A Dynamic Programming Approach to Document Length Constraints

Keith Vander Linden
Department of Computer Science
Calvin College
Grand Rapids, MI 49546, USA
kvlinden@calvin.edu

Abstract

Natural language generation (NLG) applications must occasionally deliver rhetorically coherent output under length constraints. For example, certain types of documents must fit on a single webpage, on a cell phone screen, or into a fixed number of printed pages. To date, applications have achieved this goal by structuring their content as a rhetorical tree and using a greedy algorithm to pick the discourse elements to include in the final document. Greedy algorithms are known to pick sub-optimal solutions. This paper presents an alternate approach based on dynamic programming.

1 Document Length Constraints

A number of language engineering applications have addressed the issue of generating coherent documents under length constraints, including NLG applications, e.g., SciFly (Paris, *et al*, 2008), STOP (Reiter, 2000), ILEX (O'Donnell, 1997), and summarization applications, e.g., Daniel Marcu (1999). These applications all address the issue by representing the content to be delivered as a rhetorical tree and using some formulation of a greedy algorithm that satisfies the length constraints by either selecting the most important elements of the tree or pruning the least important elements.[1]

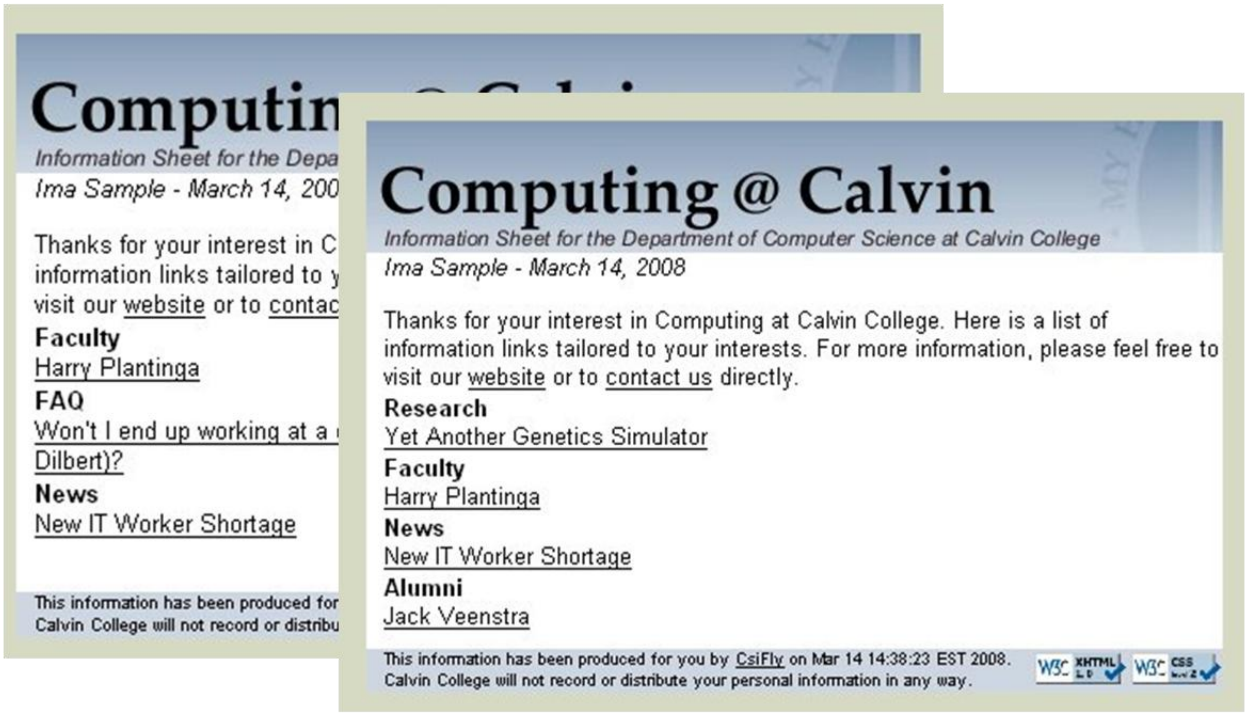
As an example, consider the two sample outputs shown in Figure 1. Both outputs were produced by a prototype that delivers information about a computer science department to prospective students via email; cf. (Paris, *et al*, 2008). The output is composed of coarse-grained elements (e.g., images, phrases and paragraphs) and is formatted in post-card size using HTML and includes hyperlinks to related pages on the main department website. The goal is to get the prospective student to visit the main website. The key difference between the two examples is their length. The one on the left, which is shorter, was generated using a greedy algorithm. The one on the right, which uses the space more fully, was generated using a dynamic programming algorithm. The greedy algorithm included the "FAQ" section because it fit at the time; the dynamic algorithm realized that waiting to include the "News" and the "Alumni" sections would be a more effective use of the space.

