Published in: EWNLG-93, Proceedings of the 4th European Workshop on Natural Language Generation, Pisa, Italy 1993, also in Trends in Natural Language Generation: an Artificial Intelligence Perspective, Springer Verlag Lecture notes series, (forthcoming 1995).

Aggregation in Natural Language Generation

HERCULES DALIANIS [1] and EDUARD HOVY [2]

1) Department of Computer and Systems Sciences, The Royal Institute of Technology and Stockholm University, Electrum 230, S-164 40 Kista, SWEDEN, ph. (+46) 8 668 90 98, E-mail: hercules@dsv.su.se

2) USC/Information Sciences Institute, 4676 Admiralty Way, Marina Del Rey, CA 90292-6695, USA, ph. (+1) 310-822-1511, fax (+1) 310-823-6714, E-mail: hovy@isi.edu

Abstract

In this paper we address the problem of redundancy in text generation. 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. Such redundant material is invariably removed by people, and ought to be removed by generator systems, to produce better quality text. We call the process of removing redundancy *aggregation*. In addressing the problem, three questions arise: Why do people object to redundancy? Which redundant portions are best removed? What mechanisms or rules are used to remove redundant information? In this paper we begin to answer the third question by identifying and describing the aggregation processes generators can use. We first survey the studies we have found on aspects of aggregation. We next outline a study we performed with human subjects. Finally, we define and describe eight aggregation strategies we identified, and discuss several associated issues and open questions.

1. Introduction: The Problem Of Redundancy

When speaking or writing discourse, people display a remarkable ability to avoid duplicating information they have already presented, or including information that is directly inferable from what they have said. We do this so naturally that the magnitude of the problem of suppressing redundancy has not really been adequately discussed in the literature. In this paper we try to present a systematic study of the problem.

Whether the knowledge is stored in a mind or a computer, it is unlikely to be stored optimally for every possible linguistic formulation. In data bases and knowledge bases, information is often represented in such a detailed and repeated way that some duplication is unavoidable. For example, in answer to a data base query: *Who is currently at ISI?*, a text generated straightforwardly from the alphabetised data may look like this:

> *Yigal Arens is an employee at ISI. Hercules Dalianis is a visitor at ISI. Eduard Hovy is an employee at ISI. Kevin Knight is an employee at ISI. Vibhu Mittal is a student at ISI. Richard Whitney is an employee at ISI.*

whereas, of course, it should look like this:

> *At ISI, Yigal Arens, Eduard Hovy, Kevin Knight, and Richard Whitney are all employees. Vibhu Mittal is a student, and Hercules Dalianis is a visitor.*

The second text, which is more compact, more understandable, and much less irritating, has undergone a process we call *aggregation.*

Addressing the problem of redundancy, three questions must be answered:

1. why perform aggregation?
2. perform aggregation when? (i.e., aggregate what items?)
3. perform aggregation how?

To the first question, as to so many questions about motivating generators' decisions, a stylistic/pragmatic answer seems appropriate. People and systems must perform aggregation to make their text more readable, understandable, and fluid; not doing so risks the reader's misunderstanding or irritation. We offer no formal account of the reasoning at this level, since we do not really know why, for example, people find unaggregated text so irritating. (After all, shouldn't input that is more explicit and complete facilitate understanding processes, and hence be preferable?) This fascinating question has to our knowledge never been addressed satisfactorily; discussions of the topic are usually grounded on Grice's maxims, (Grice 1975), which of course are not answers but just a set of observations. In this paper, we simply assume axiomatically that shorter and less redundant text is better text.

To the second question, when to perform aggregation, we refer to the work of Horacek, (1992). Like Horacek, we believe that aggregation is called for whenever parts of the information to be presented are either directly redundant with one another or are easily inferable from one another. In order to determine such redundancy or inferability, it is of course necessary to understand what the facts reader will infer on reading some piece of information stated in some particular way. Horacek calls such communication-driven inferences in general Conversational Implicature. He divides the inferences into three classes: those drawn by the author, those drawn by the reader from the reader's own knowledge, and those licensed by the communicative context. In this paper we do not offer a discussion of Conversational Implicature, since it is a very complex topic, involving questions of general inferential ability and memory organization. Instead, we assume the simplest model, namely, that *any* information that is explicitly repeated is redundant and ought to be removed in some way.

In this paper, we concentrate on the third question: *how to perform aggregation* to avoid redundancy. We discuss only the case where the generator avoids redundancy, and not the case where it completely omits information designated for communication by some other process (such as a text planner); the problem of when actually to drop information altogether introduces complexities we prefer not to address at this point. Our problem is simply to understand how information can be repackaged so as to remove redundancy while still allowing the reader to understand it fully and correctly.

