# Coping With Noise in a Real-World Weblog Crawler and Retrieval System

**James Lanagan** and **Paul Ferguson** and **Neil O'Hare** and **Alan F. Smeaton**
Clarity: Centre For Sensor Web Technologies
Dublin City University
Dublin 9, Ireland
{jlanagan, pferguson, nohare, asmeaton}computing.dcu.ie

## Abstract

In this paper we examine the effects of noise when creating a real-world weblog corpus for information retrieval. We focus on the DiffPost (Lee et al. 2008) approach to noise removal from blog pages, examining the difficulties encountered when crawling the blogosphere during the creation of a real-world corpus of blog pages. We introduce and evaluate a number of enhancements to the original DiffPost approach in order to increase the robustness of the algorithm. We then extend DiffPost by looking at the anchor-text to text ratio, and discover that the time-interval between crawls is more important to the successful application of noise-removal algorithms within the blog context, than any additional improvements to the removal algorithm itself.

## 1. Introduction

The changing make-up of the web means that in recent years there has been a shift away from the mass consumption of information generated by a small number of highly-regarded sources. Contributions from web-users in general have also begun to be recognised as useful and important sources of information which are both diverse in nature and collective in their impact (Surowiecki 2004).

It has been shown in the TREC blog track (Ounis et al. 2006) that noise is a major problem when returning relevant search results to a user's query. When we talk about noise, we are referring to the content within a blog's webpage that is not part of the post i.e. advertising, structural content (such as the title banner, archive information, previous posts etc.), or automatically-generated related content based on the post itself.

In this paper we extend the work of (Nam et al. 2009) that builds on the work of (Lee et al. 2008) by applying their Diffpost algorithm to a dynamic corpus of documents. This corpus is created by the continual crawling and retrieval of new blog posts from a large selection of financial websites. The webpages that we crawl contain dynamic content and so the content on a webpage may change between two crawls, along with the noise that surrounds the post. A consequence of this is that the Diffpost approach will fail in its attempts to find repeated content. We present one solution to this in Section 3.




Figure 1: Pagelets Within an Example Webpage.

## 2. Related Work

Much work has been done in the past on separating the content that is interesting or content-bearing from within a webpage (in our case the blog posting itself) from the content which surrounds it. In separating these two distinct types of content within a webpage, (Bar-Yossef and Rajagopalan 2002) refers to 'pagelets' or the pages within a page (Figure 1). Much of the content that is not of direct interest is structural or navigational in style makes up the template for a website, and is likely to be repeated in the same basic form across the site's constituent webpages leading to *intra-page redundancy*. The separation of a website's content and structural information falls within the field of *boilerplate detection*. An excellent overview of the current state-of-the-art is given in (Kohlschütter, Fankhauser, and Nejdl 2010).

Template detection has been an active area of research in recent years. (Bar-Yossef and Rajagopalan 2002) uses a *shingling* technique to account for slight perturbations in template consistency. Both (Vieira et al. 2006) and (Yi, Liu, and Li 2003) use an HTML Document Object Model (DOM) 'style-tree' approach to differentiate between webpage elements, looking for both contextual and style-based uniqueness to attribute importance (or noise) attributes to a specific element. (Kushmerick 2000) advocates the more generic use of wrapper induction to create templates that are site agnostic, creating a 'wrapper' class that may be used to extract the desired content.

## 3. Implementation

In place of a web crawl, we use the Really Simple Syndication (RSS) feeds for the blogs in order to find new content that has been added. So as to capture each new blog posts, we poll a subset of 50 financial blogs of varying article prolificness based on statistics from previous work (O'Hare et al. 2009). In order to find content that is of interest to us on the page, we use the Diffpost algorithm as introduced by (Lee et al. 2008). This algorithm uses the common elements within the pages of a website to filter out repeated content. (Nam et al. 2009) found that a rather simple approach of removing lines that appear in multiple HTML pages of a site can reduce noise significantly. The requirement that lines be repeated across webpages seems too vague and contingent on the correct parsing, download, or processing of the respective webpages. We change this requirement to consider an HTML element's content as a whole creating a more formal and robust method of assessment.

Diffpost suffers from a significant drawback when used in a real-world setting: The requirement for HTML elements to be replicas in order to be removed means dynamic content (advertising based on time, related stories, comments etc.) will not be detected. To combat dynamic content/noise, we introduce the requirement of a minimum number of 'new' postings being downloaded from a source during any one crawl by the system. If the minimum number of new articles have not been posted since the last polling of the source's RSS, we download the balance of articles required from the RSS item list. Whilst the dynamic content/noise may change over time, the change will be consistent across pages that are crawled consecutively.

