An algorithm for efficiently generating summary paragraphs using tree-adjoining grammar*

Bruce Eddy, Diana Bental and **Alison Cawsey**
Department of Computing and Electrical Engineering
Heriot-Watt University
Edinburgh
EH14 4AS
UK
{ceebde1,diana,alison}@cee.hw.ac.uk

Abstract

We present an algorithm which improves the efficiency of a search for the optimally aggregated paragraph which summarises a flat structured input specification. We model the space of possible paraphrases of possible paragraphs as the space of sequences of compositions of a set of tree-adjoining grammar (TAG) elementary trees. Our algorithm transforms this to a set with equivalent paraphrasing power but better computational properties. Also, it identifies an explicit mapping between input propositions and their possible surface realisations.

1 Introduction

Summarisation of simply structured data as short natural language paragraphs has recently been a focus of interest. Shaw (1998) and Bental et al. (1999) looked at generating text from database records. Robin and McKeown (1996) summarised quantitative data. Shaw's examples were drawn from patient medical records; Bental et al's from online resource cataloging information. A requirement common to all these studies has been to produce *aggregated* (Reape and Mellish, 1999) text.

Also in all these studies, the structure of the input data used was fairly flat. In particular, in (Shaw, 1998) and (Bental et al., 1999) each record is associated with a particular entity (e.g. a patient or an online resource) and is essentially a list of attribute-value pairs. We refer to pairs as *fields*, and to attributes as *field names*. The relationship between a value and the entity with which it is associated is specified by the field name. Most field names represent "is a" or "has a" relationships and hence most values represent facts about the entity. Slightly more complex structure may also be coerced into this form, but we will focus on this simple case.

For our application (summarising data about educational resources), we additionally assume that we are required to be able to summarise any subset of fields from a given record, and that our summary must include every member of that subset. The challenge from this sort of summarisation is to devise a system which satisfies two potentially incompatible constraints. First, it must be flexible enough to model, for any combination of fields, the optimally aggregated paragraph which expresses them. Second, despite the very large search space that such flexibility probably implies, it must be capable of finding that paragraph in a reasonable time.

The contribution the present work makes is a set of algorithms which prune this search space. This space is specified in terms of compositions of elementary trees of a tree-adjoining grammar (TAG) (Joshi, 1986). The first transforms a TAG into a lexicalised version of itself which has better computational properties with respect to summarising a record. The second removes those parts of a TAG which are redundant with respect to summarising a particular record. The third

* This work is supported by EPSRC grant GR/M23106.

identifies an explicit mapping from each field to its possible surface realisations, and hence allows a desirable surface form to be chosen for each field. Our partial implementation of these algorithms has produced some promising results.

The rest of the paper is organised as follows. In section 2 we discuss the problem, our approach to modelling it, and characteristics of the search space implied by our approach. In section 3 we present our algorithms for searching this space. In section 4 we summarise and discuss.

2 Searching for concise, coherent paragraphs

2.1 Aggregation is a global optimisation problem

Our aim is to generate paragraphs which are well aggregated. This notion should be defined in terms of conciseness and coherence, which terms are not formally definable. However, some reasonable approximation to them can be achieved by specifying preferences for certain types of syntactic constructions over others (Robin and McKeown, 1996), possibly by giving each generatable construction a score which reflects its relative preferability. We then define the best aggregated paragraph to be that which achieves the best sum of its constituent constructions' preference scores.

Robin and McKeown's (1996) system, STREAK, generates aggregated, fact-rich sentences. It adds facts in order of the preferability of their best possible realisation. It *revises* its syntactic choices every time an extra fact is aggregated. This is computationally expensive, and makes multi-sentence generation by the same means prohibitively slow. They do suggest how to deal with this when many of the facts occur in fixed positions, but this is not the case in our corpus.

CASPER (Shaw, 1998) delays syntactic choices until after it has decided where sentence boundaries should fall. It thereby gains computational efficiency at the cost of its sentences being less optimal aggregations.

Our algorithms are an attempt to avoid these problems and to achieve greater efficiency by precompiling detailed syntactic decisions, the results of which we store in the form of explicit mappings from fields to surface forms. At generation time, we search for the optimal selection of realisations. This approach deviates from the pipeline architecture of NLG systems which, it has been observed, is not wholly suited to the generation of aggregated texts.