In a simplified linearized model of the generation process, aggregation takes place after text planning (that is, after the content has been selected and preliminarily organized into a discourse structure) and before realization. During this intermediate stage, which we call Sentence Planning after Rambow (1990), several tasks are performed (see Hovy (1992)), including aggregation, theme and focus selection, some lexical selection, and pronominalization.

The problem of redundancy has been addressed from several perspectives in language generation research. First, we discuss some prior studies of the problem. Next, we outline a study we performed to see how and when people aggregate information. We then present our findings, a set of eight types of aggregation operations. Finally, we discuss several open questions surrounding the problem.

2. Previous Work: Some Approaches

Past research indicates that aggregation is not a unitary, simple process. Rather, we believe that aggregation rules of various kinds can apply at various times to various internal data structures, such as to the text structure or to the actual text content.

Several studies on aggregating text based on text structure appear in the literature. In fact, the term *aggregation* was first used in Mann and Moore (1980). Dale (1990) calls it discourse level optimization and Dalianis (1992b) compacting. Kempen (1991) calls it forward and backward conjunction reduction. In Horacek (1992), the most sophisticated study of aggregation we have found so far, Horacek describes the integration of aggregation (which he calls grouping) with quantification under guidance of principles of conversational implicature. One form of Conversational Implicature is the semantic aggregation performed in Chapter 3 of (Hovy 1988), namely the processes required in order to aggregate the information straightforwardly rendered as:

> *First, Jim bumped Mike once, hurting him. Then Mike hit Jim, hurting him. Then Jim hit Mike once, knocking him down. Then Mike hit Jim several times, knocking him down. Then Jim slapped Mike several times, hurting him. Then Mike stabbed Jim. As a result, Jim died.*

into, using various aggregation rules, any of:

> *Jim died in a fight with Mike.*
>
> *After Jim bumped into Mike once, they fought, and eventually Mike killed Jim.*
>
> *After Jim bumped into Mike once, they fought, and eventually he was knocked down by Mike. He slapped Mike several times. Then Mike stabbed Jim, and Jim died.*

Hovy defined inference rules containing patterns that matched against the input information and instantiated the appropriate inferences. In the first example above, the inference rule's left hand side pattern matched any series of actions defined as hurtful, while its right hand side instantiated in the system's knowledge base a single new concept, namely a fighting event. Instead of generating each hurtful action, the generator then simply generated the newly created fight concept. This rule can be summarized as:

(A hit B. B hit A. A hit B...} => {A and B fought)

Though this technique works, it is very sensitive to the exact form and content of the knowledge base, and consequently difficult to generalise.

In later work, Hovy (1990), two structural aggregation rules were used to eliminate redundant information. Rather than compare all pieces of information to be communicated to find possible redundancies (an n-squared operation in the number of data terms), these two structural rules operate within a text structure to compare terms locally (making the number of comparisons sublinear in n). The text structure was created by Hovy's text structurer using relations from Rhetorical Structure Theory (RST), Mann and Thompson (1988).

In an example in Scott and de Souza (1990), nine heuristic rules aggregate six sentences which express a set of facts using a single sentence. The heuristics are very general and unfortunately do not give enough information on how to aggregate in detail. In Dale (1990) redundancies in recipes are recognized and avoided by applying discourse level optimization on the text structure before realization.

Dalianis (1992b) uses compacting (an early version of the techniques developed here) to avoid generating redundancies.

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 1a. Original unaggregated text.

The natural language expression has one redundant noun phrase "an idle subscriber t1"; (see Figure 1a). The compacting module applies the following rule: If two or more identical noun phrases are repeated consecutively, then remove all except the first one. This operation makes the text more concise (see Figure 1b).

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

Figure 1b. Aggregated text after compacting.

However, the compacting rule is too weak to apply also to the single noun "phonenumber", which occurs three times. In this paper we address such additional aggregation problems.

3. The Study

In order to gain a better understanding of the kinds of aggregations people perform, we carried out a study with people, using an example collection of facts. This section briefly describes the domain on which the study was carried out, the study itself, and our results.

3.1. The Telephone Domain

The context for our study is provided by the LOXY-NL generator Dalianis (1992b). In this domain, the generator produces texts about various configurations of telephones and their users. The generator forms part of a system, the VIsual and Natural language Specification Tool (VINST), Engstedt and Preifelt (1992b), in which potential telephone system customers can define their desired networks, simulate the network operations, and use a theorem prover to check for potential conflicts; Echarti and Stålmarck (1988,1989). The input made to the system is in natural language and graphics, as is the output; Engstedt (1991), Engstedt and Preifelt (1992a).

