DiffPost: Filtering Non-relevant Content Based on Content Difference between Two Consecutive Blog Posts

Sang-Hyob Nam[1], Seung-Hoon Na[2], Yeha Lee[1], and Jong-Hyeok Lee[1]

[1] POSTECH, South Korea
{namsang,sion,jhlee}@postech.ac.kr
[2] National University of Singapore
nash@comp.nus.edu.sg

Abstract. One of the important issues in blog search engines is to extract the cleaned text from blog post. In practice, this extraction process is confronted with many non-relevant contents in the original blog post, such as menu, banner, site description, etc, causing the ranking be less-effective. The problem is that these non-relevant contents are not encoded in a unified way but encoded in many different ways between blog sites. Thus, the commercial vendor of blog sites should consider tuning works such as making human-driven rules for eliminating these non-relevant contents for all blog sites. However, such tuning is a very inefficient process. Rather than this labor-intensive method, this paper first recognizes that many of these non-relevant contents are not changed between several consequent blog posts, and then proposes a simple and effective DiffPost algorithm to eliminate them based on content difference between two consequent blog posts in the same blog site. Experimental result in TREC blog track is remarkable, showing that the retrieval system using DiffPost makes an important performance improvement of about 10% MAP (Mean Average Precision) increase over that without DiffPost.[1]

1 Introduction

As the number of blogs in the blogosphere increases, information retrieval communities have studied how to find topically-relevant documents from the blogosphere for a given user's query, namely blog retrieval. In this regard, TREC has organized Blog track, a large-scale blog retrieval task, leading participants to find relevant or opinionated posts from about 3,200,000 blog posts starting at TREC-2006 [1].

TREC provides blog collections consisting of original raw web text formats where noise contents such as HTML tags, Javascripts and css are not eliminated. That situation matches the realistic case that commercial blog retrieval systems have trouble with when crawling blogs in the blogosphere. However, it is technically a non-trivial issue to extract cleaned content from a blog post for each blog feed - identifying the relevant content of a blog post.

To see this, Figure 1 presents two consequent posts in a permalink from TREC Blog Track. Non-relevant content such as menu, site description and advertisements in profitable contexts is marked by a box. There non-relevant content is not ignorable where

[1] DiffPost has been applied for creating our baseline and opinion runs to TREC '08 Blog track, and contributed to our top-ranked system [1,2], by further improving the strong baseline based on completely-arbitrary passage-based retrieval and passage-based feedback [3,4].

M. Boughanem et al. (Eds.): ECIR 2009, LNCS 5478, pp. 791–795, 2009.

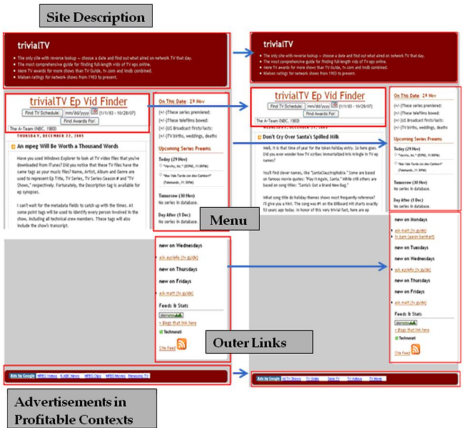

Fig. 1. Two blog posts consequently posted on timestamp in the blog site of TREC blog test collection – BLOG06-20051227-002-0019500473 and BLOG06-20051227-002-0019682198. Major non-relevant contents such as site description, menu, outlier are the same for two posts, except for advertisements.

the size of only the menu is larger than the original content size (we do not describe the full content of the menu). Normally, non-relevant content will still remain when using an open-source text extractor such as HTMLExtractor. These non-relevant contents clearly deteriorate the retrieval system in both effectiveness (the retrieval performance) and efficiency (the indexing size and the retrieval speed). The retrieval performance can be seriously much limited rather than obtainable performance, because of these non-relevant contents. Moreover, the indexing size is also unnecessarily increased, making the retrieval speed low.

To eliminate non-relevant content, a blog crawler can create specialized hand-written rules and regular expressions. However, there are many different types of blog templates which may be provided by commercial blog service vendor to personal developers. Thus, this type of rule-based approach is too labor-intensive to effectively eliminate non-relevant contents.

