GENERATING ELLIPSIS USING DISCOURSE STRUCTURES

Feikje Hielkema, Mariët Theune & Petra Hendriks
University of Groningen; University of Twente
Feikje@ai.rug.nl, M.Theune@ewi.utwente.nl, P.Hendriks@let.rug.nl

Abstract

This article describes an effort to generate elliptic sentences, using Dependency Trees connected by Discourse Relations as input. We contend that the process of syntactic aggregation should be performed in the Surface Realization stage of the language generation process, and that Dependency Trees with Rhetorical Relations are excellent input for a generation system that has to generate ellipsis. We also propose a taxonomy of the most common Dutch cue words, grouped according to the kind of discourse relations they signal.

1 Introduction

Ellipsis and co-ordination are key features of natural language. For a Natural Language Generation system to produce fluent, coherent texts, it must be able to generate co-ordinated and elliptic sentences. This article describes an effort to implement syntactic aggregation in the Virtual Storyteller, a Java-based story generation system. The focus lies on the generation of co-ordinated and elliptic structures for the Dutch language.

In this abstract, syntactic aggregation is defined as the process of combining two clauses at the surface level using any kind of syntactic structure, for instance ellipsis, co-ordination or a subordinate clause. We make two important claims:

- The process of syntactic aggregation belongs in the Surface Realizer module of a natural language generation system
- The combination of dependency trees and discourse structures constitutes excellent input to the Surface Realizer

Cue phrases are a natural language's resources to signal different rhetorical relations, and as such are a vital part of syntactic aggregation. They have great influence on the syntactic structure of an aggregated sentence. Therefore we have designed a taxonomy of the most common Dutch cue words, to use in the aggregation process.

The Virtual Storyteller is a multi-agent NLG-system that creates and narrates fairy tales. At the time of writing, the generated tales feature only two characters, but have different events and endings. The plot is created by so-called Actor agents, autonomous computer programs that represent the characters in the story, each with its own personality and goals. They are able to reason logically and are affected by emotions. These emotions, when felt strongly, cause the agent to adopt a new goal that overrides its original goal.

The Narrator Agent transforms the plot into text. The initial version of the Narrator only presented the bare facts of the story, mapping the events from the plot to simple, fixed

sentences. This resulted in a very monotone, uninteresting narrative. Syntactic aggregation should help enormously to improve the liveliness of the generated narratives. The goal of our project was to make the Narrator produce at least the following structures:

- Paratactic constructions: *Diana left the desert and Brutus entered the forest*
- Hypotactic constructions*: Diana left the desert, because she saw Brutus*
- Conjunction Reduction: *Diana entered the desert and saw Brutus*
- Right Node Raising: *Diana entered and Brutus left the desert*
- Gapping: *Diana entered the desert and Brutus the forest*
- Stripping: *Diana entered the desert and Brutus too*

Different cue words should be available, to improve the variety and to signal different rhetorical relations. Although our work aims in the first place at improving the texts produced by our story generation system, we believe that our approach to syntactic aggregation and ellipsis is sufficiently general to be relevant for all kinds of language generation systems. In addition, we will argue that our approach is largely language-independent.

The rest of this paper is structured as follows. In section 2 we describe the design of the Narrator, and argue that syntactic aggregation should be located within the last stage of the language generation process. In section 3, we present the cue word taxonomy that we developed for use in the aggregation process, and we discuss how we perform aggregation in our system, using this taxonomy. We end with a brief discussion and conclusions.

2 Design

2.1 The Narrator

The design of the Narrator agent is based on the three stage pipe-line structure described by Reiter & Dale (2000). The Content (Document) Planner receives as input a list of propositions representing the events that make up the plot, plus related background material (typically, characters' actions and their causes). It removes superfluous information and adds rhetorical relations between the propositions (see section 3.2.2). The Sentence Planner (Microplanner) maps the propositions to Dependency Trees (see section 3.2.1), while maintaining the rhetorical relations between them. Finally, the Surface Realizer performs syntactic aggregation and generates the surface form.

Although Reiter & Dale (2000) see aggregation as a Microplanning task, we have decided to situate syntactic aggregation in the Surface Realizer module instead. The RAGS-project (Cahill & Reape, 1999) showed a lack of consensus on the location of the aggregation process in the NLG pipe-line; instead the situation varied widely over different NLG-systems. This divergence is partly caused by the fact that many, quite different processes are gathered under 'aggregation' (see Reape & Mellish, 1999). Our project only dealt with syntactic aggregation. As syntactic aggregation deals with grammatical processes (co-ordinating sentences and deciding which elements can be

deleted without rendering the sentence ungrammatical), in our view it should be situated with the other grammatical processes, in the Surface Realizer.

