Tailoring Text for Automatic Layouting of Newspaper Pages

Leonhard Hennig
DAI Labor, TU Berlin
Berlin, Germany
leonhard.hennig@dai-labor.de

Thomas Strecker
DAI Labor, TU Berlin
Berlin, Germany
thomas.strecker@dai-labor.de

Abstract

We consider layouting news articles on a page as a cutting and packing problem with output maximization. We propose to tailor news articles by employing automatic summarization to find new solutions for the packing problem. Tailoring text items allows us to use an efficient ϵ-approximate greedy layouting algorithm, which scales well for larger data sets, to explore the search space. We also propose a function for adjusting the value of summarized articles. Our results show that the overall solution value as well as the individual quality of articles are notably improved, with the solution value in some cases even exceeding the optimum score achievable by an exhaustive search without summarization.

1 Introduction

Automatic layouting is the task of placing items on a two-dimensional plane such that the resulting layout fulfills a set of external constraints. Often, these constraints take the form of specifying the size of the plane that is available for layouting, as well as requiring that the resulting layout takes up as much of the given area as possible. A layouting algorithm therefore has to determine which items and which arrangement of items optimally fill the available area. This general class of problems is known as "Cutting and Packing" (C&P) which is an NP-hard problem [5]. The task is to constrain the search space of the layouting algorithm in such a way as to still allow for nearly-optimal layouts, without the need to iterate over all possible combinations of item selection and placement options.

Simple approaches which already achieve good results employ approximations [15] and heuristics [1]. Another group of approaches adapts various general AI algorithms to the task, e.g. greedy or relative greed algorithms [8], genetic algorithms [3], and simulated annealing [6], which create, and subsequently alter, valid layouts in order to optimize the result.

We further constrain the problem by assigning a value to each item. This transforms the problem into an output maximization problem, where the algorithm must not necessarily minimize wasted space, but maximize the overall value of the packed items. We consider the C&P problem in the context of layouting news articles to create a newspaper page. To produce a visually pleasing layout which conforms as much as possible to a typical newspaper page layout, the algorithm must consider aesthetic and design criteria, such as wasted space on the page. An item's value is thus determined not only by its inherent value but also by its contribution to an aesthetically pleasing overall page layout.

Typically, even optimal solutions will contain some wasted space, e.g. because none of the available items fits the remaining space. Since we are dealing with text documents, we propose to employ *automatic text summarization* to adapt items to the available space. This solution is most useful when employed in combination with a density-based greedy algorithm, where we know that all items in the remaining search space have a smaller density than the current item.

We consider a second application of summarization in this paper, namely increasing the coverage area of an item. In our approach items are laid out in a rectangular (sub)space of a fixed-size page grid. An item's text may therefore not suffice to completely fill its final row of grid cells. We employ text summarization to search for a condensed version of the item that increases coverage in a subspace of the originally allocated grid space, i.e. by reducing the height of the item.

Text summarization is the task of creating a text from one or more textual sources that is smaller in size but retains some or most of the information contained in the original sources [10, 12]. How much and which information, as well as other characteristics of the source document(s), are kept, depends on the intended use. Research in text summarization has largely focused on determining the salience of sentences based on e.g. location [4], term prominence [11], or rhetorical structure [13]. Recently, graph-theoretic features [14] as well as semantic features have attracted attention [10]. Most approaches create a summary by scoring and ranking sentences using a linearly weighted combination of the features. In [9, 10], a Naive Bayes and an SVM classifier are trained on the task of separating summary sentences from non-summary ones. For a more complete overview of the state of the art in automatic text summarization, see [7].

The rest of this paper is organized as follows: Section 2 describes the summarization component, section 3 describes the layouting algorithm and the integration of the summarizer. In section 4 we desribe our data sets, and discuss the evaluation methodology and our results. Finally, section 5 concludes, and gives an outlook on our planned work.

2 Summarizing News Articles

We employ an extractive approach to summarization where each sentence from the article's text is assigned a score indicating the relevance of this sentence for a potential summary. Sentences are ranked by this score, and selected for inclusion in the summary until the desired text length as specified by the layouting algorithm is reached. We represent sentences as a set of features, and compute a sentence's score as the linearly weighted combination of these features. After extracting sentences and terms from the input text, and applying stemming and stop word removal, the following features are computed for each sentence: sentence-document cosine distance, sentence position, and sentence length. The sentence position feature is normalized to [0,1], and sentence length is a binary feature set to 0 if the sentence length is smaller than a fixed threshold, and 1 otherwise. We restrict the approach to these well-known features (see e.g. [12]) since the focus of this paper is not on the quality of the computed summary.