The LOXY-NL generator used only the single compacting rule mentioned above, although we observed that the generated LOXY text could be further aggregated.

An example in the telephone domain was represented using part of the LOXY formalism (see Figure 2), in which t1, t2, t3 are telephone subscribers with telephone numbers, hot numbers, and states of being idle or busy, etc.

Example telephone domain representation

has(t1,hotnumber(200)) & has(t1,phonenumber(101)) &
has(t1,phonenumber(100)) & subscriber(t1) & idle(t1) &
has(t2,phonenumber(200)) & subscriber(t2) & idle(t2) &
has(t3,phonenumber(300)) & subscriber(t3) & idle(t3) &
speechconnection(sp1) & idle(sp1)

Figure 2. Example LOXY-NL input data propositions in the telephone domain.

The input data is unordered and therefore needs some sort of structuring to make obvious the relations between the objects. Hence we manually put together a semantic network representing their internal relations, to illustrate the relationships among unordered input data.

3.2. The Questionnaire

In order to identify the aggregation rules that people use to build LOXY texts, we created a questionnaire containing the propositional inputs to the LOXY-NL generator as in Figure 2, together with definitions of the various terms used (see Figure 3).

subscriber	means a person who has a phonenumber or a hotnumber.
t1,t2,t3	are instances of subscribers.
has	is a relation between an object and an object/attribute.
dialtone	is an attribute which a subscriber may have.
phonenumber	is a number which a subscriber can call.
hotnumber	is a number to which a subscriber is automatically connected without dialing, upon lifting the receiver.
100, 200,..	are numbers and instances of phonenumbers, hotnumbers etc.
speechconnection	is a physical connection between different subscribers.
sp1	is an instance of a speechconnection.
idle, busy	are different states which t1,t2,sp1......, can have.
&	is the logical "and"

Figure 3. Definitions in natural language of the input data symbols.

The questionnaire contained five example sets of input data and each example contained between 11 and 18 propositions such as those shown in Figure 2. The order of data propositions was scrambled in different questionnaires.

In the questionnaire, we asked the test subjects to create the most appropriate text(s) they could think of, for each of the five sets of input (more than one text per set of input data was acceptable). Twelve out of fifteen subjects (all computer scientists working at ISI) completed the questionnaire, giving a set of 12 paragraphs of text.

3.3. Analysis

In order to identify the various possible aggregations, the authors represented the content of the propositions in Figure 2 in a semantic network (see Figure 4).

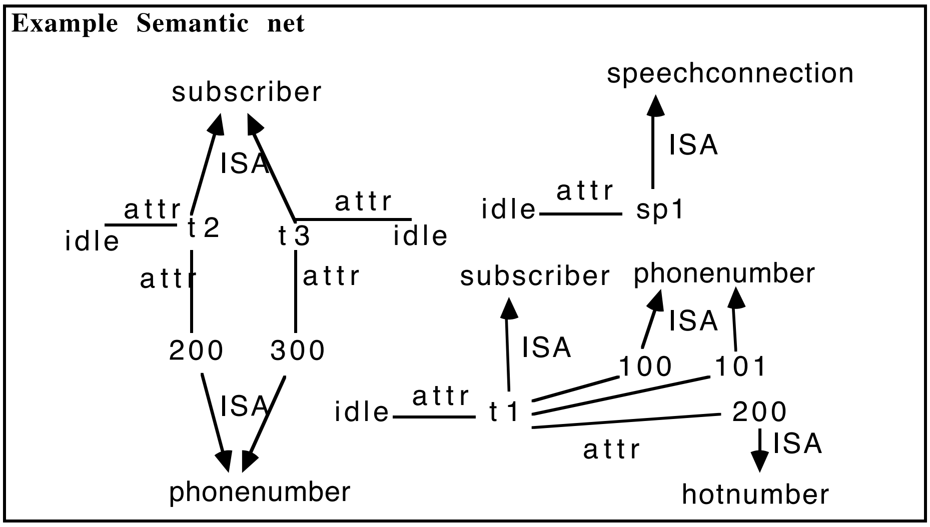


Figure 4. Semantic network of input data propositions of Figure 2.

Figure 4 contains three independent groups, some sharing elements (for example, *t1*, *t2* and *t3* share the elements *idle* and *subscriber*) that can act as aggregation points. However, for purposes of paragraph structuring, we found the semantic networks not to be the most suitable representation. We therefore manually created for each example an RST text structure (see Figure 5); given the semantic simplicity of the texts, we did not encounter any difficulty of multiple analyses.

