Blog Post Extraction Using Title Finding

Linhai Song[1, 2], Xueqi Cheng[1], Yan Guo[1], Bo Wu[1, 2], Yu Wang[1, 2]

[1] *Institute of Computing Technology, Chinese Academy of Sciences, Beijing*

[2] *Graduate School of the Chinese Academy of Sciences, Beijing*

Abstract: With the development of Web2.0, web mining applications pay more attention to blog pages. In order to prevent noises in blog pages from affecting the precision of web mining algorithms, it is very necessary to acquire posts from blog pages correctly. In this paper, we propose a blog post extraction algorithm which uses title finding. There are two stages in this algorithm. In the first stage, text nodes which indicate the title of the post are found and used as the beginning of the post. We take a machine learning approach to realize this stage, and employ SVM as classification model. [*]In the second stage, we find the end of the post. Two methods are introduced in this stage, one uses VIPS segmentation results, and the other is based on hand-coded rules. Experiments are conducted to see how we find titles and how we extract posts. Experimental results show that our algorithm has ideal effects.

Keywords: Blog; Post; Title Finding; VIPS; SVM

1. Introduction

In this paper, we address the issue of extracting posts from blog pages.

Posts are content of blog pages and they are very useful information for web mining applications [1] [2]. However, blog pages not only contain posts but also include some other noises, like blogrolls, permalinks, comments, and trackbacks, and these noises may affect the precision of web mining algorithms. In order to improve the performance of web mining systems, it is necessary to acquire posts from blog pages.

Many traditional methods are employed to extract news from news pages. One of the most common heuristic rules used to design these algorithms is text-to-link ratio [3] [4] [5]. But there are two obvious differences between blog pages and news pages: firstly, blog pages may have comments, and news extraction algorithms based on text-to-link ratio may not work well on differentiating useless texts from content, so these methods are not suitable for blog pages; secondly, posts are created by common users, many personalized factors are integrated into the design of blog pages, and this fact causes it difficult to summarize new heuristic rules from content of posts.

Through the observation of blog pages' structures, we find a common feature among blog pages: each blog page has one or two text nodes which indicate the title of the post, and these text nodes appear just before the post of the blog page. Based on this observation, we design a two-stage blog post extraction algorithm:

[*] [†] Corresponding author.

Email addresses: songlinhai@software.ict.ac.cn (L. Song), cxq@ict.ac.cn (X. Cheng), guoy@ict.ac.cn (Y. Guo), wubo@software.ict.ac.cn (B. Wu), wangyu2005@software.ict.ac.cn (Y. Wang).

In the first stage, we find text nodes which indicate the title of the post in each blog page, and use these nodes as the beginning of the post. We take a machine learning approach to realize this stage. We annotate title nodes in sample documents and take them as training data to train a classification model. Then title extraction is performed by using this model. There are in total 88 features defined in the model. As classification model, we employ SVM (Support Vector Machine).

In the second stage, we find the end of the post for each blog page. We propose two methods for this stage. One is based on VIPS (a Vision-based Page Segmentation Algorithm) [6], and the other uses hand-coded rules.

Fig.1 An example of English blog

Fig.2 An example of Chinese blog

Figures 1 and 2 show examples of the problem we encounter. The parts marked Post are targets which we want to extract in our work.

The rest of this paper is organized as follows. Section 2 describes the process of our algorithm. Section 3 and Section 4 describe how we find the title and the end of the post in each blog page. How we design our experiment and experimental results will be presented in Section 5. We then present related work and conclude with a summary and what we will do in the future.

2. Process of algorithm

Figure 2 shows the process of our algorithm. Firstly, each input blog page is parsed and partitioned into blocks. Both our Html Parser and VIPS module are developed using MS IE core. Feature Extractor module picks up features which are used to find title nodes and changes these features into a format which the SVM module can recognize. After finding title nodes, the trained SVM classifier sends results to Post End Extractor module and Post Extractor module. There are two methods in Post End Extractor module to find the end of the post: one is based on segmentation results of VIPS module, and the other uses hand-coded rules. The post of the input page is finally extracted in Post Extractor module according to title nodes and the end of posts.