The first author to discuss this was Meteer (1992), who showed that the microplanning stage of the pipeline is constrained by surface realisation in two ways. First, a microplan must be realisable in the target language; second, a realisable microplan must make best use of the capacities the target language for concise expression.

More generally, in order to generate aggregated text, constraints imposed by and opportunities afforded by the surface form may be taken into account at any stage in the pipeline. Reape and Mellish (1999) provided examples of different systems each of which takes aggregation decisions at a different stage. It may not be easy to determine what effect a decision taken at an early stage will have at the surface; and decisions taken at one stage may preclude at a later stage a choice which results in a more aggregated surface form. Similarly, it may not be easy to make a decision at an early stage which makes best use of the surface possibilities.

Consider the examples of figures 1 and 2. Both summarise the same set of fields; figure 2 summarises additionally the field "subject = science". Both paragraphs summarise their fields in the most concise and coherent manner possible (although this is, of course, a subjective judgement). Note that they treat the fields they have in common differently with respect to ordering and distribution between the sentences.

Various types of constraints cause this. Syntactic constraints include: "science" may be used as an adjective to pre-modify "lesson plan". Semantic constraints include: "Constellations" lasts 4 hours, but ProLog does not. Stylistic constraints include: 'Maureen Ryff wrote ...' is preferable to '... was written by Maureen Ryff'. We suggest, as do Stone and Doran (1997), that integrating these constraints simultaneously is more efficient then pipelining them. We additionally suggest that representing these constraints in a unified form can provide further efficiency gains.

"Constellations" is a 4-hour lesson plan published by online provider ProLog. Maureen Ryff wrote it for small group teaching.

Figure 1: A paragraph which summarises the set of fields of figure 3 in an aggregated manner.

"Constellations" is a science lesson plan which lasts 4 hours. Maureen Ryff wrote it for small group teaching and ProLog, an online provider, is its publisher.

Figure 2: A well aggregated paragraph which summarises the set of fields of figure 1, together with field `subject` = "science". Notice the non-linear effect the addition of a single extra proposition can have on the structure of the paragraph.

2.2 Modeling paragraphs with TAG

Our approach uses, as its primary representation, TAG formalism extended to include unification based feature structures (Vijay-Shanker and Joshi, 1991). Joshi (1986) describes the advantages TAG possesses as a syntactic formalism for NLG. In generation, the TAG model has usually been applied to generating clauses and sentences. Recently, Webber et al. (1999) outlined the benefits of modelling longer strings of text by the same means.

The most important characteristic of TAG for our purposes is the local definability of dependencies: constraints between the nodes of elementary trees are preserved under adjoinings which increase the distances between them. For example, in the sentence fragment "Springer published ...", which might be modelled by a single initial tree, the object is constrained to be some entity published by Springer. If an adjunction is made so that the fragment becomes "Springer, the largest publishing company in Europe, published ...", this constraint is undisturbed.

Our approach presupposes the existence of a TAG whose string set is exactly those paragraphs which are comprehensible summaries of subsets of fields. We do not discuss the creation of such a TAG here. We have made progress with designing one; we believe that it is the flatness of the input data which makes it possible.

Let us restate the problem somewhat more formally. Suppose that we have a set F_0 of n fields whose values we may be required to express. Suppose that for every $C \subset F_0$ there is a *template* which *expresses* C. A template is a paragraph in which certain words are replaced by *slots*. A slot is a reference to a field-name. A template T expresses a set of fields C if the name of every element of C is referenced by a slot, and every slot refers to an element of C. We say that T expresses C and that the resulting paragraph is the expression of C with respect to T. See figure 3.

Let $e(C)$ denote the template which "best" expresses some $C \subset F_0$. Suppose also that we have a TAG T_0 with string set $S(T_0)$ such that $\{e(C) \,|\, C \subset F_0\} \subset S(T_0)$. The creation of T_0 is not discussed here. Every string in $S(T_0)$ is the yield of D, some derived tree of T_0.

Each of a TAG's derived trees is represented by a unique derivation tree. Typically a derivation tree represents several (differently ordered) sequences of compositions of (the same set of) elementary trees, all of which result in the same derived tree.

