Generating Texts with Style

Richard Power[1], Donia Scott[1], and Nadjet Bouayad-Agha[2]

[1] Information Technology Research Institute, University of Brighton
Lewes Road, Brighton BN2 4AT, UK
{richard.power,donia.scott}@itri.bton.ac.uk
[2] Departament de Tecnologia, University Pompeu Fabra
Passeig de Circumval-lació 8, Barcelona, Spain
Nadjet.Bouayad@tecm.upf.es

Abstract. We describe an approach for generating a wide variety of texts expressing the same content. By treating stylistic features as constraints on the output of a text planner, we explore the interaction between various stylistic features (from punctuation and layout to pronominal reference and discourse structure) and their impact on the form of the resulting text.

1 Introduction

Any reasonably complex message, comprising perhaps ten propositions, can be expressed in billions of ways in English or any other natural language. An important challenge for computational linguists is to understand the contextual and stylistic factors that make one version preferable to another. In the final chapter of 'Elements of Style' [1], E.B. White posed this problem by contrasting a memorable quotation from Thomas Paine:

These are the times that try men's souls.

with several alternative formulations of the same literal content, including:

Soulwise, these are trying times.

As White points out, recognising the absurdity of the second formulation is easy; explaining exactly why it is absurd is altogether harder. We describe in this paper an approach that addresses some aspects of stylistic variation, particularly those that apply at the structural level. This work is carried out within the context of natural language generation (NLG).

Most NLG systems are restricted to one style of text. Moreover, with the notable exception of Hovy's PAULINE, existing NLG systems cannot reason about the style(s) they produce; such decisions are hard-wired within the system [2]. Since style impacts on all aspects of text — from syntax and lexical choice through to discourse structure and layout — changing the style of the output texts of such systems can have far reaching implementational consequences. The approach we describe here achieves a wide variety of styles in a flexible and

efficient manner, at least at the level of text planning. Where appropriate, it also gives control over stylistic choices to the user.

In describing a text as being 'good', one is really addressing two distinct classes of criteria. The first concerns correctness: does the text conform to the rules of the language? In other words, is it grammatical, is the punctuation correct, are the correct words chosen, and so forth. Clearly, one would rule out any text which does not satisfy such criteria. However, for a given content there will be a wide variety of correct texts, their number increasing exponentially with the size of the content (i.e., number of propositions). The second class of criteria relates to the suitability of the selected text compared with alternative 'correct' solutions.

The first criterion accounts for 'stylistic' rules that are applied to *any* situation of communication and whose violation hinders seriously the quality of the text. The second criterion brings us back to E.B. White's point: to what extent is a particular version of a text appropriate to the given situation of communication — e.g., the genre, the register, the idiosyncratic style of the author or the house style of the publisher?

In what follows, we provide a general description of our approach to the implementation of style at the level of text planning, taking into account the different types of stylistic rules. We exemplify our approach through a working example showing the operation of stylistic settings on the production of a text.

2 Hard and soft constraints

We treat stylistic rules as constraints and distinguish between the two classes of stylistic criteria through their classification as *hard or soft constraints*.

A hard constraint is a fatal defect; texts violating hard constraints should never be generated, regardless of their other merits. Among hard constraints we include correct structure, both as regards syntax and higher textual levels, and correct realization of rhetorical relationships. The system currently imposes about 20 such constraints during text planning, including the following:

- Spans linked by a subordinating conjunction (e.g., *since*) must occur within the same text-clause.
- The text-category hierarchy [3] must be respected: for instance, a text-sentence cannot contain a paragraph (unless the paragraph is indented).
- Vertical lists cannot be used for the arguments of a nucleus-satellite relation; the relation must be multinuclear.

Since soft constraints are non-fatal defects, there may be circumstances in which violations are accepted as a necessary evil. Many stylistic dilemmas arise because one soft constraint conflicts with another. For example, we may prefer to avoid the passive voice, but we also prefer to place the current discourse topic in subject position. How can such a conflict be resolved? Following Optimality Theory [4], the constraints could be arranged in an order of priority, but this leads to absurd anomalies: a single violation of one constraint might outweigh

a hundred violations of constraints with lower rank. The alternative, which we adopt, is to compute for each solution a cost, weighting violations according to their relative importance. Style preferences defined by the user make their presence felt by modifying some of these weights.

For efficiency, it is obviously desirable that solutions violating hard constraints are eliminated *before* any candidates are generated. We have shown elsewhere [5] that this can be done by formulating text planning as a Constraint Satisfaction Problem, which can be solved by Constraint Logic Programming [6]. Briefly, the idea is that the features over which hard constraints are defined are represented by finite-domain variables, and that unacceptable combinations of values are ruled out before any specific solutions are generated.

