Toward balancing conciseness, readability and salience: an integrated architecture [*]

Bruce Eddy
Department of Computing and Electrical Engineering
Heriot-Watt University
Edinburgh
EH14 4AS
UK
B.D.Eddy@hw.ac.uk

Abstract

In this paper we present an architecture for generating texts that vary in the emphasis put on conciseness, readability and the marking of particularly salient items. We abandon the traditional pipeline architecture, and use an integrated approach which makes the search for an optimum text explicit, taking into account both inter-sentential and intra-sentential features. We describe a context sensitive scoring system which can relate surface properties to a deeper representational level. We show how this approach can be used in generating paragraph length texts, optimised against various criteria.

1 Introduction

1.1 Overview

The consensus view of the generation community is that the pipeline architecture (Reiter and Dale, 2000) is suitable for many generation tasks. However, this architecture does not easily support the control of certain properties of the surface form of the generated text. An example of this is the text's length (Reiter, 2000). However, a more fundamental difficulty is that many of the textual characteristics which are problematic for a pipeline to generate are also difficult to measure computationally. Examples of these are readability and the marking of particularly salient items.

The reason for the pipeline's difficulty with certain surface forms was given by Zock (Zock, 1987) and Danlos (Danlos, 1987). It may not be easy to determine what effect a decision taken at an early stage in the pipeline will have at the surface; and decisions taken at one stage may preclude at a later stage a choice which results in a more desirable surface form. In general, the interrelations between the levels of representation used in NLG are complex and imperfectly understood. Various solutions to this problem are described in (Reiter and Dale, 2000) and (Reiter, 2000). The pipeline's primary competitor is an integrated architecture.

The systems STREAK (Robin and McKeown, 1996) and MAGIC (McKeown et al., 1998) illustrate the trade-offs implied by pipelining or integration. The former integrates aggregation and lexicalisation, thereby producing concise output text, but slowly. The latter pipelines these functions, thereby generating quickly, but at the cost of less optimal texts. Another disadvantage of an integrated architecture is that a representation which can model all the constraints to be integrated is likely to be unwieldy. SPUD (Stone and Doran, 1997) integrates communicative intent and semantic and syntactic constraints, but it is acknowledged that it is difficult to build resources for use with this system.

In the next section we present the problem we have investigated. It requires flexibility in the use of textual properties (length, salience marking and readability) which cannot be controlled easily by a pipeline. In section 3 we describe our system, which

[*]This work is supported by UK Engineering and Physical Sciences Research Council grant GR/M23106.

uses an integrated architecture. We explain that it can generate the texts we require, but that it requires suitable measures of the desired surface properties in terms of deeper representations. In section 4 we describe a possible solution to the problem of finding such measures. In section 5 we describe a preliminary evaluation of this solution.

2 The problem we address

We have investigated a domain in which flexibility in certain characteristics of the surface text is required. This is the generation of simple concise descriptions from data about objects. In particular, we have examined the generation of descriptions of online educational resources from rich meta-data; such data is associated with a large number of these.

For this task, certain components of the description should be marked as particularly salient; the text should be fluent and readable; the text should be concise. These constraints may conflict with each other. Our texts may be used in a variety of contexts; in each of these, the relative importance of the constraints may be different. So we require to be able to control which prevails in a particular generated text.

The input data is created from a catalogue record for an online resource (drawn from a large database of such records), which consists of a set of fields. The specification for these records provides for about 50 fields. The fields are attribute value pairs. Each generation run expresses a record, but not necessarily all of its 50 fields. This is because, first, the database is imperfect: not every record has data in every field. And second, not all fields are of equal interest to a particular user: we employ a user model which filters out fields which are of no interest, and marks those which are particularly salient.

Hence, the input data set to a particular generation run is a collection of fields drawn from a particular record, with some fields marked as particularly interesting. See figure 1. The generator's task is to express all these fields as a concise, readable paragraph, and to express those marked as salient in a way which indicates their status.

The target text, with the publisher's name and the grade deemed to be particularly salient, and with a requirement for for conciseness and readability, might be:

> "Wild World of Words Challenges" is published by online provider AskOLLY, whose address is OLLY/IT, Delmott University, Delmott, NY 13244. It is suitable for grade 3 students and is an HTML format lesson plan. It will benefit students and is a tool for teaching professionals which is available for free.

The salient items are indicated as such by their position in the main clauses of the first and second sentences. Notice that the prominence of the salient items would be increased somewhat by the absence of the relative clause "whose address...". We may suppose that the conciseness requirement precluded the use of a (marginally) longer form, with single sentence "AskOLLY's address...", placed as the fourth sentence.

So the problem is one of document structuring and microplanning. Document structuring is the task of deciding in which order the items should be presented; microplanning that of setting sentence boundaries and aggregating. However, these two tasks conflict, because microplanning which aims to produce syntactically aggregated text must re-order the items.