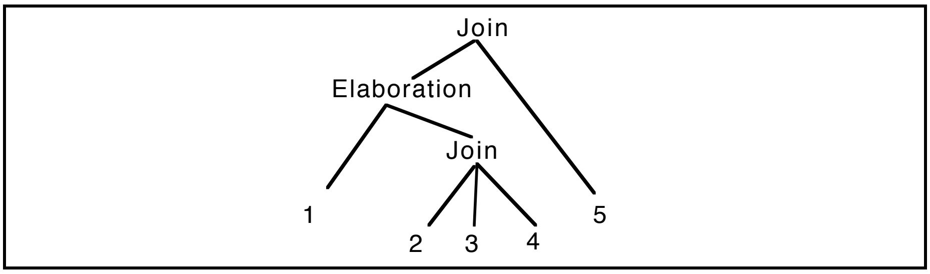


Figure 5. Example LOXY-domain RST text structure, which is better for presentation purposes than the semantic network.

We assume here that the subjects in our study all used some such text representation when creating their paragraphs. We also assume that they performed aggregation after their RST-like discourse structures were created.

After aggregation, in which the aggregation rules transform the discourse structure into a more streamlined structure, surface sentences are generated from each leaf. An aggregated text for the example appears in Figure 6. This is of course only one of the numerous possibilities.

1) t1, t2, t3 are all subscribers and are idle
2) t1 has the phonenumbers 101 and 100 and a hotnumber 200
3) t2 has the phonenumber 200
4) t3 has the phonenumber 300
5) sp1 is a speechconnection and is idle

Figure 6. Example aggregated LOXY-NL text.

It is easy enough to create specific aggregation rules for any specific example. We found that we ended up with more aggregation rules than could apply to any single example. Also, we noticed that the rules seemed to be of various types, and to apply to various parts of the underlying data structures. We therefore analyzed the subjects' texts to determine how many possible aggregation rules people used within the very small and constrained set of possibilities of the LOXY domain.

For each completed questionnaire, we first analyzed each text produced, creating the corresponding text structure tree. We then compared the unaggregated networks of input concepts drawn, by us, as semantic networks with the corresponding aggregated text structures. (It was interesting that three of the subjects drew on the sides of the questionnaire their own semantic network).

On analysis, four classes or types of aggregation rules became obvious:

1) Grouping and collapsing rules

2) Ordering rules

3) Casting rules

4) Parsimony rules

Some of the rules apply strictly within a clause (and these we call internal rules); some strictly across clauses (i.e., within the discourse structure; called external rules), and some apply to both cases. Obviously, external rules have to apply before internal rules can. In the next section we describe these rules in more detail.

Of the 12 subjects, 9 performed aggregation in their texts. Of the remaining three, one did not understand the questionnaire, another generalized the text and did not mention instances at all, and the third did not do any aggregation (or any other kind of sentence planning), except clustering, and then simply mechanically turned each clause into English.

In Table 1 we tabulate the results of the 9 subjects' texts made of the five input propositions in the questionnaire. The table names the aggregation rules we found. The various types of aggregation rule are defined in Section 4, following immediately below. It is clear from the table that most people performed some of each type of aggregation.

	Group.			Order.			Cast.	Parsim.	
Proposition	Sub.	Pred.	Cue	Int.	Ext.	Clust.	Cast	Econ.	Rep.
Input data 1	5	5	4	6	5	1	7	5	6
Input data 2	10	4	2	9	7	7	11	6	0
Input data 3	9	0	0	7	4	3	6	5	4
Input data 4	1	7	0	7	6	3	4	5	5
Input data 5	5	5	2	6	5	1	7	5	6

Table 1. Number of times each aggregation rule is used for each input proposition by subjects creating text.

4. Four Types Of Aggregation Rules

As mentioned above, we identified four types of aggregation rules: Grouping, Ordering, Casting and Parsimony rules. Grouping and Ordering rules can be applied in any order; but of course give different results; Casting rules apply when the other two are finished. The Parsimony rules contain the Economy rules which can be applied whenever, and the Repetition rules, which apply early during content selection.

4.1. Grouping Rules

Grouping, the principal aggregation operation, implies collecting clauses with common elements and then collapsing the common elements. The common elements in our study are either subjects or predicates.

RULE 1: SUBJECT GROUPING

DEF: Subject Grouping

Two or more propositions with identical subjects are aggregated to form a single proposition with a compound predicate.

