Web Indexing on a Diet: Template Removal with the Sandwich Algorithm

Tom Rowlands, *Paul Thomas*, and *Stephen Wan*
CSIRO ICT Centre
tom.rowlands@csiro.au, *paul.thomas@csiro.au*, *stephen.wan@csiro.au*

Abstract *Web pages contain both unique text, which we should include in indexes, and template text such as navigation strips and copyright notices which we may want to discard. While algorithms exist for removing template text, most rely on first completing a crawl and then parsing each page. We present a cheap and efficient algorithm which does not parse HTML and which requires only a single pass of the document. We have used two web corpora to investigate the performance of a retrieval system using our algorithm and have found similar eectiveness with an index 9-54% smaller. Further experiments using a marked-up corpus have shown 97% of desired lines are returned.*

Keywords Web documents, information retrieval

1 Introduction

Retrieving information from within a web document is made more difficult by the presence of template text. Such templates include, for example, the header and footer information that sandwiches the real content of the document. These are typically inserted automatically by HTML authoring tools and scripts that dynamically generate HTML pages, in order to provide a website with a consistent look-and-feel. Ideally, an information retrieval system would be able to discard such template material when it doesn't contribute to the topic of a page.

In this paper, we treat template detection and removal as a longest common subsequence (LCS) problem, giving an efficient solution. Our experiments with the WT10g corpus and an enterprise data set demonstrate gains in efficiency with low complexity.

2 Related work

Related work has been characterised as using either a *local* or a *global* approach [3]. A local approach examines a page in isolation to find the template material. In contrast, a global approach determines shared templates by examining two or more documents from a collection.

Most approaches handle templates with a two-pass algorithm: the first pass identifies the template and the second extracts it. Approaches to identifying the template have included structural comparisons, often using the document object model of the HTML document. Tree comparison methods have been used to examine similarities in HTML tag elements [8]. Similarly, Wang et al. [9] look for tables specifying layout. Visual blocks have been segmented using classification approaches [7].

In contrast to examining document structure, other approaches simply examine page text and are thus cheaper to run. Word-level features such as term frequency and word position statistics have been exploited to induce templates [2]. A similar approach using text fragment frequencies is explored by Gibson et al. [3]. Our work is similarly non-structural but does not require any statistical modelling.

3 The sandwich algorithm

We investigate template detection and removal from the viewpoint of improving the efficiency of a web search engine. As such, we start with the constraint that the solution must be able to operate as documents are crawled.

The algorithm is derived from the intuition that, given the prevalence of HTML authoring tools and website content management systems, documents in the same directory will likely share the same template. The template lines are detected by comparing the target file—line by line—with a sibling document in the directory, referred to here as a *peer*. The longest common subsequence (LCS) of lines is a non-contiguous set of lines in common to a document and its peer. Our approach assumes all such lines are from a template and can be discarded. The remaining lines are considered indexable material and kept. If there are no other pages in the directory, and therefore no candidate peers, no template removal is attempted.

Our approach is global but reduces to a single pass. That is, identification of the template is performed per document, and template material is removed at the same time. As a result, this approach can be implemented in a crawler before material is stored. If the crawl is breadth-first, in most cases an appropriate peer will simply be the last page crawled.

Different algorithms produce the LCS in $O(n^2)$ to $O(n \log n)$ time [5, 6], where n is the number of lines in each document. No HTML parsing is required; the algorithm is entirely independent of

Proceedings of the 14th Australasian Document Computing Symposium, Sydney, Australia, 4 December 2009.

the markup language.[1] The algorithm can remove template text from "split" content, where template material is injected in between portions of useful text. Implementation is straightforward and simpler than competing approaches, which makes template removal an option where engineering resources are limited.

4 Experiments

Our early experiments consider two measures. First, we examine the effectiveness and efficiency of a retrieval system which employs the sandwich algorithm. Second, a corpus with templates explicitly marked allows us to investigate our algorithm's accuracy. (These only provide a quick check of the algorithm's performance; in this first work we have not conducted an in-depth comparison with competing, more complex, techniques.)

To investigate the performance of a retrieval system which incorporates the sandwich algorithm, we used PADRE [4]—which implements a variant of BM25—and two corpora. The WT10g corpus, used by the TREC web track [1], includes about 1.7 million web pages from a variety of hosts. Peers were found for 92% of pages. We used three sets of associated queries ("topics"). Topics 451–500 (from TREC 2000) and 501–550 (from TREC 2001) are reverse engineered from search engine query logs. Topics EP1–145, also from TREC 2001, concentrate on finding home pages. Since by removing navigation blocks we will remove a number of links to each site's entry page, performance on this latter set of queries seems likely to degrade.

The second corpus is in the media domain, and was collected from a large, national media organisation's website. It comprises about 760,000 documents for which peers were found for 98%. 88 queries were used from a sample of the organisation's query log, with judgements by an author who was familiar with the organisation. A subset of this corpus has templates explicitly marked.

