Published in: The proceedings of The Seventh International Conference of the European Chapter of the Association for Computational Linguistics, (EACL-95), Student Session, pp 286-290, Dublin, Ireland, March 27-31, 1995.

Aggregation in the NL-generator of the VIsual and Natural language Specification Tool

Hercules Dalianis
Department of Computer and Systems Sciences
The Royal Institute of Technology and
Stockholm University
Electrum 230
S-164 40 Kista
Sweden
ph. 08-668 90 98
mob. ph. 010-668 13 59
email: hercules@dsv.su.se

Abstract

In this paper we show how to use the so-called aggregation technique to remove redundancies in the fact base of the Visual and Natural language Specification Tool (VINST). The current aggregation modules of the natural language generator of VINST is described and an improvement is proposed with one new aggregation rule and a bidirectional grammar.

1.Introduction

This paper describes the aggregation process in the natural language generator of the Visual and Natural language Specification Tool (VINST), and how the aggregation can be improved.

Aggregation is the process which removes redundancy in texts. Redundancy typically occurs when the material selected for communication contains information that is duplicated in the text, or else is so closely related that the reader can automatically infer one piece when reading another. Aggregation is also called ellipsis by linguists.

In the VINST-system Natural Language generation is applied in various places. In the specification part to paraphrase the rules expressed in formal language, to paraphrase automata, further on to paraphrase questions asked to the theorem prover, and to paraphrase the executed events and the newly created fact base. We will in this paper only treat the generation of NL from fact bases.

The kind of text produced in this domain is illustrated in the right hand window of VINST in figure 1.

When generating a text from a fact base in VINST the text becomes very tedious to read since the text is very redundant and does not feel correct conceptually. To make the text smoother to read a new architecture is suggested where a new aggregation rule from (Dalianis & Hovy 1993) is suggested to be used, namely predicate grouping rule.

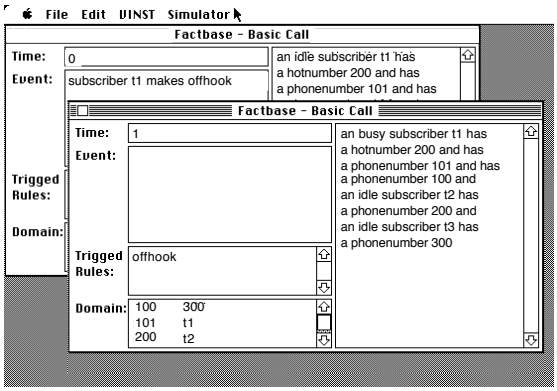

Figure 1. The event window, where the user can execute events and the inter preter interprets the specification.

The VINST-system is a multi-modal specification and validation tool, specifically for the functionality of telecom services. The specification is carried out with a Visual Language (VL) and a restricted Natural Language (NL), which are translated to LOXY (Echarti & Stålmarck 1988), a First Order Logic language extended with time.

The VINST system is a ready usable prototype which has been demonstrated and tested on various sites in the world (Engstedt 1991, Preifelt & Engstedt 1992). The VINST prototype is implemented in AAIS-Prolog and SuperCard on Macintosh. The Prolog is used for the NL-system and the SuperCard for the VL-part and for the user interaction of the system.

2.Previous research

Several studies on aggregating text based on text structure appear in the literature. In fact, the term *aggregation* was first used in (Mann & Moore 1980). In (Horacek 1992), is described the integration of aggregation (which he calls grouping) with quantification under guidance of principles of conversational implicature. (Dale 1990) calls it discourse level optimization, (Kempen 1991) calls it forward and backward conjunction reduction.

In (Hovy 1990) two structural aggregation rules are used to eliminate redundant information. In an example in (Scott & de Souza 1990), nine heuristic rules aggregate six sentences which express a set of facts using a single sentence. In (Dalianis & Hovy 1993) are eight different aggregation rules described.

3.The current NL-generator

To solve the problem of the not "naturalness" of the LOXY-formulas and make them more "natural" the following two modules have been constructed: the *natural* and *compact* modules and finally the *surface* grammar.

The LOXY-formula which is to be paraphrased is processed step by step to natural language by the different modules to a deep structure. The *natural,* and *compact* modules can be activated and deactivated separately. Finally the surface generator generates natural language text from the deep structure.

The *surface* grammar contains its own generation grammar and uses the same dictionary as the NL-parser. The surface generation grammar is a Definite Clause Grammar, DCG, (Pereira & Warren 1980, Clocksin & Mellish 1984), and is not treated in this paper.

4.Natural module

The *natural* module creates a deep structure from the flat LOXY-formula, by looking up its elements in the dictionary. From this information it can decide what the deep structure should look like. The natural module is also called sentence planner. i.e. it plans the length and the internal order of the different sentences.

t1 is a subscriber and t1 is idle and
t1 has 100 and 100 is a phonenumber and t1
has 101 and 101 is a phonenumber and
t2 is a subscriber and t2 is idle and
t2 has 200 and 200 is a phonenumber.