Formula: ab + ac...an => a(b+c...n)

Horacek (1992) calls this *grouping motivated by structural reasons*; *embedding* is the term used by Scott and de Souza (1990), Dalianis (1992b) calls it *compacting*, and Kempen (1991) *forward conjunction reduction.*

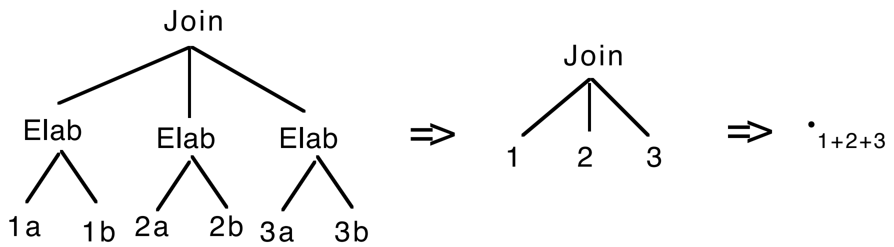


Figure 7. Operation on discourse tree by the Subject Grouping rule applied twice.

An example; before aggregation:

1) a) t1 is an idle subscriber, b) t1 has a phonenumber 100.

2) a) t1 is an idle subscriber, b) t1 has a hotnumber 200.

3) a) t1 is an idle subscriber, b) t1 has a forwardnumber 300.

After the subject grouping rule is applied once the Elaboration relation disappears:

1) t1 is an idle subscriber with a phonenumber 100.

2) t1 is an idle subscriber with a hotnumber 200.

3) t1 is an idle subscriber with a forwardnumber 300.

And finally the Join relation disappears:

1+2+3) t1 is an idle subscriber with a phonenumber 100, a hotnumber 200, and a forwardnumber 300

This aggregation rule applies externally to clauses and merges separate clauses into a single clause, thus producing results internally. The Join relation is pushed "down" into the clause to form a compound nominal group. This transformation is shown in Figure 7. Incidentally, the above example illustrates the inadequacy of limiting RST analyses to full clauses.

RULE 2: PREDICATE GROUPING

DEF: Predicate Grouping
Two or more propositions with identical predicates are aggregated to form a single proposition with a compound subject.

Formula: an + bn...mn => (a+b...m)n

An example of this type of aggregation also appears in Hovy (1990). Kempen (1991) uses a similar grouping on objects, which he calls *backward conjunction reduction*.

An example; before aggregation:

1) t1 is a subscriber
2) t2 is a subscriber

and after aggregation with the predicate grouping rule:
1+2) t1 and t2 are subscribers

Just as for the Subject Grouping rule, the Join relation is moved from the discourse structure to a compound nominal group within the clause.

Obviously, in a more complex domain, the Predicate grouping rule can be reformulated to handle the various subparts of predicates: direct and indirect objects, etc. The question becomes complex, however, since it gets involved with lexical choice and focus. For example, *John gave Mary a book and Tom gave Mary a pen*, can be grouped as: *John and Tom gave Mary a book and pen respectively*, but possibly better as: *Mary received a book from John and a pen from Tom.*

4.2. Ordering Rules

DEF: Ordering

Before aggregation, input propositions are ordered based on the characteristics of their subjects. The order given below holds for the domain investigated in this study, but is probably characteristic of the order in most domains.

Clearly, the aggregation of information and the order of presentation of information are closely related; one can only aggregate two clauses sharing information if the clauses are adjacent. In the study we found that people happily reordered the input propositions as needed to facilitate aggregation.

Ordering rules may only affect the ordering of clauses within free-order zones of the text. A free-order zone is created either externally (among clauses) under the RST relation Join (sometimes called List), in which the various clauses are all at the same level of rhetorical generality and importance and hence are free to be ordered on other grounds, or else internally (in a clause) with a compound noun phrase, in which the nouns can be freely scrambled.

Without any exception, whether the questionnaire contained the data terms in one order or another, we found that subjects ordered the terms within a Join zone based on the propositions' subjects, according to the following priorities:

state-change > animate > inanimate > concept-supertype (isa) >
> attribute > much information > less information

where:

state-change	object changes state, e.g., from idle to busy
animate object	e.g., a subscriber
inanimate object	e.g., a speech connection
supertype	e.g., is a subscriber
attributes	e.g., is idle, has a phonenumber 100

RULE 3: INTERNAL ORDERING

DEF: Internal Ordering

Ordering applies within a sentence component such as an noun phrase.

For example:

supertype > attribute : sp1 is a speechconnection and is idle.

RULE 4: EXTERNAL ORDERING

DEF: External Ordering

External ordering applies to clauses in the discourse structure

For example:

animate > inanimate : t1 is a subscriber. sp1 is a speechconnection

The external order used between Supertype and Attribute relations was also used by Dalianis (1992a), with the discourse relation Elaboration.

RULE 5: CLUSTERING

DEF: Clustering

Clustering occurs after Internal and External Ordering, among propositions sharing a free-order zone, so that propositions about the same subject are adjacent.

