representation is always expressible in surface NL.

Although pieces of semantic network are always directly realisable, the generation allows *flexibility* and high *usability* by performing transformations on the representation to allow generation to be tailored for different tasks. Parameters dependent on the particular application, the context, the required style and analysis of the dialogue situation [5] are used to guide the control and effect of these transformations on the final text. Of course, control of variation is a difficult problem and research into this area is still very much on-going.

The transformations can be roughly categorised into the common planning and realisation divisions but again, the high *integration* of the generator and semantic representation avoids some problems encountered by others. For example, the organisation of clauses according to some discourse structure relations [7] (particularly the *ideational relations*) is already explicit in the semantic representation and does not have to be achieved by the generator. Of course, the problem of how and when to make these relations explicit in the final text is a generation task.

The generator is largely description directed (or procedural) [9]: the content of the semantic network to be expressed plays a large part in the control of the generator. This method is usually more efficient than grammar directed control (e.g., functional unification is inherently non-deterministic). The *feasibility* of the generator is very acceptable: it does not require extensive memory and operates in real time.

Because the semantic network is large and contains a vast amount of linguistic knowledge (e.g. after the incorporation of WordNet) the generator is very large scale with respect to lexical information. The grammatical coverage of the generator, however, is not particularly good. This problem is closely linked with that of *maintainability*. While so far the coverage of grammar has been increased relatively easily, it is expected that because of the lack of a separate grammar, this may be a future problem. This lack of coverage caused by poor maintainability is not as paramount to a similar problem in, for example, parsing: a portion of semantic network can always be realised and so *robustness* is not affected. The separation of the grammar is to be investigated in the near future and it is believed that this will improve the maintainability and grammatical coverage of the generator.

A full description of the LOLITA generator cannot be presented here but, by way of example, two transformations which operate on the semantic network during the generation process are now outlined.

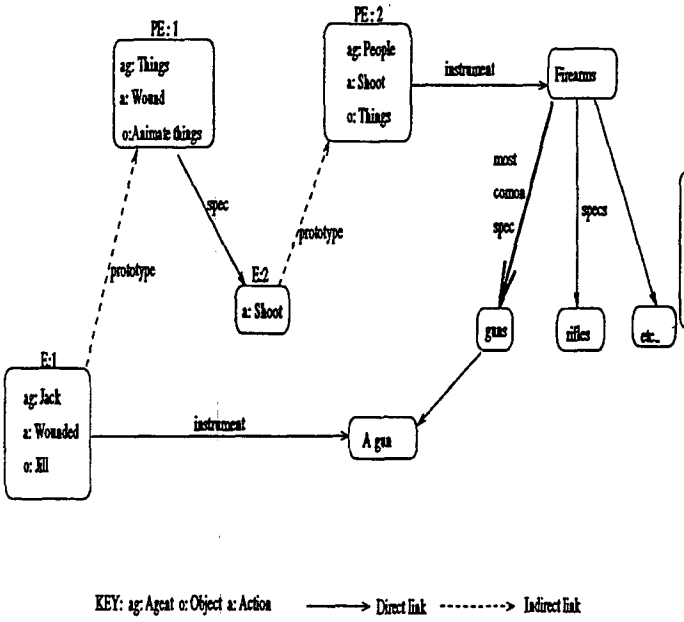

Figure 1: Example of a verb/instrument specialisation transform

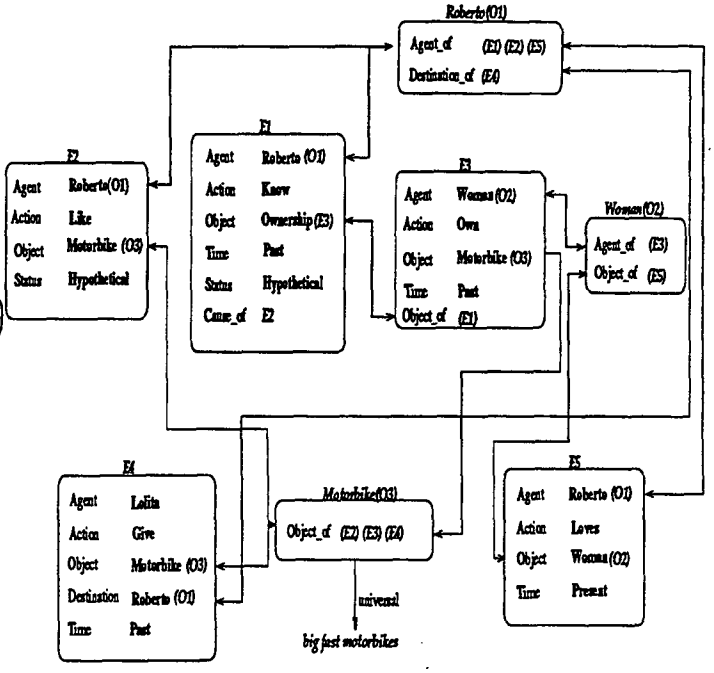

Figure 2: Simplified portion of semantic network for the realisation example

3.1 Abstract Transformations

Abstract transformations are low-level transformations which act on the semantic network immediately before realisation. Associated with each abstract transformation are theoretical issues on why particular normal forms are chosen, rules which allow us to move away from these normal forms and effects the transform has on the final utterance (apart from the obvious one that variations are more natural).