Hence, a derived tree D of a TAG T with elementary trees E is the result of some sequence of compositions of the elements of E which is equivalent to some derivation tree δ. We write $D = \delta(E)$ or just $D = \delta(T)$, Hence, our problem is, given T_0 and some $C \subset F_0$, find some δ such that $\delta(T_0) = e(C)$.

There are two parts to the problem of finding $e(C)$. First we must recognise $e(C)$, which we may do, as described in section 2.1, by defining $e(C)$ to be the paragraph which achieves the best sum of its constituent constructions' preference scores. Second, since the search space of derivation trees grows exponentially with the number of trees in the grammar, we must find its derivation in a reasonable amount of time.

For each field to be expressed, the slot which refers to it may be expressed by means of one of several different syntactic constructions. So each slot will contribute one of several possible preference scores to the paragraph in which it occurs, depending on the syntactic form in which it is realised. However, the syntactic forms by which a

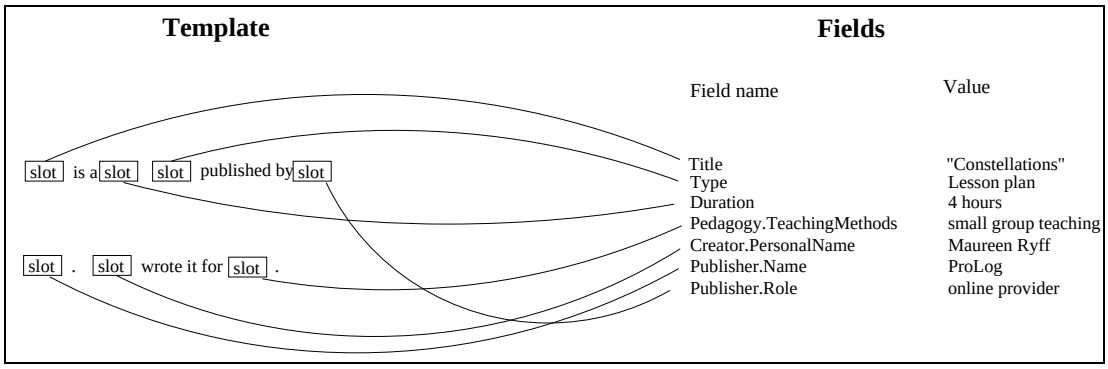

Figure 3: A template which expresses a set of fields. The curves indicate the field name to which each slot refers. The fields' expression with respect to the template is the paragraph in figure 1.

slot may be expressed are an implicit property of the grammar: it requires search to discover what they are, and then further search to find their optimal configuration.

2.3 The search space

We model possible paraphrases with a TAG, the paraphrases being the elements of its string set. The nodes in the search space are TAG derivation trees, and an arc from A to B represents the composition into the partial derivation corresponding to A, of an elementary tree, with result the partial derivation corresponding to B. The size of the search space may be reduced by collapsing certain paths in it, and by pruning certain arcs. These operations correspond to *specific lexicalisation* and the removal of redundant trees from the grammar respectively.

A tree in the grammar is redundant if it cannot contribute to a paragraph which expresses the required fields; or if it cannot contribute to a 'best' paragraph which expresses those fields. We will expand on redundancy removal after describing the specific lexicalisation algorithm. The specific lexicalisation algorithm converts a TAG into a *specific-lexicalised* version of itself, in which certain paths in the search space, which it can be known will be used in any derivation, are collapsed.

3 The algorithms

3.1 Specific lexicalisation: creating a clausal lexicon

We begin by introducing some notation, and defining some properties of TAGs and their elementary trees. Let $\langle T, N \rangle$ denote T, a TAG, and N, some set of elementary trees, not necessarily in T. A leaf node of an elementary tree may be labelled as a *slot*, which is a special case of the lexical anchor.