The Ordering rules above do not impose a unique order upon propositions. Within an ordering position, people in the study tended to cluster together all propositions dealing with the same material. For example, when all Attribute propositions are ordered together, they may appear in this order:

t1 is idle
t2 is busy
t1 has a phonenumber 100
t2 has a phonenumber 200

After clustering, however, they appear in this order:

t1 is idle
t1 has a phonenumber 100
t2 is busy
t2 has a phonenumber 200

Both examples satisfy the internal and external ordering rules. However, in the second example, propositions about t1 are clustered together and about t2 as well. Obviously, such clustering enables further aggregation.

4.3. Casting Rules

Casting rules govern the choice of realization of propositions as parts of speech; an action or event such as SING, for example, may be cast as a verb, *to sing*, or nominalized as a noun, *the singing*.

RULE 6: CASTING

DEF: Casting

The same syntactic constructions and lexical items are used for semantically similar items throughout the whole discourse.

In texts, the syntactic phrasing of sentences expressing similar semantic information is usually constant: When a certain sentence structure is used in the first sentence then for similar expressions the same verb and argument order is used in the next sentence (internal order).

For example, compare:

t1 is a subscriber and is idle.

sp1 is a speechconnection and is idle.

and

t1 is *a subscriber and is idle*.

sp1 is *an idle speechconnection*.

The first pair of sentences conforms to the casting rule, while the second does not. In the second pair of sentences, the first sentence and the second sentence have different syntactic forms, and hence different casting.

We found this rule one of the most striking and consistent in our study (seeTable 1). Since, by the Precedence Ordering rule, state changes were always generated first, the syntactic casting affected subsequent sentences:

sp1 is busy and is a speechconnection.

t1 is idle and is a subscriber.

or

sp1 is a busy speechconnection.

t1 is an idle subscriber.

In longer text, we surmise that the regularity of syntactic expression will become tedious. In this case, and for this reason, we believe people may choose instead to vary the syntactic form, or to make tables that express the same information in condensed form. Tables, for example, embody a type of aggregation that goes beyond the textual medium.

4.4. Parsimony Rules

The Economy and Repetition rules are called Parsimony rules because they determine the verbosity of the discourse.

RULE 7: ECONOMY

DEF: Economy
Prefer the casting of a proposition that has the fewest elements in its sentence matrix. Do not allow more than three elements in any single sentence component.

This rule can be paraphrased as: keep the text short, but not too short. The rule operates in conjunction with the Subject and Predicate Grouping rules.

a subscriber t1 has a phonenumber 100

is better than

t1 is a subscriber and has a phonenumber 100

RULE 8: REPETITION

<u>DEF: Repetition</u>

In highly structured texts, elide sentence-length propositions that repeat information already presented.

In longer texts it is occasionally necessary for whole groups of propositions to be repeated. Such repetitions occurs for reasons of smooth theme development. However, in shorter or highly structured text direct repetitions may be avoided. In our study, for example, portions of the five input proposition sets were identical and we found that most subjects simply omitted groups of identical propositions from their texts.

4.5. Examples

In this section we present two examples with all rules applied.

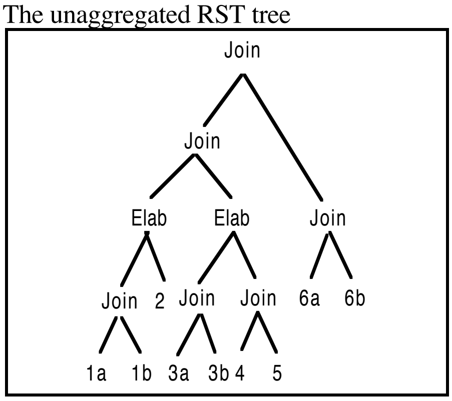
Example 1.

We start with the initial information:

subscriber(t1) & subscriber(t2) & speechconnection(sp1) &
idle(t1) & idle(t2) & idle(sp1) &
has(t1,phonenumber(100)) & has(t2,phonenumber(200)) &
has(t2,hotnumber(300))

Figure 8. Example input data propositions.

The unaggregated RST tree and the corresponding unaggregated text



1a) t1 is a subscriber b) t1 is idle
2) t1 has a phonenumber 100
3a) t2 is a subscriber b) t2 is idle
4) t2 has a phonenumber 200
5) t2 has a hotnumber 300
6 a) sp1 is a speechconnection
b) sp1 is idle

Figure 9. Redundant and unaggregated RST text structure and corresponding text.