Figure 2a) Normal mode, only surface generation.

The natural module does what (Dalianis & Hovy 1993) calls ordering and economy.

an idle subscriber t1 has a phonenumber 100 and
an idle subscriber t1 has a phonenumber 101
and
an idle subscriber t2 has a phonenumber 200.

Figure 2b) Natural mode

5.Compact module

The natural language expression, after being processed by the natural module has a lot of redundant noun phrases. This is solved by the compact module. Our aggregation rule says: If two or more identical (and hence redundant) noun phrases are repeated consecutive then remove all the noun phrases except the first one This operation will remove the repetitive generation of the noun phrase and the text becomes concise. (Dalianis & Hovy 1993) calls this subject grouping.

an idle subscriber t1 has a phonenumber 100 and
has a phonenumber 101 and
an idle subscriber t2 has a phonenumber 200.

Figure 2c) Natural mode + compact mode

What we see is that the text can be aggregated in a different way and also that the subject grouping has not been fully applied on the *phonenumbers*.

6.Paraphrase fact bases

Fact bases can be paraphrased into natural language either after that an event is executed with the interpreter or as an answer to a question to the theorem prover. Here we show an example of the latter, (see Figure 3).

A question expressed in NL (It is difficult to express questions in VL) is translated to a LOXY expression that the theorem prover tries to prove. The generation module takes the proved query and generates an NL-answer.

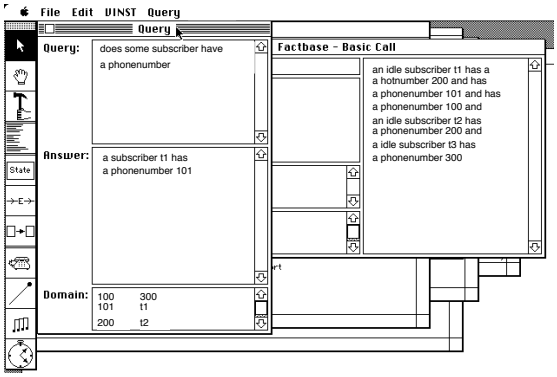

Figure 3. The query window, where the user can ask questions and obtain answers via the theorem prover.

7.Improvements on architecture

The present natural language generator of VINST is difficult to control because there are only two control features (natural and compact) available. It is required great effort to adapt the NL-generator to new domains or to extend it without writing new grammar rules. Further on it is difficult to express the NL-paraphrase in a similar fashion as the user expresses him/herself, therefore are some improvements suggested.

One suggestion is is to use as a natural language grammar the Core Language Engine (CLE) (Alshawi 1992). CLE is a bidirectional, unification and feature-based grammar written in Prolog.

CLE uses Quasi Logical Form (QLF) as linguistic representation for the parsed NL-string. QLF can be used to direct the generator, but it needs to be augmented. We have to construct an Intermediate Generation Form (IGF) which will contain the suitable linguistic primitives. The IGF will be acquired both from the user and from the context where the NL is to be paraphrased. e.g. simulation- or query window. The used words of the user will be reused for generation together with the LOXY formula.

When the paraphrasing will be carried out from a VL-expression, then we have to use preset linguistic primitives and words for the NL-generation because there will not be any linguistic primitives.

8.Intermediate Generation Form

The Intermediate Generation Form (IGF) will contain the type of sentences, e.g. a fact or an assertion (dcl), a rule (rule), a yes-no-question (ynq), a what, which or who-question (whq), a noun phrase (np) and many more.

The Quasi Logical Form (QLF) of CLE uses already *dcl, ynq and whq* and could be extended to also treat *np* . The rest of the type of sentences are context dependent. i.e. *rule* etc. The sentence types above are identical with the ones in the QLF, except of the sentence type *np* and some others which are VINST specific.

To each type of sentence, above, there is a set of features. e.g. adjective form (adj), subjective predicative complement (predcomp), subject grouping (sg) and predicate grouping(pg) and many more.

The features can be unordered and the number can be arbitrary. Some of the features are the same as the one QLF uses, except for: *predcomp*, *sg* and *pg*.

The IGF contains also two aggregation features; subject and predicate grouping which makes the text nicer to read.

Observe that there is no time feature in the IGF, since LOXY has an embedded time.

What we also need is a list of words used by the user. The words are obtained from the parser. The IGF needs to be stored together with the LOXY expression until they are going to be used by the NL-generator. The syntax of the IGF is described by showing the Prolog predicate int_gen_form/3 and its content.

```
int_gen_form(REFNR,
        TYPE(FEATURE_LIST),
                USED_WORD_LIST).
```

REFNR is a reference number to the LOXY-expression to be paraphrased. TYPE is type of sentence and FEATURE_LIST is a list of feature names describing the sentences.
USED_WORD_LIST is a list of previous used words.