3 Using an integrated architecture

Our approach uses as its primary representation tree-adjoining grammar (TAG) (Joshi, 1985) extended to include unification based feature structures (Vijay-Shanker and Joshi, 1991).

The TAG used by our system is such that its string set is exactly those paragraphs which are comprehensible summaries of subsets of fields. That is, for every possible input data set, there is a string in the grammar which expresses it. It should be noted that such a TAG not only models syntactically correct sentences, but also integrates semantic checking and, to some extent supra-sentence level features (i.e. ordering and pronominalisation). We have designed a prototype for such a TAG, which models a limited number of summary paragraphs.

(Joshi, 1986) described the advantages TAG possesses as a syntactic formalism for NLG. TAG has usually been applied to generating clauses and sentences. (Webber et al., 1999) outlined the benefits of modelling larger strings of text by the same means.

Field Name = Value	Particularly interesting?
DC.Type = lesson plan	
GEM.Grade · Grade = 3	*
DC.Format · ContentType = text/html	
DC.Publisher · Role = onlineProvider	
DC.Publisher · Name = AskOLLY	*
DC.Publisher · Postal = OLLY/IT, Delmott University, Delmott, NY 13244	
DC.Rights · PriceCode = 0	
DC.Title = "Wild World of Words Challenges"	
GEM.Audience · ToolFor = teaching professionals	
GEM.Audience · ToBenefit = students	

Figure 1: A typical input data set.

We also use a scoring system which indicates the extent to which its string obeys surface constraints. The scoring system is the subject of sections 4 and 5. At this point, it is sufficient to note three points. First, it maps a TAG derivation structure (or fragment of a derivation structure) to a tuple of numerical numerical scores. Second, each entry in this tuple indicates how well a complete derivation's string obeys one surface constraint. Applied to a fragment, it evaluates what contribution that fragment is expected to make to a complete derivation in which it might occur. Hence it may be used during generation as an heuristic to guide the search. Third, the relative weight attached to each constraint can be set as a parameter before generation begins.

The generation task may be viewed as a constraint satisfaction problem. The system must find a text which

- is syntactically correct
- is semantically correct
- contains all the input data
- is short
- is readable
- has the salient items at prominent positions

The first two are incorporated into the grammar as outlined above: the text modeled by any derivation in the grammar meets these two constraints. This is achieved by the careful design of feature structures on elementary tree nodes. For example, in a particular sentence the subject and object must agree with the person and number of the verb; also. they must refer to entities which may stand in the relationship represented by the verb. Feature structures may impose these constraints.

The other constraints are integrated by the generator in two stages. The first stage is a transformation of the grammar, similar to that described in (Eddy et al., 2001). It is a bottom-up procedure on the elementary trees which directly express the required fields. Its input is the original grammar, G, together with the input data set, and the scoring table (see section 4) associated with each of G's elementary trees. The output is a number of sub-grammars of G, $G_1 \ldots G_n$. Each G_i's string set is a subset of G's; each string expresses only fields in the input data. Further, the enumeration of each G_i's string set is feasible (although this has not been proved). In addition, the transformation associates with each of the G_i an indication of the likelihood of the string which optimally meets the surface constraints being found in G_i's string set. This indication is informed by the scoring system.

The second stage selects some of the G_i and enumerates their string sets. It bases its selection on the indication provided by the first stage. It then selects, from the strings found by the enumerations, the best, as judged by the scoring system.

4 Context-sensitive scoring system

The scoring system we now describe is similar to the evaluation metric used by (Cheng and Mellish, 2000) in that it relates surface properties to deeper structures, in our case TAG derivation trees. However, we hope that it is more flexible, and can subsume that metric. We make two observations about the scoring system. First, the scoring system is independent of any particular generation system. We have implemented the generation system described in the previous section, and it makes use of the scoring system to guide its search, but other, perhaps better, uses could be made of the scoring system. Second, we intend that the scoring system be sufficiently parameterisable that it is independent of any particular surface constraint. It is dependent on its parameters being set up (or "trained") appropriately. We describe in this section how it might be used to model conciseness, readability and salience. In section 5 we describe and evaluate the results of training it to model readability.

The scoring system operates on TAG derivation structures (whole or fragmentary). It maps each derivation to a tuple of numerical scores; one entry of the tuple is for each constraint being modeled. For example, suppose the scoring system models conciseness, salience and readability, and suppose that some string has a corresponding derivation whose tuple the scoring system calculates as (1,5,2). Then the extent to which the string is concise may be read off as 1, salient as 5, and readable as 2. (Of course, these values are only meaningful relative to those for other strings.)

Each elementary tree in a derivation structure supplies a tuple of scores. The first element in the tuple for the complete structure is calculated as the mean of the first entries in the tuples supplied by each constituent elementary tree, and so on for the second and subsequent entries.