This paper discusses the mechanisms used to generate these two examples. It starts with a discussion of the rhetorical tree used as input for both sample outputs, and then details the algorithms used to satisfy the length constraint. The length constraint problem is viewed as a precedence-constrained 0-1 knapsack problem, and the algorithm is formulated using dynamic programming.

2 Rhetorical Structure Trees

Researchers have long viewed rhetorical trees as a means of structuring textual output and of distinguishing those elements that should be expressed (often called nuclei) from those that could be omitted (often called satellites) (Spark-Jones, 1993), and there is psycholinguistic evidence that this view is valid (Marcu, 1999). This paper will

[1] The STOP system identifies the problem as a bin-packing problem but then describes its mechanism using terms common to greedy algorithms (Reiter, 2000).


Computin

Information Sheet for the Depa

Ima Sample - March 14, 200

Thanks for your interest in C
information links tailored to y
visit our website or to contac

Faculty

Harry Plantinga

FAQ

Won't I end up working at a
Dilbert)?

News

New IT Worker Shortage

This information has been produced for
Calvin College will not record or distribu

Computing @ Calvin

Information Sheet for the Department of Computer Science at Calvin College

Ima Sample - March 14, 2008

Thanks for your interest in Computing at Calvin College. Here is a list of information links tailored to your interests. For more information, please feel free to visit our website or to contact us directly.

Research

Yet Another Genetics Simulator

Faculty

Harry Plantinga

News

New IT Worker Shortage

Alumni

Jack Veenstra

This information has been produced for you by CsiFly on Mar 14 14:38:23 EST 2008.
Calvin College will not record or distribute your personal information in any way.

Figure 1. Two Sample Outputs – The greedy output is on the left, the dynamic is on the right.

build its trees using Rhetorical Structure Theory (RST), the most common of the theories deployed for rhetorical structuring (Mann & Thompson, 1988).

Figure 2 shows the annotated RST tree used as the basis for generating both of the sample outputs shown in Figure 1. The highest level of the tree shows a template that affixes the header image as a preparation satellite and the footer as a background satellite. The next level down shows the structure of the content of the email. Each node is associated with a block of text or an image in the output. The initial line (i.e., "Ima Sample…") is represented as a satellite setting the context for the main content in the nucleus (i.e., "Thanks for your interest…"). There then follows a set of six elaboration satellites, each with internal structures of its own (i.e., an image/caption pair, which is not realized in either sample output because of its cost, and a set of five topic/hyperlink pairs).

Each expressible node in the figure has an estimated length, denoted as a cost *c* measured in pixels, and an estimated importance, denoted as a benefit *b*. The cost of an expressible node is estimated based on the size of the text/image and specified format. Generally, the leaves are the expressible nodes, but in the case where multiple leaves should be expressed together, the RST tree places a combined cost value in the parent node (e.g., the header and footer should be expressed together or not at all, so their parent node shows their combined cost of 100 pixels).

The benefit setting, denoted *b(n,D)*, for a node *n* in a discourse tree *D* is formulated as follows:

$$b(n,D) = \begin{cases} 1.0 & n \text{ is the root} \\ b(P(n),D) & n \text{ is a nucleus} \\ b(P(n),D) \times W(r) & n \text{ is a satellite} \end{cases}$$

Here, *P(n)* denotes the parent of node *n* and *W(r)* denotes the weight of the satellite's rhetorical relation (e.g., in this paper, more important relations like context have weight 0.5 and less important relations like preparation, elaboration, background have weight 0.4). This formulation gives higher benefit values to nuclei. No penalty is placed on nodes lower in the tree. This formulation is implemented as a recursive decent algorithm whose values for the samples can be seen in Figure 2.

The tree is produced by a Moore and Paris-styled text planner (Moore & Paris, 1993). Selecting the appropriate content from a content management system, structuring it appropriately and estimating the cost/benefit of each node are interesting problems, but are not the focus of this paper.

In preparation for the length constraint algorithm, the system creates a queue of expressible nodes, each with its cost and benefit values. Nuclei are queued before satellites.

