Pipelines and Size Constraints

Ehud Reiter*
University of Aberdeen

Some types of documents need to meet size constraints, such as fitting into a limited number of pages. This can be a difficult constraint to enforce in a pipelined Natural Language Generation (NLG) system, because size is mostly determined by content decisions, which usually are made at the beginning of the pipeline, but size can not be accurately measured until the document has been completely processed by the NLG system. I present experimental data on the performance of single-solution pipeline, multiple-solution pipeline, and revision-based variants of the STOP system (which produces personalised smoking-cessation leaflets) in meeting a size constraint. This shows that a multiple-solution pipeline does much better than a single-solution pipeline, and that a revision-based system does best of all.

1 Introduction

Some types of documents need to fit on a limited number of pages. For example, this article, because it is a squib, must fit on eight pages in the style (font, layout, etc) specified by *Computational Linguistics*. However, in certain cases it is useful to include as much information as possible given the size limit; for example I want to convey as much information as possible about my research in the allowed eight pages.

Maximising the amount of content subject to a size limit is also a problem for some Natural Language Generation (NLG) systems. For example, the STOP system (Reiter, Robertson, and Osman, 1999) produces personalised smoking-cessation leaflets which must fit on four A5 pages, in a certain style; but it is useful if the leaflets can convey as much information as possible given this size constraint.

One problem with performing this optimisation in an NLG system is that the size of a document is primarily determined by how much content it contains, that is by decisions made during the content determination process. However, an NLG system cannot accurately determine the size of a document until the document has been completely processed by the NLG system and (in some cases) by an external document presentation system, such as LaTeX or Microsoft Word. This is because the size of the document is highly dependent on its exact surface form. This is a phenomenon which may be familiar to readers who have tried to revise a paper to fit a page-limit constraint by making small changes to wording or even orthography.

In consequence, it may be difficult to satisfy the size constraint while 'filling up' the allowed pages in a pipelined NLG system which performs content determination in an early pipeline module, before the surface form of the document is known. This is especially true if each pipeline module is restricted to sending a single solution to the next pipeline module, instead of multiple possible solutions.

In this paper I give a brief summary of the pipeline debate and of STOP, present my experimental results, and then discuss the implications of this work.

* Dept of Computing Science, Aberdeen AB24 3UE, UK. Email: ereiter@csd.abdn.ac.uk

2 Pipelines in NLG

For the past 20 years, the NLG community has generally agreed that modularising NLG systems is sensible. This has become even more true in recent years, because of a growing trend to incorporate existing modules (especially realisation systems such as FUF/SURGE (Elhadad and Robin, 1997)) into new systems. While different systems use different numbers of modules, all recent systems that I am aware of are divided into modules.

This leads to the question of how modules should interact. In particular, is it acceptable to arrange modules in a simple pipeline, where a later module cannot affect an earlier module? Or is it necessary to allow revision or feedback, where a later module can request that an earlier module modify its results? If a pipeline is used, should modules pass a single solution down the line, or should they pass multiple solutions and let subsequent modules choose between these?

Many authors have argued that pipelines cannot optimally handle certain linguistic phenomena. For example, Danlos and Namer (1988) point out that in French, whether a pronoun unambiguously refers to an entity depends on word ordering. This is because the pronouns *le* or *la* (which convey gender information) are abbreviated to *l'* (which does not contain gender information) when the word following the pronoun starts with a vowel. But in a pipelined NLG system, pronominalisation decisions are typically made earlier than word-ordering decisions; for example in the three-stage pipelined architecture presented by Reiter and Dale (2000), pronominalisation decisions are made in the second stage (microplanning), but word ordering is chosen during the third stage (realisation). This means that the microplanner will not be able to make optimal pronominalisation decisions in cases where *le* or *la* are unambiguous, but *l'* is not, since it does not know word order and hence whether the pronoun will be abbreviated.

Many other such cases are described in Danlos's book (Danlos, 1987). The common theme behind many of these examples is that pipelines have difficulties satisfying linguistic constraints (such as unambiguous reference) or performing linguistic optimisations (such as using pronouns instead of longer referring expressions whenever possible) in cases where the constraints or optimisations depend on decisions made in multiple modules. This is largely due to the fact that pipelined systems cannot perform general search over a decision space which includes decisions made in more than one module.