The tuple supplied by an elementary tree is based on a scoring table, and the context in which the elementary tree finds itself. Figure 2 is an example of a scoring table. Each elementary tree in a grammar is associated with one. A context is a derivation structure, or a fragment of a derivation structure. Figure 2 evaluates as a 3-tuple. The n^{th} column supplies the score for the n^{th} entry in the tuple. Each column contains a default score and a sequence of tree fragments, each associated with a score.

The procedure for selecting a score from a column is as follows. Each fragment in the column is compared with the context of which the elementary tree is a constituent. If any fragment matches the context, that is, if the fragment is a subtree of the context, then the score associated with that fragment is chosen. If no fragment matches then the default score is chosen. If more than one fragment matches, then a conflict resolution procedure decides which score should be used (for example, the score with the highest absolute value might be chosen).

We now illustrate the procedure described above. We use an abbreviated linear notation for derivation fragments: elementary trees are named by Greek letters, a tree is represented by a lisp-like list, and compositional information (adjoin or substitute, and address) is omitted. Suppose figure 2 is the score table for elementary tree ζ. In the derivation $(\alpha\ (\beta\ \gamma)\ \zeta)$, it contributes the tuple $(0, 10, 4)$. It scores 0 for conciseness (because (α) is a subtree of the context), 10 for salience ($((\alpha\ (\beta\ \gamma))$ is a subtree) and 4 for readability (none of the listed contexts match, so the default value is used).

To illustrate the use of the context mechanism, consider two problems: avoiding repetition in syntactic construct used, and marking salience. We can assign scores for (say) a relative clause which are lower when used in the context of another relative clause. We can also assign scores for a tree containing a given data item that are higher when that item occurs in the context of the main clause. While such a local context mechanism no doubt miss aspects of the interactions which result in quality text, it is manageable and seems to capture many useful features.

5 Evaluation

A small-scale evaluation of the scoring system with respect to the readability of paraphrases was performed. We compared the system's assessment of texts' readability with that of human judges. Baselines were established by comparing other readability measures with the judges' assessments. The results were inconclusive, but provide a basis for future work.

Constraint	Conciseness	Salience	Readability
Values in context (context=value)	(α)=0 $(\gamma\ \delta)$=-1	$(\alpha\ (\beta\ \gamma))$=10	(ϵ)=0.2 (δ)=2 $(\gamma\ \delta)$=-3
Default value	5	1	4

Figure 2: An example of a scoring table for some elementary tree.

17 sets of input data were chosen, and divided randomly between "training" and "test" portions, containing 5 and 12 input sets respectively. We then "trained" the parameters of the scoring system by hand, by modifying the scoring tables associated with the elementary trees in the grammar. At the end of the training procedure, the context sensitive scoring system's assessment of the texts approximately matched our intuitions about their readability.

The test phase was carried out in a procedure similar to that used by (Bangalore et al., 2000). Human subjects were asked to read short paragraphs. For each, they answered two questions on a 7 point scale.

- **Understandability:** "How easy is this paragraph to understand?" The options were labelled "Difficult" (=1), "Fairly easy" (=4), "Very easy" (=7). Intermediate values were possible but unlabelled.
- **Quality:** "How well-written is this paragraph?". The options were labelled "Horrible" (=1), "Alright" (=4), "Very well-written" (=7). Again, intermediate values were possible but unlabelled.

There were 10 subjects, who were staff and students of the Department of Computing at Heriot-Watt University. 50 paragraphs were used. These were selected according to the following scheme. For each of the 12 test input data sets, a collection of paragraphs was generated. (The texts in each collection are paraphrases of the same information). For each collection of paraphrases, the best, worst, and two (in two cases, three) intermediate texts were selected, the judgment of quality being made by the (trained) scoring system. Each subject judged 20 paragraphs, and hence each paragraph was judged by 4 subjects.

The data was normalised[1] to correct for the variation between subjects. The final score for a text's understandability was the mean of the normalised judgments for understandability for that text. Similarly, the final score for a text's quality was the mean of the normalised judgments for quality for that text. It was expected that there would be strong correlation between the normalised quality and understandability judgments. The data confirmed this: the correlation co-efficient between the mean normalised understandability scores and the mean normalised quality scores was 0.92 ($p < 0.05$).

The subjects' judgments were then compared with various readability metrics:

- The scores assigned by the trained context sensitive scoring system
- (Baseline 1) The Flesch easy reading index[2]
- (Baseline 2) The length in characters of the the paragraph.

Our main hypothesis was that the scoring system would correlate better with understandability and quality than our baseline metrics. This was not confirmed. The correlations of the scores with normalised readability and quality were 0.23 and 0.21 respectively. For both $p > 0.05$. The correlations of the Flesch Index with normalised readability and quality were 0.43 and 0.39 respectively. For both $p < 0.05$. The correlations of simple length with normalised readability and quality were 0.02 and 0.02 respectively. For both $p > 0.05$.