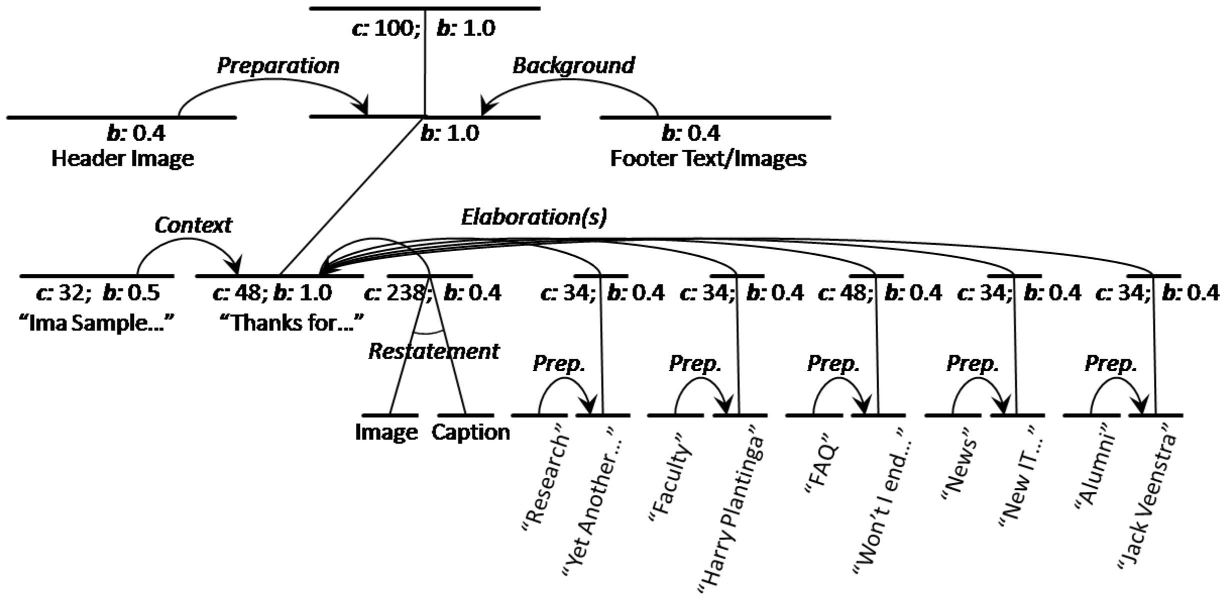


Figure 2. Sample Discourse Tree for the Output in Figure 1 with Cost/Benefit Settings

3 A Length Constraint Algorithm

This paper views the length constraint problem as a precedence-constrained 0-1 knapsack problem (Samphaiboon & Yamada, 2000); the output images/paragraphs are viewed as items for the knapsack, their length in pixels is viewed as their cost, and their rhetorical importance is viewed as their benefit. The prohibition against expressing a satellite without also expressing its governing nuclei is viewed as a precedence constraint on the items.

A common formulation for this problem is to define the solution in terms of the maximum benefit for a subset of the first *k* nodes given a maximum cost *c* as follows:

$$B(k,c) = \begin{cases} B(k-1,c) & c_k > c \\ \max\,(B(k-1,c), & c_k \le c \\ \qquad B(k-1,c-c_k)+b_k) & \end{cases}$$

Here, b_k and c_k are the benefit and cost of node k respectively, and b_k is defined by *b(n, D)* above. If the node *k* will fit within the maximum cost constraint *c*, then *B(k, c)* is defined as the maximum of either the:

- previous solution for the first *k-1* nodes not including node *k*; or
- previous solution with space for node *k*.

A dynamic programming algorithm that implements this formulation can be specified as follows:

Function: *format(C, Q, T)*
Input:

- A positive integer maximum cost limit *C*
- A queue *Q* of *N* nodes with positive integer cost (c_i) and real benefit (b_i) settings
- A tree *T* specifying rhetorical relationships

Output:

- A 2-D array *B*[*n+1, c+1*] specifying the values for *B(k,c)* as defined above

```
for $c \leftarrow 0$ to $C$
    $B[0,c] \leftarrow 0$
for $k \leftarrow 1$ to N
    for $c \leftarrow 0$ to $C$
        $B[0,c] \leftarrow B[0,c-1]$
    for $c \leftarrow c_k$ to $C$
        if (not unexpressedNucleus(B,c,k,T) and
            $B[k-1,c-c_i] + b_i > B[k-1,c]$)
                $B[k,c] \leftarrow B[k-1,c-c_i] + b_i$
```

The *format(C, Q, T)* algorithm declares a 2-D array of maximum benefit values and sets the first row to 0s. For each expressible node *k* in *Q* (rows *1* through *N*), it copies the previous row of benefit values and then, for each cost value above the cost of the current node (columns c_k through *C*), it either keeps the previous benefit value without node k or inserts the benefit that includes node *k*.