2.2 The Surface Realizer

The Surface Realizer receives as input a 'Rhetorical Dependency Graph': a graph specifying the rhetorical relations between sentences, represented as *Dependency Trees.* Dependency Trees are a prominent feature of Meaning-Text theory (Mel'cuk, 1988). They are constructed on the basis of predicates and arguments. There is no dependency on linear word order, and no limit on the amount of children a node can have. This means the trees are able to handle variation in word order easily, so that they translate well over different languages. In fact, Dependency Trees have been used with success in Machine Translation (Lavoie et al., 1999). This means that a generation system using Dependency Trees can be adjusted to another language quite easily; only the rules specific to the generated language have to be replaced, the generation algorithm remains intact.

The independence of word order, and the dependency labels that specify which role a node performs in its parent syntactic category, cause Dependency Trees to be easy to manipulate, especially for the purpose of generating ellipsis. In our project, we follow the Alpino format for Dependency Trees (Bouma et al., 2001), with some minor changes. A tag for morphology has been added, and the tags that indicate the position of a word or syntactic category in the Surface Form are left out.

The discourse structure graphs, of which the Dependency Trees form the nodes, were inspired by Rhetorical Structure Theory (Mann & Thompson, 1987). This theory was originally developed as a descriptive framework for the analysis of text structure, but it is also used in several NLG applications (for an overview, see Hovy, 1993). Among other things, rhetorical relations influence the syntactic structure of a co-ordinated and elliptic sentence. Shaw (2002) uses discourse structures to determine the syntactic structure that should be used to combine two propositions. Hendriks (2004) showed that rhetorical relations do hold as well for certain elliptic structures, such as gapping: a gapped sentence can only have a Resemblance relation between its clauses.

As well as Dependency Trees, rhetorical relations are language independent, as is illustrated by the fact that Rhetorical Structure Theory has been applied successfully to various languages including English, Dutch, German, and Portuguese. This means that rhetorical structures can be passed on to the Surface Realizer, to be used for the selection of a correct and fitting co-ordinating syntactic and/or elliptic structure, when combining two Dependency Trees. The rhetorical relations that we currently distinguish are Cause, Contrast, Purpose, Temporal and Additive. These were judged to be most important for storytelling. The number of relations can be easily expanded if necessary.

Because the Narrator is closely connected to the rest of the multi-agent system, it can access the reasoning, plans and emotions of the Actor agents, and use this information to determine which rhetorical relations to include in the Rhetorical Dependency Graph. For instance, an agent makes a plan to reach a certain goal (Purpose), or can have two conflicting goals (Contrast).

3 Cue Phrase Taxonomy

Sanders & Noordman (2000) show that coherence (rhetorical) relations play an important part in human text processing, and that linguistic markers (cue phrases) cause faster processing of coherence relations between two segments. As a useful linguistic feature in text processing, cue phrases should certainly be included in text generation.

3.1 Related work: Taxonomies for English and Dutch cue phrases

Knott & Dale (1993) use cue phrases as an objective measure to determine a set of rhetorical relations that is based on linguistic evidence. They classified a corpus of cue phrases according to their function in discourse, using a substitutability test. Put simply, this test is used to determine whether two cue phrases signal (partly) the same features, by checking whether one can be substituted by the other in a particular context.

On this basis a taxonomy of cue phrases was created. This taxonomy was hierarchical, as some cue phrases signal more features than others. Knott & Sanders (1998) have created a similar taxonomy for Dutch cue phrases, using the cognitive primitives that were proposed by Sanders et al. (1992) to differentiate between the classes. However, this taxonomy is rather complex and will presumably be hard to implement. Moreover, Knott & Sanders admit that their taxonomy was created using only those cue phrases that were easiest to classify; other cue words will be even harder to classify and cause the taxonomy to be even more of a labyrinth. For these reasons we have decided to create a less convoluted taxonomy for our own purposes.