An elementary tree is *specific-lexicalised* if at least one[1] of its leaf nodes is a slot. An elementary tree is *λ-lexicalised* if it is specific-lexicalised or if it has no substitution nodes or foot nodes. A TAG is *specific-lexicalised* if all its elementary trees are λ-lexicalised.[2]

Given some $\langle T, N \rangle$, let t be an element of $T \cup N$ which is not specific-lexicalised. Let s be an elementary tree of T. Suppose that there is some composition of s into t and that the resulting tree is specific-lexicalised. Then we say that t is *single-step-lexicalisable in* $\langle T, N \rangle$. We call any such resulting tree a *single-step-lexicalisation at n of t in* $\langle T, N \rangle$, where n is the node at which the composition occurred.

We now present the algorithm for our transformation *Specific Lexicalisation.*

1: $T = T_0$ is some TAG, $N = H = \emptyset$.
2: **repeat**
3: Add $\langle T, N \rangle$ to H.
4: **for all** $t \in T \cup N$ **do**
5: **if** t is sinlge-step lexicalisable in $\langle T, N \rangle$

[1] This is an important parameter. It specifies how many slots each elementary tree in the transformed TAG may have (and consequently how many times the "derivation tree" of each of these trees branches).

[2] This definition is compatible with that used in the literature. A TAG is lexicalised (Joshi and Schabes, 1991) if it is specific-lexicalised according to this definition. The implication does not necessarily hold in reverse.

then

if $t \in T$ **then**

Remove t from T.

Add t to N.

end if

For some node of t, n, add all the single-step-lexicalisations at n of t in $\langle T, N \rangle$, to T.[3]

end if

end for

until $\langle T, N \rangle \in H$

T is a specific-lexicalisation[4] of T_0.

To illustrate this procedure, we have provided some figures. Consider the TAG T_1, whose elementary trees are shown in figure 4. We have chosen, for reasons of space and simplicity, not to show the feature structures attached to each node of these trees. Their approximate form can perhaps be deduced by examination of the templates modelled by T_1, shown in figure 6. A specific lexicalised version of the TAG, T_2 is shown in figure 5. We have named each elementary tree in T_2 by concatenating the names of its constituents from T_1. The templates generated by T_1 (and hence T_2) are shown in figure 6.

3.2 Redundancy removal

We can further remove redundancy in a specific-lexicalisation, T, of some TAG. Let $\langle T, N \rangle$ be a pair as in the previous section. The following three subsets of the elementary trees of T are redundant. *First*, those trees $t \in T$ which are not rooted on the distinguished symbol and for which there is no $s \in T \cup N$ such that t can be composed into s. *Second*, those $t \in T$ which have a substitution node into which no $s \in T \cup N$ can be substituted. *Third*, those $t \in T$ such that for each tree r which is the result of the composition of some $s \in T \cup N$ into t, $r \in T \cup N$. Our program which implements the algorithm in fact removes these redundancies, not only after completion, but also after every iteration.

3.3 Finding the (approximately) global optimum

Specific-lexicalisation causes the (previously implicit) grammatical constructions by which an element of C may be expressed to become explicit properties of the transformed grammar. Specifically, each element of C occurs as the anchor of each of a number of elementary trees. Let us refer to the set of elementary trees in the transformed grammar anchored by $c \in C$ as $a(c)$.[5] Each of these trees corresponds to a grammatical form in which the element may be realised. Hence, rather than performing an exhaustive search of the space of derivation trees $\sigma(T_2)$, specific-lexicalisation allows us to instead perform a *best first* search.

That is, we choose exactly one element of $a(c)$ for each $c \in C$. Let $choices(C)$ denote the set of all sets which contain exactly one element of $a(c)$ for each $c \in C$. Recall that we may assign to each syntactic form in which an element of C may be realised a preference score, and that each element of $a(c)$ corresponds to some syntactic form. So, for each element of $choices(C)$ we may sum the preference scores of its elements. Hence, we may impose an order on the elements of $choices(C)$ according to their sum of preference scores. We may then refer to each element of $choices(C)$ as $pref(C, i)$, where i is the element's position in the order, with $pref(C, 1)$ being first.

We then search, in order, the spaces of possible compositions of the $pref(C, i)$s combined with some necessary supporting trees which are not anchored by an element of C. Call these spaces $pref_s(C, i)$. In terms of redundancy removal, $pref_s(C, i)$ is the specific-lexicalised TAG with those trees which might be redundant with respect to the search for $e(C)$ removed. We begin the search with $pref_s(C, 1)$. It is not guaranteed that $e(C)$ is in this space. If it is not, we repeat the search using $pref_s(C, 2)$, and so on. At worst (if $e(C) \in pref_s(C, n)$ where $choices(C)$ has n elements), this procedure exhaustively searches the space of compositions of the elements of T_2.