Despite these arguments, most applied NLG systems use a pipelined architecture; indeed a pipeline was used in every one of the systems surveyed by Reiter (1994) and Paiva (1998). This may be because pipelines have many engineering advantages, and in practice the sort of problems pointed out by Danlos and other pipeline critics do not seem to be a major problem in current applied NLG systems (Mittal et al., 1998).

3 STOP

The STOP system (Reiter, Robertson, and Osman, 1999) generates personalised smoking-cessation leaflets, based on responses to a questionnaire about smoking likes and dislikes, previous attempts to quit, and so forth. The output of the system is a four page leaflet; each page is size A5. An example of the two 'inside' pages of a leaflet produced by STOP is shown in Figure 1. A STOP leaflet also contains a front page which is only partially generated (the rest is logos and fixed text) and a back page which is selected from a collection of 16 possible back pages, but is not otherwise personalised; these are not shown here due to space restrictions.

A STOP leaflet must fit on four A5 pages; this is a hard constraint. Furthermore, it is important to communicate as much information as possible subject to the size constraint; this is a characteristic that the system tries to optimise. However, it is even

Figure 1
Inside pages of an example STOP leaflet

Smoking Information for Ewan McDonald

You have good reasons to stop...

People stop smoking when they really want to stop. It is encouraging that you have many good reasons for stopping. The scales show the good and bad things about smoking for you. They are tipped in your favour.

THINGS YOU LIKE

you enjoy it
it's relaxing
it stops stress
it stops weight gain
it stops you craving

THINGS YOU DISLIKE

it's bad for you
it's expensive
it makes you less fit
it's a bad example for kids
it's bad for others' health
you're addicted
it's unpleasant for others
other people disapprove
it's a smelly habit

You said you don't like smoking because it is *bad for your health*. You are right to think this. Although you do not have any chest problems at the moment, inhaling smoke makes it more likely that you will do in the future.

If you stop smoking you are less likely to get heart disease and lung cancer in the future.

You also dislike smoking because it is *expensive*. Smoking 20 cigarettes a day costs over £1200 a year. You could spend that money on other things that you enjoy!

You could do it...

Although you don't feel confident that you would be able to stop if you were to try, you have several things in your favour.

- You have tried to stop before.
- You have good reasons for stopping smoking.
- You expect support from your partner, your family, your friends, and your workmates.

We know that all of these make it more likely that you will be able to stop. Most people who stop smoking for good have more than one attempt.

Overcoming your barriers to stopping...

You said in your questionnaire that you might find it difficult to stop because smoking helps you cope with *stress*. Taking a cigarette only makes you feel better for a short while. Most ex-smokers feel calmer and more in control than they did when they were smoking.

You also said that you might find it difficult to stop because you are *addicted to cigarettes*. If you were to stop smoking your body might take a while to get used to not having nicotine. This can cause unpleasant side effects, but they **will** go away. It might be worth thinking about using nicotine patches if you decide to stop smoking again. They help to reduce the withdrawal symptoms while you break the habit of smoking.

And finally...

We hope this letter will help you to think more about the benefits of stopping smoking tobacco. Many people who feel like you do now, do eventually stop smoking. Although it might be hard, if you really want to stop you will be able to do it.

With best wishes,

Aberdeen Medical Group.

more important that leaflets be easy to read, and size optimisation should not be at the expense of readability. For example, replacing an itemised list (such as the one at the top of the second page in Figure 1) by a complex multi-clause sentence can reduce size but often makes leaflets harder to read, especially for poor readers; hence we do not do this.

The original version of STOP used a three stage pipelined architecture (with each pipeline module producing only one solution) similar to the one presented by Reiter and Dale (2000). An initial document-planning stage produced a 'document plan' data structure which specified the content of the document in terms of 'messages'. In STOP, messages were represented as strings (or lists of strings) which specified word forms and word order, but not punctuation, capitalisation, and inter-token white space. The document plan also specified how messages were grouped into higher-level structures (such as paragraphs); discourse relations between messages or groups of messages; and the importance of each message and message group.

Once an initial document plan had been produced, the Document Trimmer component of the document planner attempted to ensure that the document produced by the document plan did not exceed four A5 pages. It did this using a heuristic function which estimated the size of the final document from the document plan. If the heuristic size-estimator indicated that the document was too large, the trimmer identified the least important message in the document plan, deleted this message, and recalculated the document's estimated size.[1] This process continued until the document fitted on four A5 pages according to the size estimator. At this point the document plan was passed on to the other stages of the system, microplanning and realisation. These performed tasks such as deciding when discourse relations should be expressed via cue phrases, and adding appropriate punctuation, capitalisation, and white space to the text (both of which tasks, incidentally, are affected by trimming and hence must take place after it). The realiser produced an RTF file which was printed using Microsoft Word; in a sense Word could be considered to be a fourth pipeline stage.