3.2 Cue Word Taxonomy for syntactic aggregation

For the purpose of syntactic aggregation in our storytelling system, a small taxonomy charting only the most prevalent cue words in Dutch, has been constructed using a variant of the substitutability test described by Knott & Dale (1993). Because the taxonomy, given in figure 1, is meant to be used by the Surface Realizer *before* the words are ordered to produce the surface form (linearization), unlike Knott and Dale we paid no attention to any changes a cue word might make in the word order in the clauses. The clause order could be changed as well, if this did not influence the meaning of the sentence. In short, we only looked at substitutability with respect to meaning, regardless of surface form.

The test was used on two types of data: sentences taken from a fairy tale book (Andersen, 1975), and sentences based on the output of the story generating system. The tested cue words all had a frequency of over a hundred in a representative sample from the Spoken Dutch Corpus (Wouden et al., 2003), to exclude rare cue words. Only cue words that seemed appropriate for narrating a story (not too difficult, because the target group are children) were included.

When grouping the cue words, it turned out that rhetorical relations such as Cause, Purpose and Contrast were not enough to distinguish the classes; extra variables needed to be introduced. We selected features which are in principle available to the story generating system, such as volitionality and the chronological ordering of events.

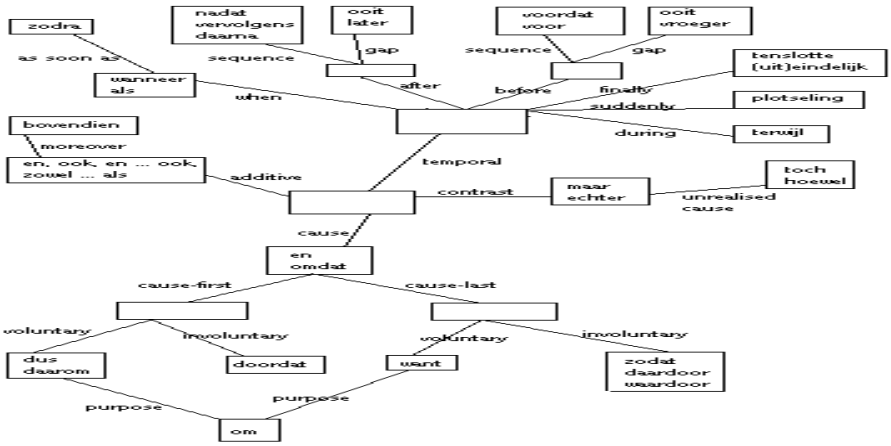

Figure 1: Cue word Taxonomy

3.3 Using the taxonomy

The cue word taxonomy is used in the Surface Realizer, during the syntactic aggregation process. On the basis of the rhetorical relation between two Dependency Trees, an appropriate cue word is selected that has, if possible, not recently been used (to achieve some variation in the generated texts). The cue word is given in the shape of a Dependency Tree node, with its part-of-speech tag and its dependency relation. The relation and the cue word are then passed to the Conjunction algorithm. This algorithm either combines both clauses with the cue word, or adds the cue word to one of them, depending on the cue word's dependency relation. The rhetorical relation determines the cue word, and the cue word determines the syntactic construction that is used.

After Conjunction, Elliption takes place. If the aggregate tree is paratactic and contains redundant elements (an identical node in both conjuncts), the Ellipsis algorithm will try to find a suitable elliptic structure, based on the number and kind of identical nodes. The rhetorical relation can rule out certain elliptic structures. For instance, a Causal relation can rule out gapping. Redundant nodes are removed and their parent nodes receive a connection to the twin node in the other conjunct, labeled 'borrowed' to show that the node should not appear in the Surface Form at this point. This way the elliptic conjunct has the same structure as the intact conjunct, but is ellipted in the Surface Form. If necessary, a node can be added (such as 'ook' (too) in Stripping).

Then the tree is ordered, obeying rules dictating the order of the child nodes, using their dependency labels. During this ordering, names of characters, objects or places will be substituted by pronouns if they were the last of their gender to be mentioned, and if they had the same syntactic role then. For instance, in 'Diana is bang. Toch wil Diana Brutus doden', the second occurrence of 'Diana' is replaced by 'zij' (she) because she was the last female named, and was subject both times. All verbs, nouns and determiners are inflected. Punctuation is added when the Surface Form is complete.