Abstract transformations can act on and produce variations for events with, for example, antonym verbs, copula verbs, complemented verbs, verbs which have de-lexical structures, verbs which can be either specialised or generalised and events with multiple subjects. In each case the procedure is to disconnect arcs from particular nodes and reconnect them to different ones (which might already exist or have to be constructed in the network). Figure 1 is an example network portion showing how a verb specialisation transform can be applied. The verb in the original sentence 'Jack wounded Jill with a gun' may be specialised into 'shot' because firearms (of which guns are specialisations) usually wound by shooting. Furthermore as 'guns' are considered to be the *most common specialisation* of firearms, the instrument clause can (assuming no special context or high precision flag is set) be dropped leaving 'Jack shot Jill'. This abstract transformation relies on a certain amount of plausible inference: if a high precision flag has been set (according to the required style of output) then the output may be modified, e.g., 'It is likely that Jack shot Jill'.

As in the LOLITA system, Jacob's KING generator [4] uses a knowledge intensive semantic network to move from normal forms of representation to alternative forms. However Jacob's method requires special entries in the semantic network (e.g., a special entry representing 'hug giving' is required to produce the delexical structure 'to give a hug').

3.2 Realisation

Realisation is the final step in the generation hierarchy and involves the traversal of the semantic network to produce sentences. Higher levels in the generation module (planning and dialogue analysis) pass down instructions which indicate which events and relations should be included in the utterance (i.e the content). The textual organisation of the utterance has to be decided by the realisation module using the constraints passed down from above. The general operation of the realiser module is to follow the arcs starting from the input node to find further nodes and their associated information. According to the type of these nodes (e.g., if this node is itself an event node with many links) this process may be continued recursively.

The realiser combines both the deep and surface realisation of the network. Choices between, for example, passive and active, or dative and non-dative, sentences are passed to the realiser as parameters. In a sense the

grammar of the output is hardwired into the code, but variation and flexibilty is allowed through the use of these parameters. This setup has proved sufficient for all the applications for which prototypes have been built.

Figure 2 shows a (much simplified) portion of the semantic network containing an example event E1. The realiser operates by following arcs (e.g., subject, action, object etc) from this event to other nodes in the network and recursively generating expressions for these nodes. If the default parameters are being used the events will be generated in the active voice and the default rhythm is to allow one relative clause for each object. The output produced in this example is "If Roberto knew that the woman whom he loves owned the big fast motorbike that I gave him then he would like it." An un-simplified portion of the semantic network will of course be more richly populated. There may, for example, be many more arcs from the node representing Roberto which link to more information about him. If planning instructions indicate that this information should be expressed it is likely that the realiser will have to split the utterance into separate sentences. Events which are encountered by the realiser which cannot be immediately expressed (because the resulting sentence will be too long) are placed on a stack so that they can be expressed as separate sentences. Heuristics are used to order this stack of events so that coherent focus and decipherable anaphoric references are maintained (the development of these heuristics is ongoing).

Stylistic variations can be produced by altering the realiser parameter settings and passing different nodes to the realiser so the text is realised from a different angle. Example parameter switches are Passive/Active, Dative/Non-Dative, Colour, Rhythm, Length, De-lexical transformations, Copula transformations, Complement transformations, Synonyms transformations, Verb Antonym transformations, Verb Specialisation, Verb Generalisation.

4 Conclusion

This paper has given a brief outline of the LOLITA system and some aspects of its generation component. Because of the commercial value of the LOLITA project, the system is not publicly available. However, we are extremely keen to give demonstrations of the system: for any information on the LOLITA project, please contact the authors.

References

[1] J. Galliers and K. Sparck-Jones. Evaluating natural langauge processing systems. Technical Report 291, Computer Laboratory, University of Cambridge, 1993.

[2] R. Garigliano, R. Morgan, et al. *The LOLITA Project: The First Seven Years.* Under negotiation with Afterhurst Ltd., forthcoming, 1994.

[3] E. H. Hovy. Unresolved issues in paragraph planning. In R. Dale, C. Mellish, and M. Zock, editors, *Current Research in Natural Language Generation*, pages 17–45. Academic Press, New York, 1990.

[4] P. S. Jacobs. Knowledge-intensive natural language generation. *Artificial Intelligence*, 33(3):325–378, November 1987.

[5] C. Jones and R. Garigliano. Dialogue analysis and generation: A theory for modelling natural english dialogue. In *EUROSPEECH '93 volume 2*, pages 951–954, September 1993.

[6] B. Lientz and E. Swanson. *Software Maintenance Management.* Addison-Wesley, 1980.

[7] E. Maier and E. H. Hovy. Organising discourse structure relations using metafunctions. In H. Horacek and M. Zock, editors, *New concepts in Natural Language Generation: Planning, Realization, and Systems*, pages 69–86. Pinter Publishers, New York, 1993.

[8] W. C. Mann. An overview of the Penman text generation system. In *Proceedings of the Third National Conference on Artificial Intelligence (AAAI-83)*, pages 261–265, Washington, DC, August 22-26, 1983.

[9] D. D. McDonald, M. M. Meteer, and J. D. Pustejovsky. Factors contributing to efficiency in natural language generation. In G. Kempen, editor, *Natural Language Generation: New Results in Artificial Intelligence, Psychology and Linguistics*, NATO ASI Series - 135, pages 159–182. Martinus Nijhoff Publishers, Boston, Dordrecht, 1987.

[10] M. Meteer. *Expressibility and the Problem of Efficient Text Planning.* Francis Pinter Publishers, London, 1993.

[11] G. Miller. Wordnet: An on-line lexical database. *International Journal of Lexicography*, 3(4), 1990.

[12] J. Sowa. *Conceptual Structures (Information Processing in Mind and Machine).* Addison-Wesley, 1984.