The main difficulty in this approach was estimating the size of the final document. Since messages were represented as strings, we initially thought it would be easy to build an accurate size estimator. But in fact this proved to be a difficult task, because the size of a document is highly dependent on its exact surface form, including cue phrases, punctuation and capitalisation, and even typographic features such as bold face.

For example, consider the leaflet extract shown in Figure 1. This fits on two A5 pages, as desired. However, if "*bad for your health*" in the paragraph just below the graphic were changed from italic face to bold face, then this paragraph would require four lines instead of three lines. Our layout style does not allow a section to start on a page unless both the section header and two lines of section text can fit on the page. Therefore, increasing the size of this paragraph to four lines causes Word to start the section headed "**You could do it...**" on the next page; this makes the the leaflet overflow onto an additional page, and thus violate the overall size constraint.

Thus, a very small change in a document (such as changing a few words from italics to bold) can cause significant changes in a document's size. The fact that a document's size is so sensitive to its exact surface form is what makes size estimation difficult.

As a result of such problems, although the size estimator soon grew in complexity considerably beyond what we had originally intended, it still made mistakes. In most cases it was fairly accurate, but it was not 100% accurate on 100% of the documents.

As the estimator grew in complexity, another problem appeared, which was the diffi-

1 This is a simplification, as the trimmer also considers dependencies between messages and the importance of message groups. Trimming is in essence a type of bin-packing, and no doubt there is scope for improving the trimmer by incorporating into it sophisticated bin-packing algorithms.

culty of keeping it up-to-date. A clinical trial of the STOP system started in October 1998, and in the months immediately preceding the trial, numerous bug fixes and improvements were made to STOP by the development team. Some of these changes impacted the size estimator, but developers did not always update the size estimator accordingly. In part this was because updating the size estimator in some cases required considerably more work than making the actual bug fix or improvement, and the developers had many urgent changes that needed to be made to the core software in this period.

In other words, another difficulty with building an 'estimator' which predicted the behaviour of the microplanner, realiser, and Word was that it was difficult and time-consuming to maintain the accuracy of the estimator as changes were made to the microplanner and realiser, and also to the exact RTF structures produced by our system for Word to process.

4 Experimental Results

STOP is currently being tested in a clinical trial, in order to determine its effectiveness in helping people stop smoking. The version of STOP used in the clinical trial had a single-solution pipeline architecture as described above. Its trimmer used a size estimator which was tuned to be conservative (and hence often produced leaflets which were smaller than they could have been), but still in a few cases underestimated true length and hence resulted in leaflets which were five A5 pages instead of four. Such leaflets were manually fixed by the researchers running the trial, usually by adjusting the formatting of the leaflet (for example, margins or inter-paragraph separation). We felt this was not acceptable for a production version of STOP, however; such a system should guarantee conformance to the length constraint without needing manual intervention. Also, conformance should be achieved by adjusting content, not formatting. The formatting of STOP leaflets was designed by an expert graphic designer with the goal of enhancing readability, and we believed it should be treated as fixed, not variable.[2]

In order to explore what should be done in a production version of STOP, we conducted some experiments (after the STOP clinical trial had started) on the impact of different architectures on satisfying the size constraint while utilising as much as possible of the available space. For these experiments, we took the version of the system used in the clinical trial (including accumulated bug fixes and enhancements), and re-tuned the size estimator to take into account these accumulated changes. After re-tuning, STOP produced leaflets that fit the size constraint for all members of a 'tuning set' of 150 questionnaires. Then we made the following changes:

- A 'delta' parameter was added to the size estimator; essentially a delta of N made the estimator think that a page could hold N more lines of text than it could in reality contain.
- A 'multiple-solution' mode was added to the system. In this mode, the trimmer is run several times, at different delta values. The resultant document plans are processed by the rest of the system and by Word, and a choice module picks the resulting document which has the highest word count while still satisfying the size constraint.
- A 'revision' mode was added to the system. In this mode, the system generated an initial document using a fixed delta. Then, a revision module obtained the