In these experiments, which used a pre-existing crawl, a page's peer was based on its name n: it was that page in the same directory whose name was closest to n. "Closest" was defined with respect to edit distance.

The first question we ask is: *how much more efficient can an index be if templates are removed?* To our knowledge, template removal approaches have not been examined by this measure. Table 1 summarises the size of each corpus with and without processing; and the number of postings in an index of each.

Since a lot of templates are formatting or scripting instructions, which will not be indexed anyway, the savings in postings are less than the savings in corpus size—however even the smallest saving, 9% of postings for WT10g, seems worthwhile, and the figures for the media corpus represent a substantial savings. The figures for WT10g are smaller than the 40–50% suggested by Gibson et al., but the WT10g crawl is older than the one used there and the use of templates has been growing since [3]. By insisting on exact string matches we are also conservative in identifying possible templates.

[1] If the input is known to be, e.g., HTML or SGML then a tokeniser could be run first and the LCS computed over streams of tokens. We have not yet pursued this idea.

	As-is	LCS removed
WT10g	10.7 GB	9.0 (−17%)
	1.4×10^9 postings	1.2×10^9 (−9%)
media	12.3 GB	4.0 (−67%)
	1.1×10^9 postings	0.5×10^9 (−54%)

Table 1: Corpus and index sizes for two corpora, before and after processing.

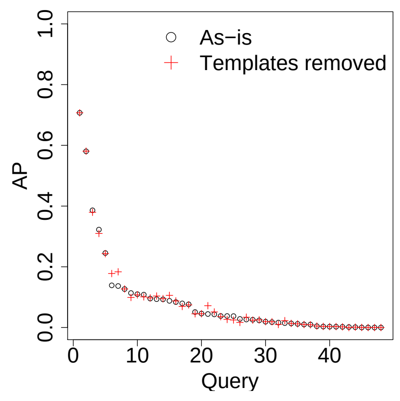

Figure 1: AP scores for queries 451–500, processed with and without templates in the index.

Although a substantial fraction of the index has been removed, retrieval performance is unaffected. Figure 1 illustrates the AP scores for each of topics 451–500: on most queries there is no discernable change and overall there is no significant difference (Wilcoxon $p > 0.99$). Topics 501–550 and EP1–145, and the media set, are similar ($p > 0.2$, $p > 0.5$, and $p > 0.4$ respectively).

A further question is: *how accurate are we in removing templates?* We compared our output, line by line, with a subset of the media corpus explicitly marked by the organisation. Blank lines and contentless HTML (e.g. a sole `<p>` on a line) were not considered in the comparison. The precision and recall of lines classified as non-template material (and hence kept) is 57% and 97% respectively, with an F1 score of 0.59. The algorithm is correctly keeping the great majority of interesting text, although our conservative approach means we are also keeping a portion of templates.

5 Conclusions and Future Work

Templates represent a substantial, though generally uninformative, portion of text on the web. Removing templates leads to a reduction in index size, without a drop in query performance. Line-based LCS comparison provides a cheap method for template detection and removal, allowing for easy integration within a web crawler. In future work, we intend to use the sandwich algorithm with question answering systems and automatic text summarisers, both of which can benefit greatly with the accurate removal of irrelevant template material.

Acknowledgements

We thank the anonymous reviewers for their feedback and for their useful ideas.

References

[1] Peter Bailey, Nick Craswell and David Hawking. Engineering a multi-purpose test collection for web retrieval experiments. *Info Proc & Management*, Volume 39, Number 6, pages 853–871, November 2003.

[2] Laing Chen, Shaozhi Ye and Xing Li. Template detection for large scale search engines. In *Proc. ACM Symposium on Applied Computing*, pages 1094–1098, 2006.

[3] David Gibson, Kunal Punera and Andrew Tomkins. The volume and evolution of web page templates. In *Proc. WWW*, pages 830–839, 2005.

[4] David Hawking, Peter Bailey and Nick Craswell. Efficient and flexible search using text and metadata. Technical Report 2000/83, CSIRO Mathematical and Information Sciences, 2000. `http://es.csiro.au/pubs/hawking_tr00b.pdf`.

[5] Daniel S. Hirschberg. Algorithms for the longest common subsequence problem. *J. ACM*, Volume 24, Number 4, pages 664–675, 1977.

[6] James W. Hunt and Thomas G. Szymanski. A fast algorithm for computing longest common subsequences. *Comm. ACM*, Volume 20, Number 5, pages 350–353, 1977.

[7] Ruihua Song, Haifeng Liu, Ji-Rong Wen and Wei-Ying Ma. Learning block importance models for web pages. In *Proc. WWW*, pages 203–211, 2004.

[8] Karane Vieira, Altigran S da Silva, Nick Pinto, Edleno S de Moura, João M B Cavalcanti and Juliana Freire. A fast and robust method for web page template detection and removal. In *Proc. CIKM*, pages 258–267, 2006.

[9] Yu Wang, Bingxing Fang, Xueqi Cheng, Li Guo and Hongbo Xu. Incremental web page template detection. In *Proc. WWW*, pages 1247–1248, 2008.