Though Diffpost is shown to work well in the case of advertising/template noise, it does not do so well in the cases of dynamic *contextual* content. This is content created as a result of the blog post itself; links to related posts or stories, and comments on the post. In the particular use-case of our system, these features were considered noise, though this is not always the case. We address the issue of user comments by varying the frequency of crawls: If there is no time for comments to be made on a post before they are crawled, the problem in theory no longer exists. Lists of related stories, top stories, and other linked content within a blog post's webpage can result in a large decrease in retrieval performance if not removed. These links by their nature contain related keywords and terms that can cause a related but non-relevant post to be returned incorrectly in response to users' queries. We use the fact that these lists are predominantly made up of anchor-text links to other webpages to aid in there removal.

## 4. Experiments

In the experiments that we have performed, our aim was to see firstly how well we can eliminate noise automatically, and secondly to measure what effect the remaining noise has on our retrieval performance. In order to do so, we have created a number of different search indexes on our blog corpus, each based on variations in crawl frequency, Diffpost noise comparison document numbers, and HTML element anchor-text proportion thresholding.

During the collection of our initial corpus of 150 blog feeds and sources for a related project (O'Hare et al. 2009), we have seen that the vast majority of blogs within our initial corpus use one of a select number of blogging template services. The most common of these are Wordpress and Blogger[1], making up 42% and 36% respectively. (16% of blogs either use no generator template, or do not specify.) Once the template is known, we can use the HTML DOM to retrieve just the content of the HTML element containing the post text.

We first took a cross-section of 120 documents from within our corpus that came from 20 distinct sources. These documents were then marked up on a word-token level as noise and non-noise, evaluating both blog post as 'non-noise', and also the comments as 'non-noise'. When only considering the blog-post itself as 'non-noise', only 0.098% of the noise remains in the document, whilst 0.004% of the content is incorrectly removed at a word-token level. Considering comments as 'non-noise' reduces the percentage of noise remaining to 0.080%, though incorrect removals increases marginally to 0.312%. Again however, within our current context, we continue to consider comments to be noise.

We call the 'Template' corpus of documents containing only the text from the filtered blog webpages our ground truth. These documents are then indexed by a Lucene search engine[2], and used for searches. We also create indexes using the Diffpost processed and unprocessed 'Original' documents. Using the approach outlined we are able to measure the effectiveness of the Diffpost algorithm, as well as the improvements that are possible through extensions to it. The blogs used in our experiments are a subset of our initial corpus of 150 blogs comprised of the 34 top sources ranked by prolificness. While it is true that Wordpress and Blogger blogs may be parsed using only a simple parser, using this technique as a comparison standard we are able to look at the effectiveness of our various system configurations.

In order to generate a list of queries to judge the accuracy of our different system configurations, we first queried the 'Template' index with the names of the 500 stocks from the Standard and Poor's (S&P) index[3]. We only consider queries that return a minimum of 20 relevant documents from our 'Template' index. In doing this our list of queries is reduced to 25, but is sufficient for a through investigation of the differences between the system configurations (Harman 1993).

### 4.1 Comparison Documents

The requirement for comparison documents used in the Diffpost algorithm to find repeated content blocks means that it is necessary to ensure at least two documents are downloaded from any source on the same crawl. It can not be

[1] http://www.wordpress.com; http://www.blogger.com

[2] http://lucene.apache.org/

[3] The stocks included in the S&P 500 are those of large publicly held companies that trade on either the NYSE Euronext or the NASDAQ OMX

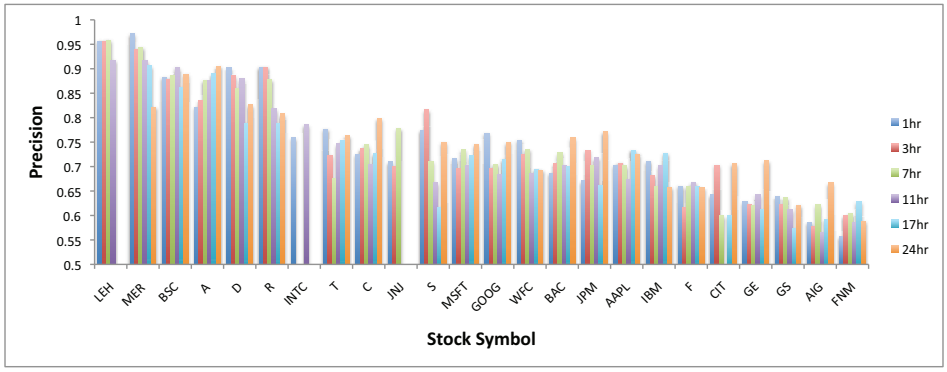


Figure 2: Precision Values for All Stock Queries Based on Different Crawl Frequencies.

assumed however that every source will publish at least two new posts per-crawl interval. We must therefore download older posts again, although these documents are used for comparison purposes only and are not added to the indexes. We know ask how many of these comparison documents are needed/optimal for Diffpost comparisons ?