To evaluate violations of soft constraints, we can find no sufficiently flexible alternative to a generate-and-test method. Several hundred candidate text plans are generated; each is assigned a cost by summing weighted scores for each violation; the solution with lowest cost is selected, and passed forward for tactical generation and formatting.

The obvious drawback to generate-and-test is that the number of candidates increases exponentially with complexity (roughly, with the number of elementary propositions in the semantic input). From informal experiments, we find that the number of candidate solutions is around 5^{N-1} for N elementary propositions, which means that even for a short passage containing a dozen propositions the text planner would find about 50 million solutions satisfying the hard constraints. One might try to address this problem by a statistical optimization method such as a genetic algorithm [7], but we think a more natural and informative method is to break up the problem into parts, so that at each stage only a manageable part of the total solution is constructed. For instance, when planning a text, the semantic material could first be distributed among sections, then perhaps among paragraphs, thus spawning many small-scale text-planning problems for which the search spaces would be measured in hundreds rather than billions.

We have followed this approach also in combining text planning with tactical generation. Once the best text plan has been selected, it remains fixed, no matter what further costs accrue during syntactic realization. Moreover, the realization of each proposition in the linear sequence is fixed before passing to the next proposition. The tactical generator thus explores various ways of realizing the first proposition, evalutes them according to soft constraint violations, and chooses the best; the resulting linguistic context is then taken into account when searching for the best realization of the second proposition.

3 A working example

3.1 The ICONOCLAST system

We have developed a system, ICONOCLAST, which generates texts for the domain of Patient Information Leaflets (PILs) — the package inserts which explain ingredients, side-effects, instructions for using the medicine, and so forth.

The user of the system can define the *content* of a leaflet by using the WYSIWYM method [8], and can also vary the *style* of the generated text by sliding pointers along nine scales representing the following parameters:

Paragraph Length
Sentence Length
Frequency of Connectives
Frequency of Passive Voice
Frequency of Pronouns
Frequency of Semicolons
Frequency of Commas
Technical Level (use of technical terms)
Graphical Impact (use of vertical lists)

As examples of style profiles we have saved two configurations named 'Broadsheet' and 'Tabloid'[3]. The broadsheet style has long paragraphs and sentences, frequent use of passives, semicolons, and commas, relatively few pronouns, high technical level, and low graphical impact; the tabloid style is the reverse. With the broadsheet profile loaded, the output text for a short section of a PIL on 'Taking your medicine' might read as follows:

> To take a tablet, remove the tablet from the foil, and swallow it with water. If you take an overdose, tell your doctor immediately, or go to your hospital's Casualty Department.

The user might induce small changes in this text by using the sliders, for example by reducing the comma frequency or raising the pronoun frequency. Every time this is done, the system generates a new text from the current model and current style profile. Alternatively, the user might decide to change the style completely by loading the tabloid profile, with perhaps the following result:

> To take a tablet:
> 1 Remove it from the foil
> 2 Swallow it with water
> If you take an overdose
> - tell your doctor immediately, or
> - go to your hospital's Casualty Department.

3.2 Satisfying hard constraints

Figure 1 shows a simple model comprising two propositions linked by a 'cause' relation[4]. With standard settings for hard constraints, and two potential discourse connectives (*since*, *consequently*), our text planner generates eight candidate solutions for this input, including plans A and B in figure 2. After syntactic realization, the texts resulting from these plans were as follows:

[3] These labels should not be taken too seriously.
[4] Details of the semantic representation are presented in [9].

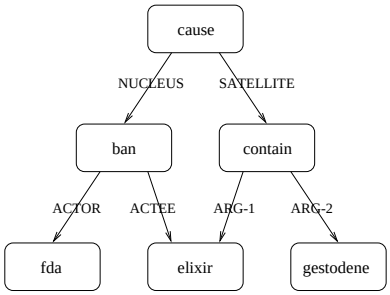

Fig. 1. Semantic Input

(A) Since Elixir contains gestodene, it is banned by the FDA.
(B) The FDA bans Elixir. It contains gestodene.

We will show later that under most settings of the style parameters, A is preferred to B; first, however, we say a little more about how these candidate solutions are obtained.