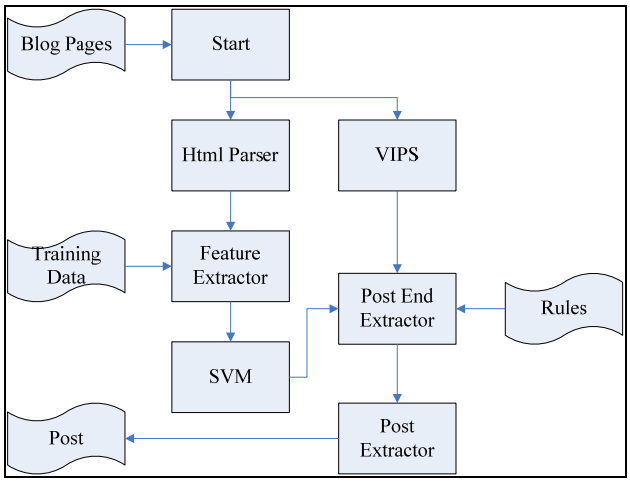

Fig.3 Process of algorithm

3. Title finding

In the first stage of our blog post extraction algorithm, we need to find text nodes which indicate the title of the post.

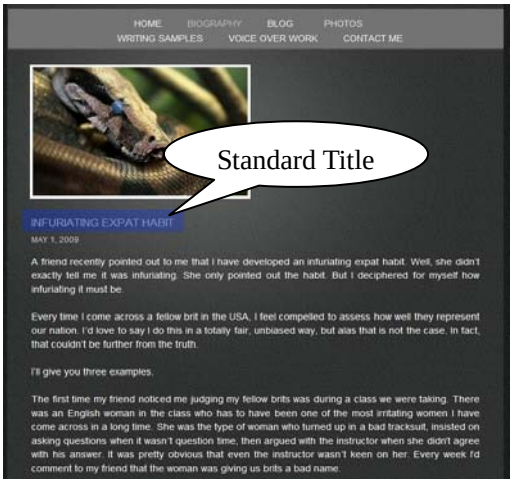

Fig.4 A page only with standardize title

Fig.5 A page with both standard title and personalized title

3.1. Title definition

In blog pages, there are two categories of text nodes which indicate titles of posts. The first category of text nodes is created by blog management systems, and content of these nodes is based on titles which blog owners submit to management systems when they publish their blogs. This kind of text nodes appears in all blog pages, and we call them *standard titles* in our paper. A few noises, like tags for posts, may appear between standard titles and posts.

The second category of text nodes which indicate titles of posts is created by blog owners, and blog owners use this kind of nodes as the first parts of their posts. We call this kind of nodes *personalized*

titles in our paper. Personalized titles do not appear in all blog pages. In our data set, there are only 40.4% blog pages with personalized titles.

If a blog page has a personalized title, we use it as the beginning of the post. But if not, we use the standard title as the beginning.

3.2. Model

Because the label for each text node is “title” or “not title”, we consider title finding stage as a classification problem, and use the nonlinear support vector machine with Gaussian RBF kennel to solve this problem.

In the concrete realization of this stage, we use open-source software libsvm [8]. During the training process, we use cross-validation to prevent the over fitting problem and grid-search to determine parameters.

3.3. Features

We consider various aspects of text nodes to design features for them.

Position features: One common sense is that both standard titles and personalized titles must be in the first screen. So we don’t deal with text nodes not in the first screen and use the size of screen to change absolute position features into relative position features.

Title features: According to web pages design specification, authors should explicitly specify the title fields marked by “<title>” and “</title>”. We have reasons to believe that titles of posts are related to title fields.