Table 1: Retrieval Statistics for Different Number of Comparison Documents Using a 60 Minute Crawl Frequency.

| *Docs* | *Diffpost* | | | *Original* | | |
|---|---|---|---|---|---|---|
| | *P* | *R* | *MAP* | *P* | *R* | *MAP* |
| 1 | 0.746 | 0.996 | 0.790 | 0.591 | 0.998 | 0.6675 |
| 3 | 0.747 | **0.982** | 0.775 | 0.591 | 0.998 | 0.6675 |
| 5 | **0.751** | **0.980** | 0.757 | 0.591 | 0.998 | 0.6675 |

Table 1 shows that the performance of the Diffpost algorithm is initially very good in terms of subsequent retrieval, providing a massive boost in precision whilst leaving recall practically untouched. This boost in precision is significant, whilst the drop in recall is not. (As with all significance testing carried out in this paper, we use a two-tail paired t-test, with $\alpha = 0.01$.) With 3 comparison documents, recall drops significantly and using 5 comparison documents results in significant increases and drops respectively for precision and for recall. The significant boost in precision is perhaps not as important as the drop in recall, since this shows that as we add comparison documents, more of the 'non-noise' content is removed. It would appear that using a single comparison document for Diffpost calculation is enough to achieve a significant boost in precision, and a comparable level of recall, to the original documents.

Table 2: Content and Noise Removal Statistics for Different Numbers of Diffpost Comparison Documents

| *Documents* | *Content* | *Noise* | |
|---|---|---|---|
| | *Stripped* | *Remaining*[1] | *Removed* |
| 1 | 0.044 | 0.111 | 0.857 |
| 3 | **0.053** | 0.105 | **0.866** |
| 5 | **0.054** | 0.103 | **0.867** |

[1]As a percentage of the new document.

In terms of noise removal, the use of additional comparison documents does result in a significant increase in noise removed as shown in Table 2, however there is also a correspondingly significant increase in the amount of real content that is stripped. It would seem therefore that the noise that remains in the documents is not as detrimental to retrieval performance as the removal of content that results from the use of additional Diffpost comparison documents.

## 4.2 Polling Frequency

Our second consideration is how often to crawl the source RSS feeds for new content. Comments are always specific to posts and so will not be repeated across them, meaning Diffpost will not eliminate them. It was our assumption therefore that more frequent crawls would reduce the opportunity for comments to be made.

Figure 2 shows the precision for each of the 25 stock queries using each of the different time-based indexes, sorted by decreasing value of average precision. There is no significant difference in the rankings created across the indexes compared with that from the 1 hour frequency, except in the case of the 11 and 17 hour frequency crawls. These two crawls appear to produce significantly better rankings ($p = 0.066$ and $p = 0.059$ respectively) than the others, but as with the 24hour crawl, the number of queries has been reduced. The reason for this is that RSS feeds may be limited in the number of posts contained within the feed, meaning posts from sources that publish often may not remain on the RSS feed for more than a few hours. Once again, we believe this presents a strong additional argument for the use of a 1-hour crawl frequency so as to retain as many posts as possible.

With regards to recall, performance is significantly better using a 1-hour crawl frequency ($p < 0.1$) than all other frequencies except the 3-hour frequency. Again however we can see that one stock (INTC) is not found within the 3-hour index. We therefore feel that with regards to both precision and recall, the optimal crawl frequency must remain that of 1 hour. It is important to be aware of the etiquette of crawling weblogs: We have shown that crawling for new content on an hourly basis is best for preventing comment inclusion etc., but one must also consider that a popular weblog/RSS

may not be appreciative of crawlers that constantly crawl the site.

### 4.3 Anchor-Text Ratios

The last consideration we make when dealing with the possible affects of noisy content is that of anchor- or link-text. Of the factors taken into account during our study, this is perhaps the most context-specific of the three. We noticed that both precision and recall are strongly affected by documents that mention the query within a list of 'related stories/posts'. These lists are mostly made up of anchor-text and are easily detectable as such.