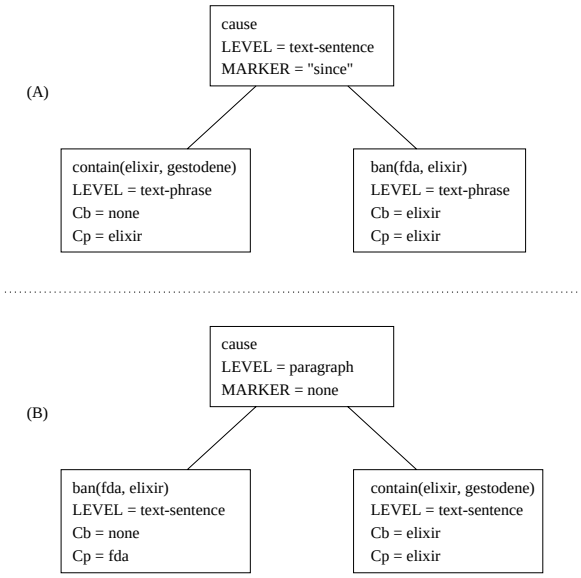

Fig. 2. Text Plans

The algorithm for producing plans like A and B, implemented in the Constraint Logic Programming language Eclipse [10], can be summarized as follows[5]:

1. A schematic plan is constructed by assigning one node to each rhetorical relation and to each elementary proposition. Since the semantic input in figure 1 has one rhetorical relation linking two propositions, we obtain a schematic plan with three nodes (figure 2).
2. Each node is assigned *text-category* variables [3]. A text-category comprises a LEVEL (e.g., section, paragraph, text-sentence) and an INDENTATION (relevant for texts with vertical lists). To simplify, we assume here that there are no indented constituents.
3. Each node (except the root) is assigned a POSITION variable representing its order in relation to its sisters. This variable is omitted from figure 2, where position is shown instead by left-to-right order on the page.
4. Nodes representing relations are assigned a MARKER variable whose value ranges over the relevant discourse markers (e.g., *since* and *consequently* for 'cause'), including an empty value (*none*) if the relation can be left implicit.
5. Nodes representing propositions are assigned Cb and Cp variables ranging over the potential backward and forward centers [11].
6. Constraints are defined over the solution variables so that ill-formed text plans cannot be generated.
7. All combinations of variables satisfying the hard constraints are enumerated; every such combination defines a candidate text plan to be submitted to further evaluation using soft constraints.

Note, incidentally, that the plans in figure 2 are still schematic: for instance, plan A needs to be elaborated so that the discourse connective *since* is coordinated with the first proposition.

3.3 Satisfying soft constraints

Having generated a set of candidate plans, including A and B, the program applies soft constraints in order to assign a cost to each plan. The cost is computed by checking for a series of violations at each node, penalizing each violation by a score that might depend on the current settings of the style parameters, and then summing over the whole plan. We list below some soft constraints that are currently implemented; note, however, that these constraints are provisional, and that so far we have no empirical basis either for choosing these particular constraints or for fixing their relative weights. The purpose of the list is to indicate some plausible stylistic constraints, and to show how they can be applied at the text-planning stage.

The first five constraints represent general principles of good style, unrelated to the style parameters controlled by the user.

[5] Details are available in [5].

Nucleus-satellite order
In most cases (including 'cause') the nucleus can be best emphasized by placing it in final position. A penalty is therefore imposed when the nucleus is placed first. *Plan B violates this constraint.*

Structure branching
Right-branching structures are preferred to left-branching ones.

Rhetorical grouping
Text plans that express rhetorical groupings explicitly are preferred to ones that leave them implicit. For instance, a plan which expressed three propositions by a paragraph of three sentences would be penalized if two of these propositions were grouped together in rhetorical structure.

Marker Repetition
If two rhetorical relations that are neighbours in the semantic input are expressed by the same discourse connective, a penalty is applied. This constraint would for example penalize a sentence in which *since* occurred twice.

Continuity of Reference
Three soft constraints based on centering theory are employed to score continuity of reference: they are salience ($Cp_N = Cb_N$), coherence ($Cb_N = Cb_{N+1}$) and cheapness ($Cp_N = Cb_{N+1}$) [12, 13]. *Plan B violates this constraint, since the Cp of the first proposition fails to predict the Cb of the second.*

On the basis of these five constraints, plan A (no violations) will be preferred to plan B (two violations). However, this difference in favour of plan A might be over-ridden by costs resulting from the remaining constraints, which depend upon the style parameters controlled by the user.

Paragraph Length
Although no words have been generated yet, paragraph length can be measured from the number of propositions expressed in each paragraph. The cost is calculated as a deviation from an ideal length determined by the user.

Sentence Length
The length of each text-sentence is also measured by the number of propositions it contains, and scored by deviation from the user-controlled ideal length.

