Comment Extraction from Blog Posts and Its Applications to Opinion Mining

Huan-An Kao, Hsin-Hsi Chen
Department of Computer Science and Information Engineering
National Taiwan University, Taipei, Taiwan
E-mail: hhchen@ntu.edu.tw

Abstract

Blog posts containing many personal experiences or perspectives toward specific subjects are useful. Blogs allow readers to interact with bloggers by placing comments on specific blog posts. The comments carry viewpoints of readers toward the targets described in the post, or supportive/non-supportive attitude toward the post. Comment extraction is challenging due to that there does not exist a unique template among all blog service providers. This paper proposes methods to deal with this problem. Firstly, the repetitive patterns and their corresponding blocks are extracted from input posts by pattern identification algorithm. Secondly, three filtering strategies, i.e., tag pattern loop filtering, rule overlap filtering, and longest rule first, are used to remove non-comment blocks. Finally, a comment/non-comment classifier is learned to distinguish comment blocks from non-comment blocks with 14 block-level features and 5 rule-level features. In the experiments, we randomly select 600 blog posts from 12 blog service providers. F-measure, recall, and precision are 0.801, 0.855, and 0.780, respectively, by using all of the three filtering strategies together with some selected features. The application of comment extraction to blog mining is also illustrated. We show how to identify the relevant opinionated objects – say, opinion holders, opinions, and targets, from posts.

1. Introduction

In recent years, blogs have become increasingly popular and have changed the style of communications on the Internet. Blogs allow readers to interact with bloggers by placing comments on specific blog posts. The commenting behavior not only implies the increasing popularity of a blog post, but also represents the interactions between an author and readers.

Due to the growing amount of blogs, many works such as blog search, summarization, opinion mining, *etc*, have been investigated. Cao et al. (2008) showed that consideration of both post content and comment region achieves better retrieval performance in blog search. Hu et al. (2007) extracted sentences from post content and regarded them as summary of the blog post. Liu et al. (2007) mentioned that bloggers express their opinions on a particular subject through writing blog posts.

Identifying the boundary between post content and comment region, and extracting the comments in a region are fundamental for blog applications. Moreover, mining opinions in a blog post, author's opinions are not enough. It is necessary to consider both author's and readers' opinions toward the same topic.

To extract comments from blog posts is challenging. Each blog service provider has its own templates to present the information in comments. These templates do not have a general specification about what components must be provided in a comment or how many complete sub-blocks a comment is composed of.

This paper studies how to extract comments in blog posts and illustrates how to identify both author's and readers' opinions. Section 2 describes the system flow including the repetitive pattern identification, filtering strategies, and binary classification. Section 3 shows the experimental setup and evaluation. Section 4 applies the results of comment extraction to opinion mining.

2. Comment Extraction

2.1 System Flow

Given a blog post P, the task of comment extraction is to extract a set of comments $C = \{c_1, c_2, \ldots, c_n\}$ associated with P. A "site-level" approach gathers information from a designated blog service provider, parses the HTML contents, and identifies comment extraction rules manually. This approach suffers from human cost to formulate the rules and fail when a new blog site is first encountered. A "page-level" approach reads blog pages from different blog sites, and identifies the repetitive patterns embedded in the pages.

Figure 1 shows the architecture of our page-level approach. It includes an encoder which accepts an input post page, a repetitive pattern identifier which recognizes the repetitive patterns and the set of blocks, three filtering strategies which remove blocks with loop or overlap, and a comment/non-comment classifier which distinguishes comment blocks and non-comment blocks.

2.2 Repetitive Pattern Identification

HTML documents are composed of various kinds of tags carrying structure and presentation information, and text contents enwrapped by tags. Because our goal is to mine general comment structures, the information irrelevant to the document structures is not considered.

The input to pattern identification is an encoded string from an encoder. Each token in the string represents an HTML tag or a non-tag text. The algorithm scans the tokens. When encountering a token that is likely to be the head of a repetitive pattern (called a "rule" hereafter too), the subsequent tokens are examined if any rules can be formed.