To this end, this paper proposes DiffPost, a simple and effective algorithm to deal with non-relevant contents. The key idea of DiffPost is based on the observation that many parts of these non-relevant contents are not changed in the same blog site over a long period of time, especially between current and previous blog posts. In Figure 1, we can see that two consecutive blog posts have the same menu, site descriptions and outer links as each other.

Based on this, DiffPost assumed that unchanged content between current and previous blog posts is non-relevant content, and regards only the changed content between them as the relevant content. Despite its simplicity, DiffPost shows a very high accuracy of about 98% F-measure when it is evaluated on randomly sampled TREC blog sites. More importantly, the effect of the retrieval effectiveness using DiffPost is remarkable, showing that MAP is significantly increased by about 10% on the TREC blog subtrack of finding topically-relevant documents. In addition, DiffPost can significantly reduce

the byte size of texts of total documents in the TREC blog test collection from about 14G to only 8.5G. Because DiffPost can be readily adopted and implemented, research on TREC blog track can be more accurately controlled by reducing the noise effect from non-relevant content.

2 DiffPost: Filtering Non-relevant Contents Based on Differences between Two Consecutive Blog Posts

Before applying DiffPost, we first preprocess the downloaded raw blog post where HTML tags, header information, Java script, css, frame, etc. are eliminated. Then, they are segmented into lines using carriage return as the separator. DiffPost tries to compare two sets of lines - one from the current target blog post and the other from the previous reference blog post, target post and reference post, respectively. Set difference between current and previous blogs becomes the relevant content, and the intersection of them becomes the non-relevant content.

To formally describe it, let P be the set of lines in the target post, and P' be lines of sets in the reference post. We assume that there are no redundant lines in each post. The case of redundant lines is trivially extended from the non-redundant version of the algorithm.

DiffPost regards non-relevant content as the intersection of P and P', and relevant content of P as the set difference between P and P'.

$$NonRelContent(P) = P \cap P' \qquad RelContent(P) = P - P' \tag{1}$$

After obtaining $RelContent(P)$, we scan the sequence of lines in the original current blog post, and we concate lines from $RelContent(P)$ and use them as feasible content for the target post, removing ones from $NonRelContent(P)$.

3 Experimentation

3.1 Accuracy of DiffPost

To see how accurately DiffPost finds the non-relevant contents, we randomly selected 10 blog feeds from the TREC blog test collections, and then randomly sampled two consequent blog posts for each blog feed – one for the target post and another for the reference post. When selecting target and reference posts, we only considered a pair which consist of the number of lines between 10 and 200. We annotated all non-relevant lines for each target post, and then applied DiffPost to all the selected target posts. By seeing whether each original non-relevant line belongs to $NonRelContent(P)$, we can calculate precision, recall and F-measure (we do not normalize the redundant lines). We call this *line-level* evaluation. However, the error on a short line may be less-significant than that of a long line. To make a short line less-important than a long line in token-level evaluation. we additionally considered *token-level* evaluation, where the importance of a line is weighted by the number of its tokens.

Table 1 shows the results using line-level and token-level evaluations. Marco metric means the mean of each test post's average per line (or token), and micro metric means the average per line (or token) for all test posts. Note that F-measure is more than 98% and we see that DiffPost shows a very high precision.

Table 1. Performances of DiffPost on different evaluation metrics. F-measure of token-level macro is around 98%.

	Token-Level			Line-Level		
	Prec.	Rec.	F	Prec.	Rec.	F
Macro	0.9775	0.9881	0.9828	0.9906	0.9804	0.9855
Micro	0.9769	0.9920	0.9844	0.9844	0.9878	0.9861

3.2 Improving Retrieval Performance Based on DiffPost

Now we prove how DiffPost is effective on the retrieval performance. To do this, we created two different content sets from a test collection - one is content extracted by applying DiffPost and another is content obtained without using DiffPost. Then, we indexed these two contents separately, and compared the retrieval results. We performed retrieval runs on the standard TREC blog test collection consisting of 3,215,171 permalinks. The evaluation was performed separately on two query sets of BLOG-06 and BLOG-07 which indicate topic set (Q850∼Q900) in TREC 2006, and that (Q901∼950) in TREC 2007, respectively. We used two different fields for a query topic – T (using only title field) and TD (using title and description fields). MAP (Mean Average Precision) and Pr@10 (Precision at top 10 documents) are considered as evaluation measures. We used language modeling approaches using Dirichlet-prior smoothing for the retrieval model [5]. We selected the best smoothing parameter which shows the best performance, and fixed it to all queries.