Additionally, we use a measure similar to MMR [2] to penalize redundant sentences. The redundancy feature is computed as the cosine distance of a sentence to the current summary, and is updated for all remaining candidate sentences each time a sentence is added to the summary, followed by a reranking of the candidate sentences. Feature weights were optimized one at a time, while holding the other weights fixed.

3 Layouting

In our scenario, a page is subdivided into a fixed-size 4-by-16 grid, where each cell corresponds to 6 lines of text (cf. Figure 1 (a)). An article itself may be laid out in several variants depending on article properties such as text length and/or the availability of media elements. Therefore, each article is transformed into a set of candidate items for the layouting process, discarding variants which do not fulfill basic aesthetic constraints. There are at most 8 items per article, corresponding to an article layout across 1, 2, 3 or 4 columns, combined with using a media element or not. The layout algorithm is responsible for ensuring that at most one item of each set is used.

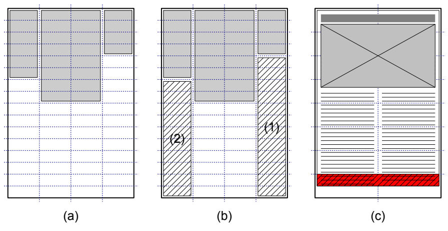

Figure 1. Layout grid and article coverage

We implemented an ϵ-approximate greedy layouting algorithm based on [15]. Greedy algorithms do not generate all possible successor states when exploring the search space. Starting at a local optimum and local information these algorithms at each step create a single successor solution ($\mathcal{B}$). Simple strategies for finding the successor solution employ sorting the items in order of decreasing value or density.

A drawback of greedy strategies is that they are unable to "undo" a partial solution, and thus can get stuck in a local optimum. ϵ-approximate algorithms offset this by initializing the solution with permutations of at most k items from the set $\mathcal{C}$ of candidates before applying the greedy scan for the remaining set. For our experiments, we set $k = 1$. After iterating over all permutations, the algorithm selects the best of the $|\mathcal{C}|$ solutions. We sort by the density of an item, which is computed as

$$d_i = \frac{v_i}{weight_i} = \frac{f(i)}{w_i * h_i}, \tag{1}$$

where v_i is the value of an item as computed by a fitness function $f(i)$, and $weight_i$ is the weight of an item, i.e. its area. If the density of two items is identical, they are sorted in order of decreasing area.

Our fitness function evaluates three criteria for scoring an item: User relevance as determined e.g. by a newspaper personalization component is the 'inherent' value of an item, item coverage is an aesthetic criterion with respect to our grid-based approach, and fractional page coverage is the global C&P-related criterion. An item's value is computed as the weighted sum:

$$f(i) = r_i + c_i + \frac{w_i * h_i}{W * H - \sum_{j \in \mathcal{B} j \neq i} w_j * h_j}$$

where i is the item to score and r_i and c_i are the relevance and the coverage of the item. The page score $p(\mathcal{B})$, i.e. the overall value of the solution, is then computed as the sum of all item values.

If an item is summarized, its information content is lowered, and therefore its user relevance r_i should be adjusted. The score that a sentence is assigned by the summarizer can be interpreted as a measure of the information content of that sentence. We therefore reduce the user relevance contribution of a summarized item by computing the summed score of the sentences contained in the condensed item, as a fraction of the summed score of the original item's sentences. This fraction is then multiplied by the original user relevance r_i to give the updated user relevance $\tilde{r}_i$:

$$\tilde{r}_i = \frac{\sum_j score(S_{ij})}{\sum_k score(T_{ik})} r_i, \qquad (2)$$

where S_i is the set of sentences of the summarized item, and T_i is the set of original sentences of item i. Sentences with low information content that are not included in the summarized item infer only a small penalty on the updated user relevance contribution.