Assume the above redundant and unaggregated RST text structure (Figure 9) was created by the text planner from the inputa data propositions in Figure 8. The following rules trigger to create the non-redundant and aggregated RST text structure in Figure 10:

Grouping the common elements together:

Subject grouping; in sentences 1a+1b, 3a + 3b, 4+5, 6a + 6b

Predicate grouping; in sentences 1+3

Precedence Order (internal and external):

Internal order; isa > attribute, in sentences (1+3) and 6

External order; isa > attribute, between sentences (1+3) > (4+5), 2

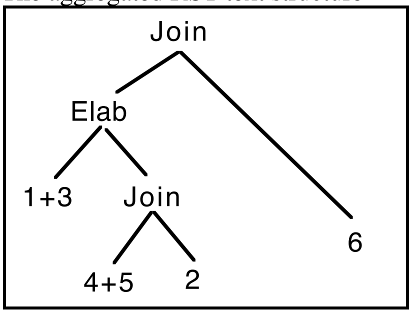
External order; much information > less information
between sentences (4+5) > 2

External order; between animate sentences 1+3, 4+5, 2 >
non-animate sentence 6

Casting rule:

on sentences (1+3), 6 and on sentences (4+5), 2

The aggregated RST text structure and the corresponding aggregated text



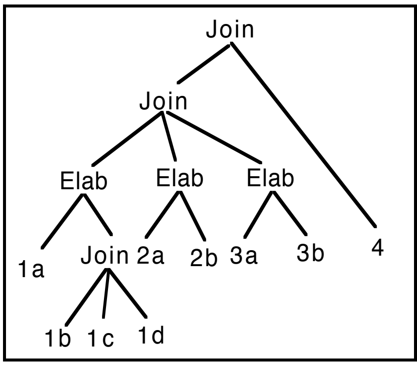
(1+3). t2 and t1 are subscribers and are idle.
(4+5). t2 has a phonenumber 200 and a hotnumber 300
2). t1 has a phonenumber 100.
6). sp1 is a speechconnection and is idle.

Figure 10. RST text structure and corresponding aggregated text.

Example 2.

Application of the aggregation rules in different order results in different texts. Observe the examples in Figures 5 and 6, 9 and 10, where the order of the aggregation rules are grouping, ordering and casting.

In next example we switch the order of the aggregation rules to ordering, grouping and casting, and obtain the discourse tree and text in Figure 12.



1a) t1 is an idle subscriber
1b) t1 has a phonenumber 101
1c) t1 has a phonenumber 100
1d) t1 has a hotnumber 200
2a) t2 is an idle subscriber
2b) t2 has a phonenumber 200
3a) t3 is an idle subscriber
3b) t3 has the phonenumber 300
4) sp1 is an idle speechconnection

Figure 11. Redundant, unaggregated RST text structure and corresponding text.

Assume the above redundant and unaggregated RST text structure (Figure 11) was created by the text planner. The following rules trigger to create the non-redundant and aggregated RST text structure in Figure 12.

Precedence Order (internal and external) and clustering

Internal order; isa > attribute in sentences 1,2 and 3

External order; much information > less information, between sentences 1 > 2, 3

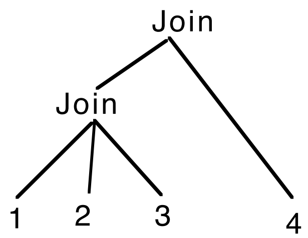
External order; between animate sentences 1,2,3 and non-animate sentence 4

Grouping the common elements together

Subject grouping: in sentences 1a+1b+1c+1d, 2a+2b, 3a+3b

Casting rule

on sentences 1,2,3,4



1) t1 is an idle subscriber and has the phonenumbers 101 and 100 and a hotnumber 200

2) t2 is an idle subscriber and has a phonenumber 200

3) t3 is an idle subscriber and has a phonenumber 300

4) sp1 is an idle speechconnection

Figure 12. Aggregated non-redundant RST text structure and corresponding text.

This text was created by applying aggregation rules in the following order: ordering, grouping and casting.

5. Cue Words Marking Aggregation

Authors often signal the fact that aggregation has taken place by including appropriate cue words. Cue words signal which kind of aggregation has occurred, thereby assisting the reader's comprehension. As shown in Table 1, these cue words appeared more often when the more aggregation occurred.

Typical cue words are:

- *respectively* : signals distribution of predicate parts over subjects, as in *James and Susan did woodwork and metalwork respectively.*
- *as well* and *also* : signal aggregation possibilities that were not taken for some reason, possibly on being blocked by another aggregation, as in *Karin wore a blue dress and a purple hat. Mary and Yolanda wore purple hats as well.*
- *both* and *all* : signal explicitly that aggregation has occurred, as in *Gerry and Gary have both eaten ice cream.*

6. Conclusion

In this paper we outline a preliminary study and propose an initial set of eight aggregation rules. A great deal of careful work and refinement must still be done before the particularities of aggregation and its relationship to the other text and sentence planning tasks are understood. In addition, some careful thought is required on the topics of why aggregation is performed and when (i.e., on which information) it is appropriate.