2 Similarly, I believe the editors of *Computational Linguistics* would not be pleased if I submitted a squib which conformed to the eight page size limit by using non-standard margins or line spacing.

delta	−2	−1	0	1	2	3	4	5	6
size constraint violations (%)	0	0	0.04	0.97	7.3	16	25	35	42
avg word count (legal leaflets)	303	320	336	350	359	364	373	375	378
word count as % of revision mode	79	84	88	92	93	96	98	98	99

Table 1
Results of single-solution pipeline mode

architecture	avg word count	avg size (% of revision)	avg proc time
1 solution (delta = −1)	320	84	2.2s
2 solutions (deltas = −1, 4)	369	96	5.2s
3 solutions (deltas = −1, 2, 6)	378	98	5.9s
4 solutions (deltas = −1, 2, 4, 6)	380	99	6.2s
revision	385	100	9.8s

Table 2
Performance of different modes when meeting the size constraint in 100% of cases

actual size of the document from Word, and either deleted an additional message (if the document was too large) or restored the last deleted message (if the document met the size constraint). This process continued until the system found the largest document which met the size constraint.[3]

The modified system was run on a set of 1000 questionnaires from the clinical trial, in the original single-solution pipeline mode, in the multiple-solution pipeline mode, and in revision mode. For the pipeline modes, the system was run with the deltas −2, −1, 0, 1, 2, 3, 4, 5, and 6. Measurements were made of

- The percentage of leaflets which exceeded the size constraint.
- For leaflets satisfying the size constraint, the average number of words in the two inside pages, both as an absolute number and as a percentage of the number of words in the inside pages when processed under revision mode.
- The average processing time (total elapsed time, not just computation time) required per document, on a Pentium 266MHz with 128MB of memory.[4]

These results are shown in Tables 1 and 2. For multiple-solution pipelines, we tried all pairs, triples, and quadruples of deltas between −2 and 6, and in Table 2 only show the results for the pair, triple, and quadruple which led to the highest average word count while always satisfying the size constraint. We also ran STOP on the full set of 2582 clinical-trial questionnaires in single-delta mode with deltas of −1, 0, and 1, in order to get a more accurate estimate of the number of constraint violations under these deltas.

3 Our revision module did not give any guidance as to where messages should be added. This sometimes led to wasted space in situations where a message could be added to one part of the leaflet but not others (for example, to the first inside page but not the second), if the next message in the undelete list was in a portion of the leaflet which had no unused space.

4 This measurement was made on a subset of 100 documents, because this is the size of collection that STOP was designed to be able to process in one run. While the core NLG system could process any number of documents, the support code (user-interface, logging, file management) worked poorly when processing more than 100-200 documents in one run. For word count and constraint violation data, we simply restarted the system if it hung when processing 1000 questionnaires; but this seemed less appropriate for execution time data.

5 Discussion of Results

As expected, the single-delta figures show that as the delta increases, both the average word count and the number of leaflets that exceed the size constraint also increase. Note that although none of the leaflets produced from the 150-questionnaire 'tuning set' violated the size constraint with a delta of 0, one leaflet produced from the full 2582 questionnaire data set did break the size constraint at this delta. This is perhaps not surprising, it merely shows that as the size of the document set increases, so does the worst-case performance of the heuristic size estimator. It is possible that in a very large data set (hundreds of thousands of questionnaires), some leaflets might break the size constraint even at a delta of -1.

Shifting to a multiple-solution pipeline dramatically improves performance. Average leaflet size while guaranteeing conformance to the size constraint jumps from 320 words in single-delta mode to 369 with two solutions; an increase of 15% in the number of words in the leaflet. We get still better results with three and four solutions, although the increase is not as dramatic. The best results of all are in revision mode, although the increase in size over a four-solution pipeline (385 words vs. 380 words) is small. However, revision mode also is robust in the face of increased data set size (we can be confident that the size constraint will be satisfied even on a set of a million questionnaires) and 'last-minute' changes to the code. If developers tweak the main STOP code and forget to update the size estimator, revision mode will still always produce documents which conform to the size constraint; it just may take longer to do the revision. In contrast, changes to the code may result in the multiple-solution pipeline producing documents which do not conform to the size constraint.

As expected, processing time is lowest for the single-solution pipeline and highest for revision mode. However, in the context of STOP, even the 9.8 seconds required in revision mode is acceptable; under this mode a batch of 100 leaflets can still be generated in under 20 minutes.