4 Results

Using the taxonomy, the Conjunction and Elliption algorithm are able to generate several different sentences on the basis of a given Rhetorical Dependency Graph. Each

relation has several cue words by which it can be expressed, and as these cue words do not all have the same relation, they are expressed by different syntactic structures as well. The program generates hypotactic and paratactic sentences, and can add modifiers to individual trees as well. If the input consists of two simple trees (figure 2 shows the input of the Surface Realizer, and the output of the Elliptor) and a Contrast relation between them, the initial version would have produced: 'Diana is bang. Diana wil Brutus doden' (Diana is scared. Diana wants to kill Brutus). The current system can produce paratactic and hypotactic constructions, or add a modifier to one of the clauses. These constructions can be produced with several different cue words as well. This causes a variety of sentences far greater than the boring sequence of fixed, simple sentences that were generated before.

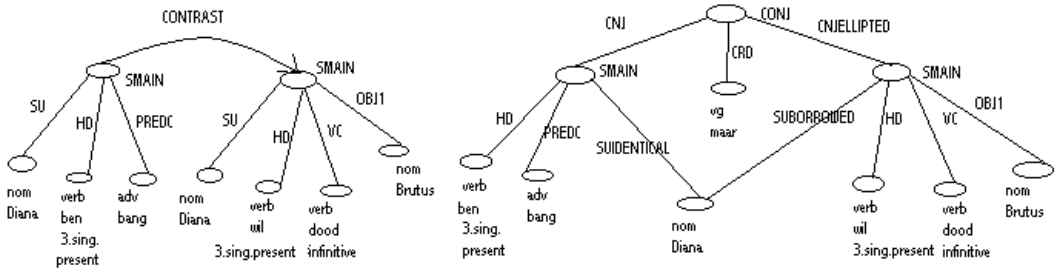

Figure 2: input of the Surface Realizer, output of the Elliptor

If the Conjunction algorithm has generated a paratactic structure, such as 'Diana is bang, maar Diana wil Brutus doden' (Diana is afraid, but Diana wants to kill Brutus), the Elliptor finds a suitable elliptic structure. In this example that is Conjunction-Reduction, resulting in 'Diana is bang, maar wil Brutus doden'. We were able to produce all the desired forms of ellipsis (see section 1), including combinations of different structures, such as gapping and conjunction reduction simultaneously (Diana wants to hug the prince but maim Brutus).

5 General Discussion

5.1 Syntactic Aggregation in the Surface Realizer

As we discussed above, one of the findings of the RAGS-project was that there is no consensus on the right location for aggregation to take place. This is at least partly caused by the use of different definitions of aggregation, which comprise processes quite different in level. Syntactic aggregation deals with grammatical processes, and so the Surface Realizer, the module where linearization is performed using grammatical rules, is the logical place to situate it.

There is evidence that ellipsis is language-dependent. Not all forms of ellipsis are permissible in all languages. Some forms depend on word order, so it seems that ellipsis is influenced by the same grammatical rules that perform linearization. The Surface Realizer already has access to these rules, an extra argument to perform syntactic aggregation (and the generation of ellipsis) in the Surface Realizer.

Discourse structures do not seem to be affected by which language is used. Because rhetorical relations are neutral, they can be processed at the latest stage, after lexicalisation, in the Surface Realizer. This also puts less strain on the component that

has to create the Dependency Trees. The trees only represent simple facts ('Diana is scared', 'Diana flees'), so the trees are uncomplicated.

5.2 Dependency Trees

Is the Dependency Tree language-independent? Mel'cuk (1988) designed Dependency Trees to be free of word order to allow for languages where the word order is vastly different from English. But is it only word order that makes languages differ from one another? In Latin, it is not necessary to mention the subject of a sentence – and that certainly shows in a Dependency Tree. And all languages have some concepts that do not translate well, though these might not crop up often in the telling of a simple fairy tale. Still, even if the Dependency Trees that the Surface Realizer gets as input are not totally language-independent, the methods to process them and turn them into Surface Forms *are*. Substituting the cue words and grammatical rules should be sufficient to enable the Surface Realizer to process Dependency Trees lexicalised to a different language. For this reason alone, we think Dependency Trees are excellent input for a Surface Realizer that tries not to commit itself to one language. Another advantage of using Dependency Trees in syntactic aggregation is that they can easily be manipulated, because the role a word or a constituent plays in a sentence is given by a label (subject-deletion is realized by deleting the node labeled 'subject').