[1]Following the procedure of (Bangalore et al., 2000), let s_{ij} be the score that subject i gave to the j^{th} paragraph. The normalisation of s_{ij} is $\frac{s_{ij} - Mean\ of\ (s_{i1} \ldots s_{i20})}{Standard\ deviation\ of\ (s_{i1} \ldots s_{i20})}$.

[2]Flesch Index $= 206.835 - 84.6\frac{syll}{wds} - 1.015\frac{wds}{sent}$, where $syll$, wds and $sent$ are the number of syllables, words and sentences respectively, in the paragraph.

Although the non-confirmation of our hypothesis is a blow for the context-sensitive scoring system, it is not the end. The training of the parameters for the present evaluation was unsatisfactory. This is primarily due the labouriousness of hand training. Because of this, only a small quantity of training data could be used and hence the resultant parameters lacked generality. Of course, the following question remains open. Could *any* training of the context sensitive scoring system yield a metric sufficiently general to perform well in the evaluation above? We believe it could, but that the successful procedure could not be carried out by hand.

6 Conclusion and further work

We have outlined the argument against using a pipeline architecture for generating texts in which flexible control of surface features is required. We have indicated the need for a computational assessment of these surface phenomena, and described the beginnings of a system for supplying that: a system of context-sensitive scores.

Devising context sensitive scores is not easy and is a time consuming task. However, we expect it to be possible to for these scores to be learned by a relatively simple procedure given texts ranked by a human judge. The judgments supplied for the above evaluation will initially provide the basis for this. We will then investigate training the system to other surface phenomena.

Success will provide two benefits. First, we will have metrics for certain surface phenomena in terms of TAG derivations. Second, these metrics might give insight into other characterisations of these phenomena.

References

Srinivas Bangalore, Owen Rambow, and Steve Whittaker. 2000. Evaluation metrics for generation. In *Proceedings of International Natural Language Generation Conference*, Mitzpe Ramon, Israel, June.

Hua Cheng and Chris Mellish. 2000. Capturing the interaction between aggregation and text planning in two generation systems. In *Proceedings of International Natural Language Generation Conference*, Mitzpe Ramon, Israel, June.

Laurence Danlos. 1987. *The Linguistic Basis of Text Generation*. Cambridge University Press, Cambridge, UK.

Bruce Eddy, Diana Bental, and Alison Cawsey. 2001. An algorithm for efficiently generating summary paragraphs using tree-adjoining grammar. In *Proceedings of the 8th European Workshop on Natural Language Generation*, pages 39–46, Toulouse, France, July.

Aravind K. Joshi. 1985. How much context-sensitivity is required to provide reasonable structural descriptions: Tree Adjoining Grammars. In D. Dowty, L. Karttunen, and A. Zwicky, editors, *Natural Language Processing: Psycholinguistic, Computational and Theoretical Perspectives*. Cambridge University Press, Cambridge.

Aravind K. Joshi. 1986. The relevance of tree adjoining grammar to generation. In *Natural Language Generation: New results in Artificial Intelligence, Psychology and Linguistics - NATO Advanced Research Workshop*, pages 233–252, Nijmegen, The Netherlands.

Kathleen R. McKeown, Desmond A. Jordan, and Vasileios Hatzivassiloglou. 1998. Generating patient specific summaries of online literature. In *AAAI Spring Symposium on Intelligent Text Summarisation*, pages 34–43.

Ehud Reiter and Robert Dale. 2000. *Building Natural Language Generation Systems*. Cambridge University Press.

Ehud Reiter. 2000. Pipelines and size constraints. *Computational Linguistics*, (26):251–259.

Jacques Robin and Kathleen R. McKeown. 1996. Empirically designing and evaluating a new revision-based model for summary generation. *Artificial Intelligence*, 85(1-2), August.

Mathew Stone and Christine Doran. 1997. Sentence planning as description using tree-adjoining grammar. In *Proceedings od the Assosciation for Computational Linguistics*, pages 198–205.

K. Vijay-Shanker and Aravind K. Joshi. 1991. Unification based tree adjoining grammars. In J. Wedekind, editor, *Unification-based Grammars*. MIT Press, Cambridge, Massachusetts.

Bonnie Webber, Aravind K. Joshi, Alistair Knott, and Matthew Stone. 1999. What are little texts made of? a structural presuppositional account using lexicalised tag. In *Proceedings of International Workshop on Levels of Representation in Discourse*, Edinburgh, July. LOIRD'99.

Michael Zock. 1987. Proposal for simulating on-line generation by giving priority either to lexical choices or syntactic structure. In *COGNITIVA*.