A follow-up study to this one would involve a larger group of people not all of whom are computer scientists for more general results. It would use a larger and more complicated domain with more relationships. The input data propositions would be selected to include more RST relations than just Elaboration and Join. The input data propositions should again be scrambled differently for each subject (more so than we did in this study). The subjects should focus on various objects for more controlled generation, and should be asked to draw the networks and trees they use to generate their texts.

An interesting question is what phenomena can block or affect aggregation. Thematization is one, since it involves reordering parts of clauses. A second is temporal change, as illustrated by

The wall is red. Tom paints the wall. The wall is yellow.

which cannot be aggregated into

The wall is red and yellow. Tom paints the wall.

Our traditional use of RST only at the clause level and above will have to be extended to relate also parts of clauses, as when for example the Elaboration relation in a discourse structure disappears into a single syntactic unit on application of the subject grouping rule (see Figure 7).

The nature and usage of cue words and phrases to signal aggregation and to delimit the scope of aggregated units is a topic that deserves considerable further study, from the perspectives of both sentence generation and linguistics.

In summary, we are satisfied with the outcomes of the study. Building as it does upon the initial work of Dalianis (1992b), we have gained a richer and clearer understanding of the problem of aggregative processes that help remove redundancy from text. We also have a clear set of goals for future research into the problem.

Acknowledgments

Many thanks to all the study participants, as well as to Vibhu Mittal, Cécile Paris, Akitoshi Okumura, Licheng Zeng, and several anonymous reviewers.

Many thanks also to Måns Engstedt and Stefan Preifelt at the Ellemtel Telecommunication Systems Laboratories for interesting discussions which contributed to this paper.

This research was carried out during a visit by the first author to the Information Sciences Institute. The visit was sponsered by a research grant from the Royal Institute of Technology, Stockholm, Sweden.

7. References

Dalianis, H. (1992a). A method for validating a conceptual model by natural language discourse generation. In *CAISE-92 International Conference on Advanced Information Systems Engineering,* Loucopoulos P. (ed.). Springer Verlag Lecture Notes in Computer Science no. 593 (425-444).

Dalianis, H. (1992b). A natural language generation system for validating specifications for telephone services. Technical report, Ellemtel Telecom Labs, Älvsjö, Sweden, and The Royal Institute of Technology, Stockholm University.

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

Echarti, J-P. and Stålmarck, G. (1988). A logical framework for specifying discrete dynamic systems. Technical report, Ellemtel Telecom Labs, Älvsjö, Sweden.

Echarti, J-P. and Stålmarck, G. (1989). Towards a logical framework for modelling and validating the behaviour of telecommunication services. Technical report, Ellemtel Telecom Labs, Älvsjö, Sweden.

Engstedt, M. (1991). *A fFexible Specification Language Using Natural Language and Graphics*. Master's Thesis. The Centre of Cognive Science, University of Edinburgh.

Engstedt, M. and Preifelt, S. (1992a). Results from the user tests of VINST. Technical report EUA F92 0864, Ellemtel Telecom labs, Älvsjö, Sweden.

Engstedt, M. and Preifelt, S. (1992b). A System Description of the VINST prototype. Technical report EUA F92 22 75, Ellemtel Telecom Labs, Älvsjö, Sweden.

Grice, H.P. (1975). Logic and Conversation. In *Syntax and Semantics 3, Speech Acts,* Cole, P. and Morgan, J.L. (eds.). New York: Academic Press (41-58).

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

Hovy, E.H. (1988). *Generating Natural Language under Pragmatic Constraints*. Hillsdale, New Jersey: Lawrence Erlbaum Associates Publishers.

Hovy, E.H. (1990). Unresolved issues in paragraph planning. In *Current Research in Natural Language Generation*, R. Dale et al. (eds.). Academic Press Limited (17-45).

Hovy, E.H. (1992). Sentence Planning requirements for automated explanation generation. In *Proceedings of the Workshop on Explanation Facilities for Model-Based Expert Systems,* DIAMOD-Bericht no. 23. GMD St. Augustin, Germany.

Kempen, G. (1991). Conjunction reduction and gapping in clause-level coordination: An inheritance-based approach. *Computational Intelligence 7(4)* (357-360).

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

Mann, W.C and Thompson, S.A. (1988). Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. *TEXT 8(3)* (243-281).

Rambow, O. (1990). Domain Communication Knowledge. In *Proceedings of the 5th International Workshop on Generation*. Pittsburgh, PA (87-95).

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