Table 3: Precision and Recall Values for Different Minimum Non-Anchor-Text Ratios Across All Stock Queries

| *Min. Non-Anchor* | *Precision* | *Recall* |
|---|---|---|
| 0(D.P.) | 0.747 | 0.988 |
| 0.1 | 0.745 | 0.980 |
| 0.2 | **0.763** | 0.978 |
| 0.3 | **0.775** | 0.978 |
| 0.4 | **0.781** | 0.978 |
| 0.5 | **0.780** | 0.974 |
| 0.6 | **0.781** | 0.974 |
| 0.7 | **0.808** | **0.930** |
| 0.8 | **0.846** | **0.895** |
| 0.9 | **0.864** | **0.858** |
| 1.0 | **0.840** | **0.690** |

Table 3 shows the precision and recall values as the permissible amount of anchor-text within each HTML entity of a blog posting's webpage is decreased. The first value of 0 is representative of the standard Diffpost algorithm. While there is a highly significant increase ($p < 0.8e^{-3}$) in precision for all values greater than 0.1, the decrease in recall only becomes significant when the anchor-text is greater than 0.7 ($p = 1.3^{-6}$). It would appear from our results that the added pre-Diffpost step of removing all entities of less than 60% non-anchor-text can significantly increase precision whilst leaving recall unaffected.

## 5. Conclusions

Noise in the data that is crawled from a web page can play a large part in the effectiveness of a retrieval system. We have based our work on the Diffpost approach to document template removal, aiming to improve on the previous results. We have shown that through the use of both content analysis and system design it is possible to improve on the results gained by implementing a standard Diffpost algorithm. These results are not as positive as those shown in past work on a static corpus (Nam et al. 2009). They do show that it is possible to use the techniques proven in a laboratory setting within a real-world system, albeit with some modifications.

The problems of dynamically changing temporal content (as well as content that is not available indefinitely) do not exist within the laboratory setting. It is therefore important to highlight these weaknesses within the original approach since they present significant challenges in the real-world scenario. Our work aims to address these issues.

## 6. Acknowledgments

The authors would like to thank the reviewers of this document for their valuable input. This work is supported by Science Foundation Ireland under grant number 07/CE/I1147.

## References

Bar-Yossef, Z., and Rajagopalan, S. 2002. Template Detection via Data Mining and its Applications. In *WWW '02: Proceedings of the 11th International Conference on World Wide Web*, 580–591. Honolulu, Hawaii, USA: ACM.

Harman, D. 1993. Overview of the First TREC Conference. In *SIGIR'93: Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 36–47.

Kohlschütter, C.; Fankhauser, P.; and Nejdl, W. 2010. Boilerplate Detection Using Shallow Text Features. In *WSDM '10: Proceedings of the Third ACM International Conference on Web Search and Data Mining*, 441–450. New York, NY, USA: ACM.

Kushmerick, N. 2000. Wrapper Induction: Efficiency and Expressiveness. *Artificial Intelligence Review* 118(1-2):15–68.

Lee, Y.; Na, S.; Kim, J.; Nam, S.; Jung, H.; and Lee, J. 2008. KLE at TREC 2008 Blog Track: Blog Post and Feed Retrieval. *Proceedings of TREC-08*.

Nam, S.-H.; Na, S.-H.; Lee, Y.; and Lee, J.-H. 2009. DiffPost: Filtering Non-relevant Content Based on Content Difference between Two Consecutive Blog Posts. In *ECIR '09: Proceedings of the 31st European Conference on IR Research on Advances in Information Retrieval*, 791–795. Toulouse, France: Springer-Verlag.

O'Hare, N.; Davy, M.; Bermingham, A.; Ferguson, P.; Sheridan, P.; Gurrin, C.; and Smeaton, A. F. 2009. Topic-Dependent Sentiment Analysis of Financial Blogs. In *TSA '09: Proceeding of the 1st International CIKM Workshop on Topic-Sentiment Analysis for Mass Opinion*, 9–16. Hong Kong, China: ACM.

Ounis, I.; De Rijke, M.; Macdonald, C.; Mishne, G.; and Soboroff, I. 2006. Overview of the TREC-2006 Blog Track. In *Proceedings of TREC*. NIST.

Surowiecki, J. 2004. *The Wisdom of Crowds: Why the Many are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies, and Nations*. Doubleday Books.

Vieira, K.; da Silva, A. S.; Pinto, N.; de Moura, E. S.; Cavalcanti, Jo a. M. B.; and Freire, J. 2006. A Fast and Robust Method for Web Page Template Detection and Removal. In *CIKM '06: Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, 258–267. Arlington, Virginia, USA: ACM.

Yi, L.; Liu, B.; and Li, X. 2003. Eliminating noisy information in web pages for data mining. In *KDD '03: Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 296–305. Washington, D.C.: ACM.