[3] Note that there is a choice at this step. Our implementation of this algorithm chooses n such that the number of single-step-lexicalisations at n is maximised. But different choices result in a transformed grammar with different properties.

[4] We claim that a specific-lexicalisation of a TAG is indeed specific-lexicalised. Note that there does not necessarily exist a specific-lexicalisation of a TAG. For certain pathological examples of TAGs, the algorithm does not terminate. Note also that if a specific-lexicalisation exists, it is not necessarily unique. Further work is required to discover the properties of the various specific-lexicalisations in these cases.

[5] This is the family (as that term is used by Yang et al. (1991) and others) of trees anchored on c.

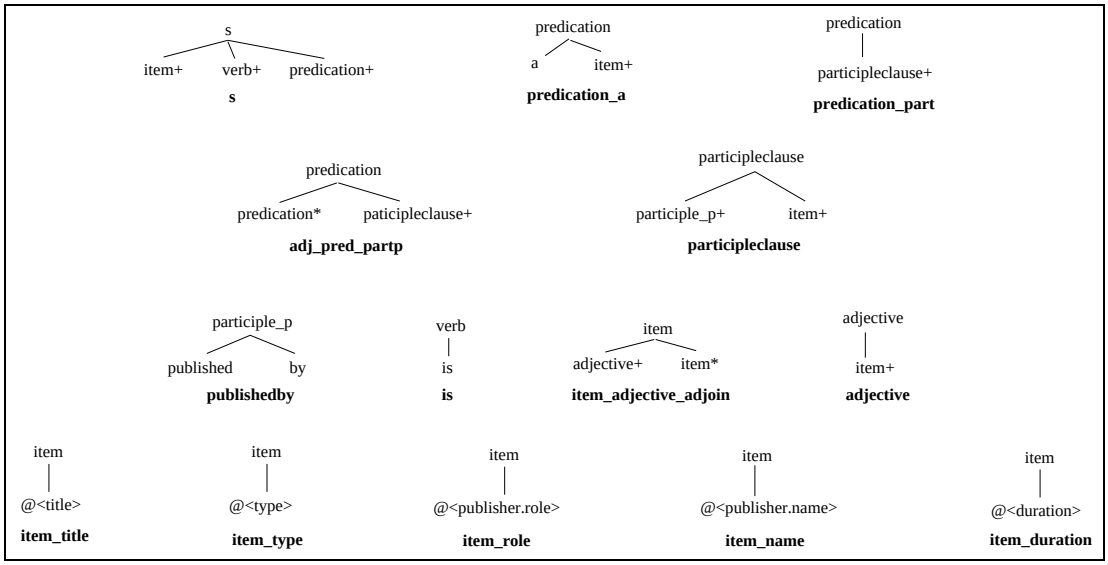

Figure 4: The elementary trees of a small TAG, T_1. The trees' names are in bold below them. Substitution nodes are indicated with a '+'; foot nodes are indicated with a '*'; the distinguished symbol is 's'. Slots are shown as '@<reference>', where "reference" is the field to which the slot refers. Note that the feature structures which are associated with each node, which prohibit certain compositions, are not shown. Note also that this is not a lexicalised TAG (LTAG). This is somewhat unusual; we intend, as part of our ongoing work, to apply our techniques to an established LTAG, such as XTAG.