Block features: The features of blocks which text nodes belong to are got from VIPS module. We also use the size of screen to change absolute spatial features of blocks into relative spatial features.

Appearance features: Titles may be conspicuous in terms of font family, font weight, font color, font style, alignment and background color.

There are in total 88 features. After being normalized, the value of each feature is a double value between 0 and 1. We could choose to change these double feature values into integers. But we have no experience about how to consider which scale of feature values as equivalent and libsvm has the ability to handle with double values, so we don’t do that. Table 1 shows the number of features for each type.

Table 1 Distribution of feature types

Feature Type	Number
Position Features	4
Title Features	4
Block Features	9
Appearance Features	20
Other	51

4. Post end finding

We have two methods to find the end of the post in each blog pages, one uses VIPS, and the other is based on hand-coded rules. Effects of these two methods are compared in Section 5.

4.1. VIPS

VIPS is a method to segment a web page into blocks. By detecting useful visual cues based on DOM tree, VIPS represent each input page into a block-tree. The root is the whole page, children nodes are obtained by partitioning the parent node into finer blocks, and all leaf nodes consist of a flat segmentation of the web page with an appropriate partitioning degree.

After observing the segmentation results of VIPS, we summarize two heuristic rules: there is only one leaf node in the block-tree carrying posts, and personalized titles and posts must be in the same leaf blocks. Based on these two rules, how we extract the post is shown as follows:

If we can get the personalized title for the input page, we use the leaf block which contains the personalized title as the post. If we only get the standard title, we check leaf blocks in the block tree in DFS order from the block containing the standard title, and use the first block with height higher than a threshold as the post

4.2. Hand-coded rules

We summarize some key words, such as “Comments”, “Post by” and “Next page”. These words often appear after ends of posts. All text nodes after standard titles or personalized titles are checked to see if they contain these key words. If a short text node contains one of these key words, we use it as the end of the post in the input page.

5. Experiment

We create two data sets to test how we find titles and how we extract posts. We collect blog pages from 9 blog sits to establish data set I and we try our best to choose pages produced by different templates. There are 160 blog pages in data set I and 45 pages have personalized titles.

Data set II is from our information system and we want to see how our algorithm supports our actual application. We randomly select 120 blog pages from results of the crawler in our information system. 68 blog pages in data set II have personalized titles.

Table 2 Distribution of feature types

Data sets	Standard title	Personalized title
I	160	45
II	120	68

5.1. Title finding Experiment

We use ‘precision’, ‘recall’ and ‘F1-score’ in evaluation of title finding. The results of the SVM module are classification results of all text nodes, but we only consider the small part about title nodes. Both standard titles and personalized titles are useful for post extraction, and we don’t distinguish them apart in results analysis.

Table 3 Performance of title finding

Data sets	Precision	Recall	F1-score
I	92.23%	90.48%	91.35%
II	96.77%	94.74%	95.74%

From the results shown in Table 3, we see that our classification model has good performance in title finding, and it enables us to continue our post extraction algorithm.

In order to investigate the ability of domain adaptation of our title finding model, we conduct two experiments. In the first experiment, we apply our title finding model trained with data set I to data set II, and then we swap the train and test set to repeat this experiment. In the second experiment, we use our model trained with pages from 5 blog sites in data set I to pages from the other 4 blog sites, and then we swap train and test set to repeat the second experiment.

Table 4. Domain adaptation of title finding model

Training sets	Testing set	Precision	Recall	F1-score
I	II	90.58%	92.02%	91.29%
II	I	79.50%	92.68%	85.59%
5 sites	4 sites	88.89%	75.29%	81.53%
4 sites	5 sites	73.03%	92.50%	81.62%

From the results, we see that cross domain performance is obviously not good as that of within domain in Table 3. Maybe after summarizing more domain independent features, we get construct a domain independent model.