Items are placed on the page in a left-to-right and top-to-bottom fashion. If an unoccupied space is detected the extent of the rectangular empty area is computed and returned as the area where the next item can be placed ((cf. Figure 1 (b), box 1). Additionally, the layout algorithm can skip areas of the page that no item fits, resulting in new spaces (cf. Figure 1 (b), box 2).

If a candidate item's height is greater than the available height, the summarizer is invoked to generate a condensed version of the item. To minimize within-item whitespace, the summarizer is also invoked if the text coverage of the item's last row of cells is less than a specified threshold. We chose this threshold to be half a cell height. A summarized version is accepted if its final cell text coverage is greater than for the original version. Figure 1 (c) shows a 2x8 item with a media element, where the available text does not fill the last cell row. The item is summarized to create a 2x7 item with improved coverage.

4 Experiments and Results

We prepared data sets by randomly sampling articles from a corpus of news articles. From each data set, we processed the first $n = 10$ or 20 articles, resulting in candidate item sets $\mathcal{C}$ containing between 31 and 104 items. Most sets include at least one full-page article, and a number of item combinations that can be laid out to completely fill the page. Table 1 shows the results obtained by the ϵ-approximate algorithm with and without summarization. The results are given for each data set and input article set size. As can be seen, scores achieved when using summarization are on average higher than without summarization. We have highlighted the runs where the score of the summarization variant exceeds the optimum value obtained when using an exhaustive search on the data set. The page score decreases for only 3 of the 20 runs. We attribute this to the reduction in user relevance resulting from a summarization for increasing article coverage. This can be seen in Table 2, where we show the amount of wasted space, comparing the baseline and the summarization variant of the algorithm. The column entries correspond to the sum of 'blank' text lines of all articles ('Art'), and the number of empty grid cells of the page ('Pg'). For runs 1-20 and 8-20, the number of blank text lines decreases considerably.

We observe that for a number of runs, the summarization variant manages to completely fill the page, indicated by a value of 0 for the number of empty cells in the corresponding row. We also note that when using summarization the number of empty cells decreases

Dataset			Opt	w/o sum		w/ sum			
#	n	$\lvert\mathcal{C}\rvert$	$p(\mathcal{B})$	$p(\mathcal{B})$	$\lvert\mathcal{B}\rvert$	$p(\mathcal{B})$	$\lvert\mathcal{B}\rvert$	#S	%Δ
0	10	36	2.21	2.11	3	**2.36**	3	1	11.8
	20	83	3.38	3.18	4	**3.99**	5	2	25.7
1	10	41	4.70	3.42	5	**5.23**	7	5	44.6
	20	72	8.04	6.96	10	6.72	10	5	−3.4
2	10	42	6.75	6.12	8	6.29	8	2	2.9
	20	66	9.11	8.39	11	8.01	11	2	−4.5
3	10	43	5.60	5.14	7	5.39	8	2	5.0
	20	81	7.21	5.79	8	6.36	9	3	9.9
4	10	31	6.73	6.32	8	**6.82**	9	3	8.0
	20	70	7.89	7.82	10	**8.64**	10	1	10.4
5	10	44	5.45	4.02	6	4.75	6	1	18.2
	20	91	7.28	5.72	8	6.89	9	2	20.5
6	10	48	5.18	4.50	6	5.11	7	1	13.7
	20	88	7.34	5.93	9	7.34	10	3	23.8
7	10	52	4.32	3.66	5	**5.12**	6	3	39.8
	20	104	4.56	3.62	5	**5.63**	7	4	55.2
8	10	54	5.21	5.01	6	**6.48**	8	5	29.3
	20	86	7.43	7.40	9	6.69	8	4	−9.7
9	10	48	5.77	4.69	7	5.28	8	3	12.6
	20	94	7.81	7.06	9	**8.35**	10	5	11.3

Table 1. Page results with/without summarization

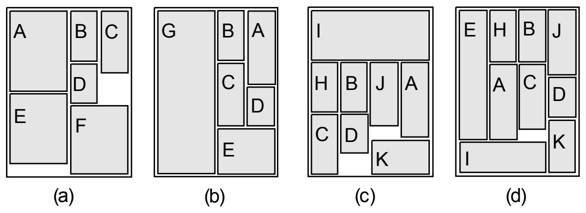

Figure 2. Baseline (a,c) and summarized page (b,d) for runs 5-10 and 5-20

significantly for most runs. It increases for only 4 of the 20 runs, and only by a single cell in these cases. For two of the four runs with an increase in empty cells, there is no obvious corresponding decrease in blank article lines, but as can be seen from Table 1, the corresponding layout solution actually contains more articles than the baseline. The increase in blank article lines is hence due to the blank lines contributed by the extra article. In Figure 2, we show example pages layouted with the baseline and the summarization variant of the algorithm, corresponding to runs 5-10 and 5-20 of our experiments (letters indicate the source article).