9.Paraphrase fact bases aggregated

Here follows two examples on how the paraphrasing would look like with the new architecture upon paraphrasing a LOXY-fact base to NL, (Not yet implemented)

The only thing which changes between the two examples is the content of the IGF.

Before generation input propositions are ordered based on the characteristics of their subjects, as described in (Dalianis & Hovy 1993).

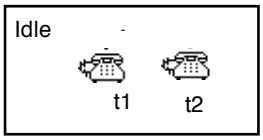

Figure 4. A fact base described in VL

a) fact(2, p(1, subscriber(t1)) &
p(1, idle(t1)) & p(1,has(t1,100)) &
p(1,has(t1,101)) &
p(1, phonenumber(100)) &
p(1, phonenumber(101)) &
p(1, subscriber(t2)) & p(1, idle(t2)) &
p(1,has(t2,200)) &
p(1, phonenumber(200)))).

b) int_gen_form(2,dcl([predcomp,sg]),
[subscriber,idle,be,have,phonenumber]).
t1 is a subscriber and is idle and
has the phonenumber 100 and 101
t2 is a subscriber and is idle and
has the phonenumber 200

c) int_gen_form(2,dcl([adj,sg,pg),
[subscriber,idle, be, have,phonenumber]).
t1 and t2 are idle subscribers and
t1 has the phonenumbers 100 and 101 and
t2 has the phonenumber 200.

In the second NL-example, figure 4c), we see how the predicate grouping works.

10.Conclusions and future work

We have in this paper shortly described the current NL-generator of the VINST-system. We have found it too inflexible and the generated text too tedious to read, therefore is suggested a new NL-architecture where the user and the context of the user interaction is used to extract an Intermediate Generation Form (IGF). The IGF will contain a new aggregation rule, the so called predicate grouping rule which will make the generated text easier to read, further on is proposed to use a bidirectional grammar for the surface generation.

One future suggestion is also to use the results from the NL-parsing for the generation.

Acknowledgments

Many thanks to Ed Hovy at Information Sciences Institute/USC for advising me and for stimulating discussions via email and many thanks also to Stefan Preifelt and Måns Engstedt at Ellemtel Telecommunication Systems Laboratory, for being inspiring workmates during the VINST prototype implementation and also for introducing me to the telecom domain.

References

Alshawi, H. ed. (1992). The Core Language Engine, *MIT Press*.

Clocksin, W.F. & Mellish, C.S. (1984). Programming in Prolog, *Springer Verlag*.

Dale, R. (1990). Generating Recipes: An Over view of Epicure, *In Current Research in Natural Language Generation,* Dale, R. et al (Eds). Academic Press Limited, pp. 229-255.

Dalianis, H. & Hovy, E. (1993). Aggregation in Natural Language Generation: *In the Proceedings of the Fourth European Workshop on Natural Language Generation,* Pisa, Italy, 28-30 April.

Echarti, J-P. & Stålmarck, G. (1988). A logical framework for specifying discrete dynamic systems, *Advanced Systems Development Dept., Ellemtel Telecommunication Systems Laboratory*, Älvsjö, Sweden..

Engstedt, M. (1991). A flexible specification language using Natural Language and Graphics, Master Thesis,*Centre for Cognitive Science*, University of Edinburgh.

Horacek, H. (1992). An integrated view of textplanning: *In Aspects of Automated Natural Language Generation, Dale, R. et al, (eds).,* Springer Verlag Lecture Notes in Artifical Intelligence no 587, pp. 193-227.

Hovy, E.H. (1990). Unresolved Issues in Paragraph Planning: *In Current Research in Natural Language Generation,* R. Dale, et al, (eds)., Academic Press Limited, pp. 17-45.

Kempen, G. (1991). Conjunction reduction and gapping in clause-level coordination: An inheritance-based approach: *In Computational Intelligence,* Vol 7, No 4, pp. 357-360.

Mann, W.C. & Moore, J.A. (1980). Computer as Author - Results and Prospects, *Report/ISI/RR-79-82*, University of Southern California/ Information Sciences Institute.

Pereira, F.C.N & Warren, D.H.D. (1980). Definite Clause Grammars for Language Analysis - A Survey of the Formalism and a Comparison with Augmented Transition Networks. *J. of Artificial Intelligence* 13, pp. 231-278.

Preifelt, S & Engstedt, M. (1992). Resultat från VINST projektet, (In Swedish, Results from the VINST project), *Ellemtel Telecommunicat ion Systems Laboratory*, Älvsjö, Sweden.

Scott, D. & de Souza, C.S. (1990). Getting the Message Across in RST-based Text Generation:I*n Current Research in Natural Language Generation,* R. et al, (eds). Academic Press Limited, pp. 47-73.