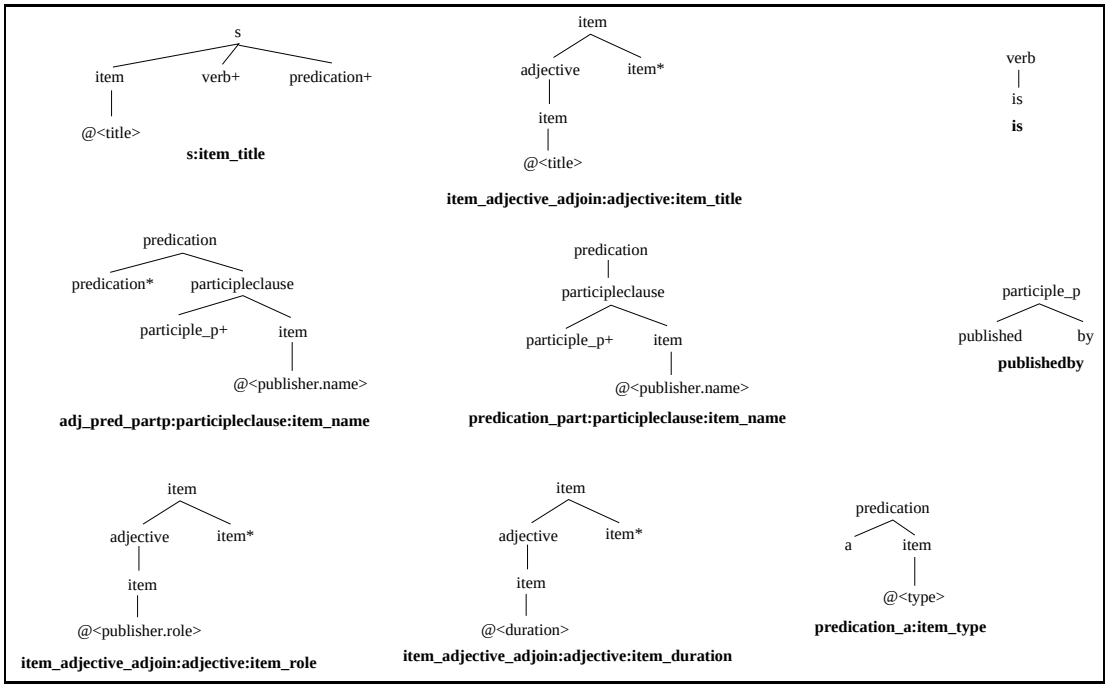

Figure 5: The elementary trees of the TAG T_2, a specific-lexicalised version of TAG T_1 in figure 4. Each tree's name is below it, in bold. Note that, since the feature structures are not shown, it is not apparent why certain trees which the algorithm seems to imply do not occur in this set.

1: @<title> is a @<type>.
2: @<title> is a @<type> published by @<publisher.name>.
3: @<title> is a @<type> published by @<publisher.role> @<publisher.name>.
4: @<title> is a @<duration> @<type>.
5: @<title> is a @<duration> @<type> published by @<publisher.name>.
6: @<title> is a @<duration> @<type> published by @<publisher.role> @<publisher.name>.
7: @<title> is published by @<publisher.name>.
8: @<title> is published by @<publisher.role> @<publisher.name>.

Figure 6: The templates modelled by the TAGs of figures 4 and 5. Note that the expression of the fields in figure 3 with respect to template 6 is the first sentence of the paragraph of figure 1.

In fact, since the $pref_s(C, n)$s do not partition $\sigma(T_2)$, in the worst cases this procedure is slower than an exhaustive search. However, $e(C)$ is defined in terms of maximal preference scores, so it is likely to be found in $pref(C, i)$ for "low" i.

For illustration, refer again to the specific-lexicalisation in figure 5. Notice that @<publisher.name> occurs as the anchor of more than one tree.[6] These trees, predication_part:-participleclause:item_name and adj_pred_partp:-participleclause:item_name which we will refer to as t_1 and t_2 respectively, represent the forms in which that slot may be expressed. Hence, @<publisher.name> may be realised as a predication in its own right using t_1, as in templates 7 and 8 in figure 6, or as an adjunct to another predication using the second, as in templates 2, 3, 5 and 6. Suppose that our preference scores rate t_2 more highly than t_1, and that we must include all four slots. Then the system would first search the space of compositions of the trees of T_2 without t_1, and generate template 6. The second choice, T_2 without t_2 leads to the generation of the concatenation of templates 4 and 8, which expresses the same fields but is less aggregated. This is as we would wish.

3.4 Redundancies in the search space

Specific-lexicalisation is a transformation which operates on a complete TAG T_1 and its result is another TAG T_2 whose string set is the same as T_1's. Also, the feature structures on the nodes of the elementary trees of T_2 contain fewer unbound variables. Unbound variables represent dependencies between parts of the grammar. A search of the space of compositions of elementary trees may make a long chain of compositions before discovering that the composed structure is forbidden by such a dependency. The forbidden chain of compositions is redundant, and specific-lexicalisation removes it from the search space.