Given the array of maximal benefit values output by *format(C, Q, T),* the following algorithm will compute set of nodes corresponding to a given benefit value:

```
Function: output(B, k, c)
Input:
    • An array B of maximal benefit values
    • A node number k
    • B's maximum cost value c.
Output:
    • The set of nodes with total cost c and total benefit B(k, c).

while k,c > 0
    if $B[k,c] \neq B[k-1,c]$
        include $node_k$
        $c \leftarrow c - c_k$
    $k \leftarrow k - 1$
```

The use of *unexpressedNucleus(B,c,k,T)* in the *format(C, Q, T)* if-condition is an extension of the standard algorithm for the 0-1 knapsack problem that addresses rhetorical precedence constraints. In RST-based NLG, satellites are not expressed without their nuclei. For example, we would not want to include the context expression (i.e., "Ima Sample…") without also including its nucleus (i.e., "Thanks for…"). Note that these "governing" nuclei are not always at the same level in the tree (e.g., the header image satellite is dependent upon the "Thanks for…" nucleus one level down).

The *unexpressedNucleus(B,c,k,T)* condition implements this constraint by requiring that $G(n,T) \subseteq output(B,k,d)$ before including any node *n*, where *G(n,T)* is the set of governing nuclei for *n* in tree *T* and is formulated as follows:

$$G(n,T) \quad = \begin{cases} null & n \text{ is a root} \\ GN_{+}(P(n),T) \cup GN_{-}(P(n),T) - n \end{cases}$$

$$G_{+}(n,T) = \begin{cases} null & n \text{ is not a root} \\ GN(P(n),T) & n \text{ is a root} \end{cases}$$

$$G_{-}(n,T) = \begin{cases} n & n \text{ is a leaf} \\ \displaystyle\bigcup_{c_i \in NC(n)} G_{-}(c_i,T) & n \text{ is not a leaf} \end{cases}$$

Here, *G_+()* looks up the tree, *G_-()* looks down the tree, *P(n)* is *n*'s parent node, and *NC(n)* is the set of *n*'s nucleus children. *G()* includes only expressible nodes. *G()* can be implemented by a set of three mutually recursive functions and can be memoized to improve efficiency.

The greedy algorithm used for the output in Figure 1 uses the same input and a precedence queue of expressible nodes ordered by decreasing benefit.

4 Analysis and Conclusions

The dynamic programming algorithm will always perform at least as well as the greedy algorithm, and sometimes better. For example, given a total cost maximum of 325, the greedy algorithm's output in Figure 1 has total cost/benefit: 297/3.7, while the dynamic algorithm's output has 316/4.1.

Dynamic programming algorithms are notoriously expensive in terms of space and time requirements. They are pseudo-polynomial time, O(NC), but if N and C are "small" they can work in practice. Typical document formatting problems with dozens of expressible nodes and hundreds of pixels of length are tractable.

Further work on this project will follow a number of directions, including: (1) doing a more complete quantitative analysis of the algorithm; (2) figuring out a more principled way to assign benefit values; (3) generalizing the problem to two dimensions and multiple pages; (4) drawing the content from a content management system.

References

Mann, W. C., & Thompson, S. A. (1988). Rhetorical structure theory: Toward a functional theory of text organization. Text , 8 (3), 243-281.

Marcu, D. (1999). Discourse trees are good indicators of importance in text. In I. Mani, & M. Maybury (Ed.), Advances in Automatic Text Summarization (pp. 123-136). MIT Press.

Moore, J. D., & Paris, C. L. (1993). Planning text for advisory dialogues: Capturing intentional and rhetorical information. *Computational Linguistics* , *19* (4), 651-694.

O'Donnell, M. (1997). Variable Length On-Line Document Generation. *Proceedings of the Sixth European Workshop on NLG.* Duisburg, Germany: Gehard Mercator University.

Paris, C. L., Colineau, N., Lampert, A., & Giralt Duran, J. (2008). *Generation under Space Constraints.* Sydney: CSIRO.

Reiter, E. (2000). Pipelines and Size Constraints. *Computational Linguistics* , *26* (2), 251-259.

Samphaiboon, N., & Yamada, T. (2000). Heuristic and Exact Algorithms for the Precedence-Constrained Knapsack Problem. *Journal of Optimization Theory and Applications* , *105* (3), 659-676.

Spark-Jones, K. (1993). What might be in a summary? In Knorz, Krause, & Womser-Hacker (Ed.), *Proceedings of Information Retrieval 93: Von der modellierung zur anwendung* (pp. 9-26). Universitatsverlag Konstanz.