6 Implications

In STOP, the single-solution pipeline does a poor job at meeting the size constraint while utilising as much of the available space as possible. No doubt the performance of the single-solution pipeline could be enhanced by adding more complexity to the size estimator; but such a system still would not give 100% accurate estimates on 100% of the generated documents. Furthermore additional complexity would make the estimator harder to maintain as changes were made to the code being estimated.

Both the multiple-solution pipeline and revision mode do a much better job of utilising the available space while observing the size constraint. Revision mode does better than multiple-solution pipeline, but only slightly. However, revision mode is robust in the face of increased data set size and changes to the code.

The effectiveness of multiple-solution pipelines should perhaps not be surprising, given the popularity of such pipelines in other areas of speech and language processing. For example, in a speech system a word-level analysis component may pass several word hypotheses to a language model; and in a natural-language analysis system, a morphology system may pass several possible analyses of a surface form word to a parser. However, multiple-solution pipelines have not received a great deal of attention in the NLG community. I am not aware of any previous NLG papers which presented experimental data comparing single-solution to multiple-solution pipelines, and many NLG pipeline critics (including Danlos) assume that pipeline modules only produce one solution.

Do these results generalise to other constraints and optimisations? In principle, it

seems that similar findings should apply to other constraints and optimisations which depend on decisions or measurements made in more than one module. However, a big caveat is that many of the constraints and optimisations important to NLG systems are difficult to measure, which may lessen the benefits of complex architectures. For example, an important constraint in STOP is that texts should be easy to read for poor readers. However, the only computational mechanism we are aware of for measuring reading difficulty is reading-level formulas (such as Flesch Reading Ease), whose accuracy is doubtful (Kintsch and Vipond, 1979). Without reliable global measures of readability, perhaps the best we can do (and the approach adopted in STOP) is to design messages which readability experts think are appropriate for poor readers; this is something which can be done in a single-solution pipeline architecture.

In other words, if we cannot properly measure the thing we are trying to optimise or satisfy (which may be the case with the majority of constraints and optimisations that today's NLG systems builders are concerned with), then there may be little value in shifting to a complex architecture which supports more sophisticated search (which is perhaps the main benefit of revision and multiple-solution pipelines). This may explain the continuing popularity of single-solution pipeline architectures in applied NLG systems.

Acknowledgments

Many thanks to the STOP team and especially Roma Robertson, who kept on producing examples of STOP leaflets which the size estimator had difficulties with. My thanks also to Michael Elhadad, Chris Mellish, Vibhu Mittal, Daniel Paiva, and the anonymous reviewers for their very helpful comments; and a special thanks to Stephan Busemann for suggesting we investigate multiple-solution pipelines. This research was supported by the Scottish Office Department of Health under grant K/OPR/2/2/D318, and the Engineering and Physical Sciences Research Council under grant GR/L48812.

References

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

Danlos, Laurence and Fiammetta Namer. 1988. Morphology and cross dependencies in the synthesis of personal pronouns in Romance languages. In *Proceedings of the 12th International Conference on Computational Linguistics (COLING-88)*, volume 1, pages 139–141.

Elhadad, Michael and Jacques Robin. 1997. SURGE: A comprehensive plug-in syntactic realisation component for text generation. Technical report, Computer Science Dept, Ben-Gurion University, Beer Sheva, Israel.

Kintsch, Walter and Douglas Vipond. 1979. Reading comprehension and readability in educational practice and psychological theory. In Lars-Göran Nilsson, editor, *Perspectives on Memory Research.* Lawrence Erlbaum, pages 329–365.

Mittal, Vibhu, Johanna Moore, Guiseppe Carenini, and Steven Roth. 1998. Describing complex charts in natural language: A caption generation system. *Computational Linguistics*, 24:431–467.

Paiva, Daniel. 1998. A survey of applied natural language generation systems. Technical Report ITRI-98-03, Information Technology Research Institute, University of Brighton, UK.

Reiter, Ehud. 1994. Has a consensus NL Generation architecture appeared, and is it psycholinguistically plausible? In *Proceedings of the Seventh International Workshop on Natural Language Generation (INLGW-1994)*, pages 163–170.

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

Reiter, Ehud, Roma Robertson, and Liesl Osman. 1999. Types of knowledge required to personalise smoking cessation letters. In Werner Horn et al., editors, *Artificial Intelligence and Medicine: Proceedings of AIMDM-1999*, pages 389–399. Springer-Verlag.