Table 2 shows MAP and Pr@10 of language modeling approaches using Dirichlet-prior smoothing for two different content indexes. Normal and DiffPost indicate that a normal method and the proposed method are used for extracting blog text contents, respectively. Normal method indicates one using only the preprocessing step of Section 2 before applying DiffPost, removing clearly relevant parts such as HTML tags and java scripts. Remarkably, DiffPost-based contents show a significantly improvement over Normal-based ones in BLOG-07 by increasing about 10% MAP. We obtained an important improvement in BLOG-06 as well. We applied Wilcoxon's sign ranked test to check whether DiffPost-based retrieval show statistically significant improvements over the Normal-based one. Marks † and ‡ at the right of the performance value for DiffPost in Table 2 represent significant results at 95% and 99% confidence value, respectively. As shown in Table 2, DiffPost-based retrieval makes statistically significant improvements

Table 2. MAP (Mean Average Precision) and Pr@10 (Precision of 10 documents) of language modeling approaches using Dirichlet-prior smoothing for two different content indexes – one is obtained by applying DiffPost and the other by the normal method

Metric	Method	BLOG-06		BLOG-07	
		T	TD	T	TD
MAP	Normal	0.3096	0.3532	0.3471	0.3971
	DiffPost	**0.3282**†	**0.3762**†	**0.4410**‡	**0.4915**‡
Pr@10	Normal	0.6400	0.7440	0.6280	0.7220
	DiffPost	**0.6640**	**0.7880**†	**0.6340**	**0.7540**†

in MAP over the Normal-based one for all test collections and topic types. This result indicates that DiffPost is clearly useful in improving the retrieval performance.

In addition, we examined how DiffPost can effectively reduce the size of text contents. Content sizes extracted using Normal and that DiffPost was about 14G (14,688, 648 byte) and 8.5G (8,820,596 byte), respectively. The resulting text size using DiffPost is about 60% of the text size using Normal method. Thus, DiffPost clearly reduces the posting size, thus it will make the retrieval speeds faster, along with a signficantly improved retrieval performance.

4 Conclusion

This paper introduced the problem eliminating non-relevant content when extracting the text of blog posts. To prevent labor-intensive work, this paper focused on the observation that many parts of the non-relevant content does not tend change in the same blog site, and proposed DiffPost, a simple algorithm which detects common parts between a currently-published post and its preceding one and regards them as non-relevant contents. In the web search community, similar ideas to ours had already proposed as "web page template detection" algorithm, including even more sophisticated algorithms [6]. Our work is the first trial in applying such template detection algorithm to blog site, showing its practical usefulness.

Acknowledgement. This work was supported in part by MKE & IITA through IT Leading R&D Support Project and also in part by the BK 21 Project in 2008. We thank the reviewers for their helpful comments.

References

1. Ounis, I., Soboroff, C.M.I.: Overview of the TREC-2008 Blog Track. In: TREC 2008 (2008)
2. Lee, Y., Na, S.H., Kim, J., Nam, S.H., Lee, J.H.: KLE at TREC 2008 Blog Track: Blog Post and Feed Retrieval. In: TREC 2008 (2008)
3. Na, S.H., Kang, I.S., Lee, Y.H., Lee, J.H.: Completely-Arbitrary Passage Retrieval in Language Modeling Approach. In: Li, H., Liu, T., Ma, W.-Y., Sakai, T., Wong, K.-F., Zhou, G. (eds.) AIRS 2008. LNCS, vol. 4993, pp. 22–33. Springer, Heidelberg (2008)
4. Na, S.H., Kang, I.S., Lee, Y.H., Lee, J.H.: Applying Complete-Arbitrary Passage for Pseudo-Relevance Feedback in Language Modeling Approach. In: Li, H., Liu, T., Ma, W.-Y., Sakai, T., Wong, K.-F., Zhou, G. (eds.) AIRS 2008. LNCS, vol. 4993, pp. 626–631. Springer, Heidelberg (2008)
5. Zhai, C., Lafferty, J.: A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In: SIGIR 2001, pp. 334–342 (2001)
6. Chakrabarti, D., Kumar, R., Punera, K.: Page-level template detection via isotonic smoothing. In: WWW 2007, pp. 61–70 (2007)