Connective Frequency
If the user requests frequent connectives, failure to use a discourse marker is penalized; conversely, if the user requests few connectives, the presence of a marker is penalized.

Passive Voice
Inclusions of passives in the output text are penalised if the user requests a low frequency of occurence, and *vice versa* for actives when a high frequency of passives are requested.

Semicolon Frequency
Following Nunberg's punctuation rules [3], every text-clause will end in a semicolon unless it is the final constituent of a text-sentence. If the user requests frequent semicolons, text-sentences with only one text-clause are

penalized; if the user requests infrequent semicolons, text-sentences with more than one text-clause are penalized.

Graphical Impact

Under standard settings, the hard constraints allow vertical lists only for multinuclear relations (e.g. alternative, sequence). Failure to use an indented structure in such cases is penalized if the user has requested high graphical impact; the presence of an indented structure is penalized if the user has requested low graphical impact.

4 Conclusions

Although we have concentrated here on stylistic constraints at the level of text planning, the approach we take to optimization means that possibilities at other levels are also considered at no extra cost. For example, text planning constraints on the setting for the preferred center (i.e., Cp, the most salient referent) will rule out certain syntactic choices since even at the text-planning stage, the eventual use of the passive can be foreseen if the ACTEE of an action is the Cp. Similarly, constraints on text category will have a direct bearing on the appearance of semicolons.

The system we have developed has proved a useful research tool for investigating the interaction between stylistic goals. Through the generate-and-test method, one can quickly evaluate the consequences of a given stylistic choice and discover new constraints that should be added.

The system can also be viewed as an authoring tool that allows users to specify not only the content of a document to be generated — as in other systems, e.g., [14–17, 8] — but also fairly fine-grained decisions over the style of the output text.

Finally, the system has the added capability of being self-critiquing: the user can, if he or she wishes, see the extent to which any or all of the generated versions deviates from what would in theory be ideal. This too is achieved at no extra cost.

References

1. Strunk, W., White, E.: The elements of style. MacMillan (1979)
2. Hovy, E.: Generating Natural Language under Pragmatic Constraints. Lawrence Erlbaum Associates, Hillsdale, NJ (1988)
3. Nunberg, G.: The Linguistics of Punctuation. CSLI, Stanford, USA (1990)
4. Kager, R.: Optimality Theory. Cambridge Textbooks in Linguistics. Cambridge University Press (1999)
5. Power, R.: Planning texts by constraint satisfaction. In: Proceedings of COLING-2000. (2000)
6. Hentenryck, P.V.: Constraint Satisfaction in Logic Programming. MIT Press, Cambridge, Mass. (1989)

7. Mellish, C., Knott, A., Oberlander, J., O'Donnell, M.: Experiments using stochastic searcg for text planning. In: Proceedings of the Ninth International Workshop on Natural Language Generation, Niagara-on-the-Lake, Ontario, Canada (1998) 98–107
8. Power, R., Scott, D.: Multilingual authoring using feedback texts. In: Proceedings of the 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics, Montreal, Canada (1998) 1053–1059
9. Power, R.: Controlling logical scope in text generation. In: Proceedings of the European Workshop on Natural Language Generation, Toulouse, France (1999) 1–9
10. ECRC: Eclipse user manual. Technical report, European Computer Research Centre, Munich, Germany (1992)
11. Grosz, B., Joshi, A., Weinstein, S.: Centering: a framework for modelling the local coherence of discourse. Computational Linguistics **21** (1995) 203–225
12. Kibble, R., Power, R.: Using centering theory to plan coherent texts. In: Proceedings of the 12th Amsterdam Colloquium, Institute for Logic, Language and Computation, University of Amsterdam (1999)
13. Strube, M., Hahn, U.: Functional centering: Grounding referential coherence in information structure. Computational Linguistics (1999)
14. Caldwell, D., Korelsky, T.: Bilingual generation of job descriptions from quasi-conceptual forms. In: Proceedings of the Fourth Conference on Applied Natural Language Generation. (1994)
15. Paris, C., Vander Linden, K., Fischer, M., Hartley, A., Pemberton, L., Power, R., Scott, D.: A support tool for writing multilingual instructions. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence, Montreal, Canada (1995) 1398–1404
16. Power, R., Cavallotto, N.: Multilingual generation of administrative forms. In: Proceedings of the 8th International Workshop on Natural Language Generation, Herstmonceux Castle, UK (1996) 17–19
17. Sheremetyeva, S., Nirenburg, S., Nirenburg, I.: Generating patent claims from interactive input. In: Proceedings of the 8th International Workshop on Natural Language Generation, Herstmonceux Castle, UK (1996)