5 Conclusion

We have shown that employing automatic text summarization improves the quality of solutions for the problem of layouting news articles to create a newspaper page. Our approach employs an ϵ-approximate greedy algorithm for selecting items, and scales well for larger item sets. We have also proposed a measure to adapt the value of a summarized item according to its information content, and we have employed summarization to improve the aesthetic quality of individual items. The page scores obtained with our approach for some data sets even exceed the optimum value achievable by an exhaustive search. We note that both the baseline and the summarization variant of the algorithm still very much depend on the placement strategy, since they are unable to undo a partial solution. We therefore plan to investigate placement strategies that overcome this drawback, e.g. by providing multiple placement options at each step.

Set		w/o sum		w/ sum		Set		w/o sum		w/ sum	
#	n	Art	Pg	Art	Pg	#	n	Art	Pg	Art	Pg
0	10	11	4	23	4	5	10	17	5	22	**0**
	20	7	2	19	**0**		20	19	3	16	**1**
1	10	35	5	19	**1**	6	10	17	1	18	2
	20	36	3	27	4		20	28	4	30	**1**
2	10	18	2	14	**1**	7	10	15	8	13	**0**
	20	20	1	24	2		20	17	6	23	**2**
3	10	22	2	26	3	8	10	21	1	22	**0**
	20	31	2	29	2		20	28	0	18	0
4	10	31	1	30	1	9	10	18	6	16	**5**
	20	33	1	18	**0**		20	19	2	22	**0**

Table 2. Blank space for page / articles

References

[1] J. E. Beasley. A population heuristic for constrained two-dimensional non-guillotine cutting. *European Journal of Operational Research*, 156(3):601–627, 2004.

[2] J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of SIGIR '98*, pages 335–336, 1998.

[3] J. P. Cohoon, S. U. Hegde, W. N. Martin, and S. Richards. Floorplan design using distributed genetic algorithms. In *IEEE National Aerospace and Electronics Conference*, pages 488–491, 1988.

[4] H. Edmundson. New methods in automatic abstracting. *Journal of the Association for Computing Machinery*, 1969.

[5] E. Goldenberg. Automatic layout of variable-content print data. Master's thesis, Information Infrastructure Laboratory, HP Laboratories, 2002.

[6] J. González, I. Rojas, H. Pomares, M. Salmerán, and J. Merelo. Web newspaper layout optimization using simulated annealing. *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 2002.

[7] K. S. Jones. Automatic summarising: The state of the art. *Inf. Process. Manage.*, 43(6):1449–1481, 2007.

[8] B. A. Julstrom. Greedy, genetic, and greedy genetic algorithms for the quadratic knapsack problem. In *Proceedings of GECCO '05*, pages 607–614, 2005.

[9] J. Kupiec, J. Pedersen, and F. Chen. A trainable document summarizer. In *Proceedings of SIGIR '95*, pages 68–73, 1995.

[10] J. Leskovec, N. Milic-Frayling, and M. Grobelnik. Impact of linguistic analysis on the semantic graph coverage and learning of document extracts. In *Proceedings of the AAAI*, 2005.

[11] H. Luhn. The automatic creation of literature abstracts. *IBM Journal of Research & Development*, 1958.

[12] I. Mani. *Automatic summarization*. John Benjamins Publishing Company, 2001.

[13] D. Marcu. The rhetorical parsing of natural language texts. In *Proceedings of the35th Annual Meeting of the Association for Computational Linguistics*, pages 96–103, 1997.

[14] R. Mihalcea. Graph-based ranking algorithms for sentence extraction, applied to text summarization. In *Proceedings of the ACL 2004*, page 20, 2004.

[15] S. Sahni. Approximate algorithms for the 0/1 knapsack problem. *Journal of the ACM*, 1975.