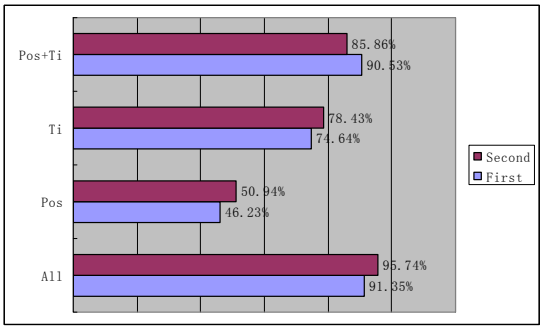

Fig 6 Contribution of each feature type

We investigate the contribution of each type of features in title finding. We find that both Position Features and Title Features are significant features, and other features just slightly improve the performance of our model. The results indicate us that we may just use Position Features and Title Features to establish a simpler model.

5.2. Post Extraction Experiment

We test post extraction on blog pages. We conduct string alignment between experiment results and standard posts to calculate evaluation indexes.

Table 1. Performance of pose extraction

Methods	Testing set	Precision	Recall	F1-score
VIPS	first	94.34%	98.07%	96.17%
VIPS	second	92.40%	98.42%	95.32%
Hand-coded	first	97.41%	90.18%	93.66%
Hand-coded	second	97.67%	92.49%	95.01%

From results in Table 5, we see that the method using VIPS segmentation results has a high recall, but a low precision, and the method based on hand-coded rules just does the opposite. Both of these two methods have ideal effects.

6. Conclusion

In this paper, we propose a two-stage post extraction algorithm. The first stage is title finding, and the second stage is post end extraction. We extract 88 double features for each text node and employ SVM as classification module to realize the first stage. In the second stage, we propose two methods, one uses VIPS, and the other is based on hand-coded rules. Experiment results show that both title finding and post extraction have ideal effects.

Song et al. [7] investigated how to find a model to automatically assign importance values to blocks provided by VIPS. This work inspires us to think whether we could design a content extraction algorithm just using VIPS segmentation results. How to use VIPS more properly to get a powerful content extraction algorithm is the work we will do in the future.

References

[1] N. S. Glance, M. Hurst and T. Tomokiyo, “BlogPulse: Automated Trend Discovery for Weblogs”, In WWW’04: Proceedings of the 13th International Conference on World Wide Web, NY, USA.

[2] W. Liu, S. Tan, H. Xu and L, Wang, “Splog Filtering Based on Writing Consistency”, In WI’2008: 2008 IEEE/ WIC/ ACM International Conference on Web Intelligence, Sydney, Australia, December 9-12, 2008, pp. 227-233.

[3] J. Prasad and A. Paepcke, “Coreex: content extraction from online news articles”, In CIKM’2008: Proceedings of the 17th ACM Conference on Information and Knowledge Management, Napa Valley, California, USA, October 26-30, 2008, pp.1391-1392.

[4] S. Gupta, G. Kailer, D. Neistadt, and P. Grimm, “DOM-based Content Extraction of HTML Documents”, In WWW’03: Proceedings of the 12th International Conference on World Wide Web, Budapest, Hungary, 2003, pp.207-214.

[5] S. Gupta, G. K. Kaiser, P. Grimm, M. F. Chiang, and J. Starren, “Automating Content Extraction of HTML Documents,” World Wide Web, 2005, vol. 8, pp. 179-224.

[6] D. Cai, S. Yu, J. Wen, and W. Ma, “Extracting Content Structure for Web Pages Based on Visual Representation”, In APWeb2003: Web Technologies and Applications, 5th Asian-Pacific Web Conference,

Xian, China, April 23-25, 2002, pp. 406-417.
[7] R. Song, H. Liu, J. Wen and W. Ma, “Learning block importance models for web pages”, In WWW2004: Proceedings of the 13th international conference on World Wide Web, New York, NY, USA, May 17-20, 2004, pp. 203-211.
[8] http://www.csie.ntu.edu.tw/~cjlin/libsvm/