Importantly, specific-lexicalisation may also take as a parameter C, the set of fields to be expressed. It then removes from T_1 all elementary trees which are anchored on slots which do not refer to elements of C and operates on this reduced TAG, with result T_2. And if $e(C) \in S(T_1)$ then $e(C) \in S(T_2)$. Then, in effect, specific-lexicalisation, as well as removing general redundant dependencies, is specifically removing some of those parts of the grammar which are redundant with respect to the search for $e(C)$.

Redundancy occurs in a grammar for two reasons. First, it is written, by hand, with linguistic rather than computational efficiency concerns in mind. It is too complex for its writer to be able to spot redundancies arising from long chains of dependencies between its parts. So specific-lexicalisation may be regarded as automatic bug removal. Second, the grammar is written to be able to model *all* the templates which express some $C \subset F_0$. So for any particular C, the grammar will contain information about how to express items not in that set. Specific-lexicalisation highlights this redundancy.

We have conducted some preliminary experiments using several small TAGs in which, for each TAG and for its specific-lexicalised equivalent, we measured the time our system takes to generate the modelled sentences. The results

[6] We are ignoring the tree item_adjective_adjoin:-adjective:item_title, which is not usable due to its features, which are not shown.

showed a decrease in the generation time after lexicalisation of orders of magnitude, with the best observed reduction being a factor of about 3000.

The specific-lexicalisation of a TAG has the property of having the same string set (and possibly the same tree set) as the original, but a smaller space of possible compositions. We have not proved either clause of this statement, but on the basis of experimental evidence we believe both to be true. Also, the following argument supports the case for the second.

Recall that a feature structure attached to a non-terminal symbol in some rule (tree in the case of TAG) of a grammar is an abbreviation for several similar rules. For example, if a node has associated with it a feature structure containing three features each of which may be in one of two states and none of which are currently instantiated, then it abbreviates $2^3 = 8$ nodes. So each tree in a TAG with feature structures is an abbreviation for n trees, where n is the number of possible configurations of the feature structures on its nodes. Hence, when we search the space of possible compositions of some number x of trees, we are in fact searching the space of compositions of ax trees, where a is some factor related to the number of possible configurations of the feature structures on the trees. Specific-lexicalisation identifies exactly which of the (non-featured) trees for which a tree with feature structures is an abbreviation are irrelevant to a search by instantiating unbound variables in its features.

4 Further work and discussion

The precise circumstances under which the techniques described are effective are still to be established. In particular, it is our intention to repeat our experiments with a standard LTAG; and with TAGs induced automatically from our corpus.

To summarise, we claim that the generation of an optimally aggregated summary paragraph requires the ability to move facts across sentence boundaries. A difficulty to achieving this is the exponential relationship between the number of possible paraphrases of a summary of a set of facts and the number of facts in that set. Our algorithm addresses this by transforming a TAG to better model the search space.

References

Diana S. Bental, Alison Cawsey, Sheila Rock, and Patrick McAndrew. 1999. The need for natural language generation techniques to produce resource descriptions in mirador. In *Searching for information: artificial intelligence and information retrieval approaches*, pages 15/1–15/3. Institution of Electrical Engineers, London.

Aravind K. Joshi and Yves Schabes. 1991. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Definability and Recognizability of Sets of Trees*. Elsevier.

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.

Marie Meteer. 1992. *Expressibility and the Problem of Efficient Text Planning*. Pinter, London.

Mike Reape and Chris Mellish. 1999. Just what *is* aggregation anyway? In *European Workshop on Natural Language Generation*, Toulouse, May 13–14.

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

James Shaw. 1998. Clause aggregation using linguistic knowledge. In *Proceedings of the 9th International Workshop on Natural Language Generation*, pages 138–147.

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.

G. Yang, K. F. McCoy, and K. Vijay-Shanker. 1991. From functional specification to syntactic structures: functional grammar and tree-adjoining grammar. *Computational Intelligence*, 7(4):207–219.