5.3 Discourse Structures

Rhetorical relations can determine which cue words can be used. They are a suitable mechanism to carry a certain meaning across to the level when it is finally of use. And if the relations even influence the grammatical structure that an elliptic sentence can have, the Surface Realizer certainly should have access to them.

However, the relations that were used in this project were not the set that is given by Mann & Thompson (1987). Only a few relations were selected for the moment, those deemed of the most importance to the narrating of a fairy-tale. Cause and Contrast are very basic concepts, and Temporal relations are vital for any narrative. They were then divided into subclasses that correspond to groups of cue words, derived from the small cue word taxonomy that was created for this purpose. The properties that distinguish the subclasses are molded in terms of information that is available to the story generation system. This way, rhetorical relations can easily be added by the Content Planner, because the information is already there. The rhetorical relations that we currently distinguish were selected based on linguistic evidence, on the groups of cue words that were determined. In the future, our cue word grouping should be experimentally confirmed, and if it is not confirmed, the taxonomy should be adapted. Because, as Reape & Mellish (1999) have said, NLG systems should be based on linguistic theories and linguistic evidence to be truly successful.

6 Conclusion

In this article, we have worked toward the following conclusions:

- The most appropriate place for syntactic aggregation is at the level of the Surface Realizer

- The combination of Dependency Trees and rhetorical relations is excellent input for such a Surface Realizer, because Dependency Trees are easily manipulated and rhetorical relations can determine the syntactic constructions that can be used

The Surface Realizer that was created is capable of syntactical aggregation, and of generating several forms of ellipsis that are prevalent in Dutch.

References

Andersen, Hans Christian; Sprookjes en Vertellingen; translated 1975 by W. van Eeden; Bussum, Van Holkema en Warendorf, 1975

Bouma, Van Noord & Malouf; 2001; Alpino: Wide Coverage Computational Analysis of Dutch. In: *Computational Linguistics in the Netherlands CLIN 2000*

Cahill, L. & Reape, M.; 1999; Component tasks in applied NLG Systems; Information Technology Research Institute Technical Report Series

Hendriks, Petra; (2004); Coherence Relations, Ellipsis, and Contrastive Topics; in *Journal of Semantics* 21:2, pp. 133-153.

Hovy, E.;1993; Automated Discourse Generation using Discourse Structure Relations; *in Artificial Intelligence*, pages 341-385

Knott, A. & Sanders, T.; 1998; The Classification of Coherence Relations and their Linguistic Markers: An Exploration of Two Languages; In *Journal of Pragmatics*, Volume 30, pages 135-175

Knott, A. & Dale, R.; 1993; Using Linguistic Phenomena to Motivate a Set of Rhetorical Relations; in *Discourse processes : a multidisciplinary journal,* vol. 18 (1994), afl. 1, pag. 35-62

Lavoie, B., Kittredge, R., Korelsky, T., Rambow, O.; 2000; A Framework for MT and Multilingual NLG systems based on Uniform Lexico-structural Processing; In *Proceedings of ANLP/NAACL 2000*

Mann, William C. and Sandra A. Thompson; 1987; Rhetorical Structure Theory: A Theory of Text Organization, ISI: Information Sciences Institute, Los Angeles, CA, ISI/RS-87-190, 1-81.

Mel'cuk, Igor A. (1988); Dependency Syntax: Theory and Practice; State University of New York

Reape, M. & Mellish, C.; 1999; Just What is Aggregation Anyway?; in *Proc. 7th European Workshop on Natural Language Generation*, 1999

Reiter, Ehud & Dale, Robert (2000); Building Natural Language Generation Systems; Cambridge University Press

Sanders, Ted J.M., Spooren, Wilbert P.M. & Noordman, Leo G.M.; 1992; Toward a Taxonomy of Coherence Relations; in *Discourse Processes* 15, pages 1-35, 1992

Sanders, Ted J.M. & Noordman, Leo G.M.; 2000; The Role of Coherence Relations and their Linguistic Markers in Text Processing; in *Discourse Processes*, Volume 29(1), pages 37-60.

Shaw, J.C.; 2002; Clause Aggregation; an approach to generating Concise Text; Phd thesis, Columbia University

Wouden, T. van der, H. Hoekstra, M. Moortgat, B. Renmans & I. Schuurman; Syntactic Analysis in the Spoken Dutch Corpus. In *M. González Rodriguez & C. Paz Suárez Araujo, Proceedings of the third International Conference on Language Resources and Evaluation*. 768-773.