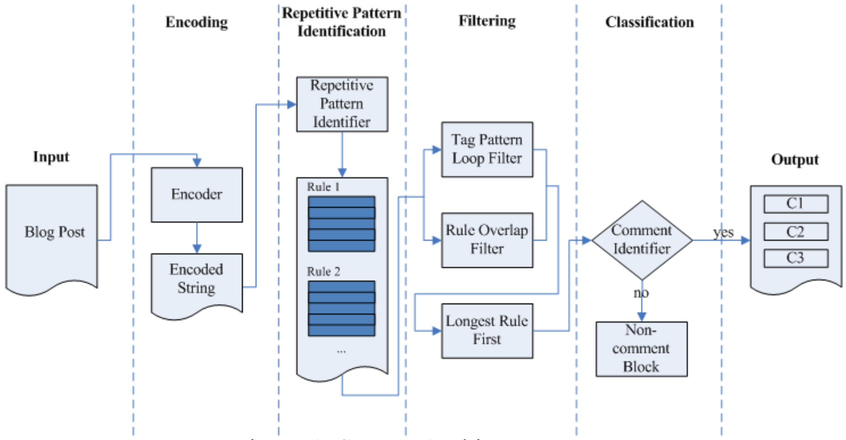

Figure 1: System Architecture

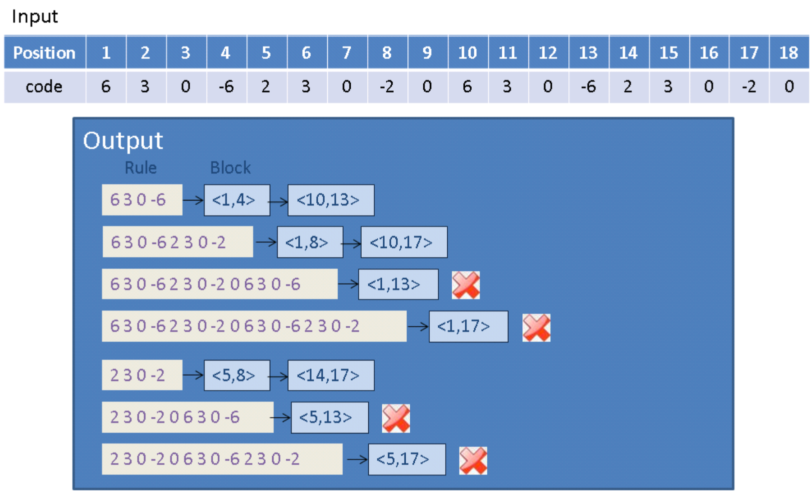

Figure 2: An Example of Repetitive Pattern Identification

HTML Tag	Code String	Class
Non-tag Text	0	string
STRUCTURE	1	begin
DIV CLASS	2 3	begin with attribute
DIV ID	2 4	begin with attribute
DT CLASS	5 3	begin with attribute
LI CLASS	6 3	begin with attribute
DD CLASS	7 3	begin with attribute
TR CLASS	8 3	begin with attribute
/STRUCTURE	-1	end
/DIV	-2	end
/DT	-5	end
/LI	-6	end
/DD	-7	end
/TR	-8	end
Other Tags	0	string

Table 1: Coding Scheme for Repetitive Pattern Identification

Figure 2 shows an encoded string '6 3 0 -6 2 3 0 -2 0 6 3 0 -6 2 3 0 -2 0' corresponding to an HTML document denoting a blog post. Table 1 lists all the tags and the corresponding codes used by the encoder. We mine seven rules and corresponding blocks from this string. We remove those rules (3^{rd}, 4^{th}, 6^{th}, and 7^{th}) with only one block, and keep the remaining repetitive patterns (1^{st}, 2^{nd} and 5^{th}). Finally, 6 candidates are proposed and sent to the next stage.

2.3 Filtering Strategies

Since not all mined repetitive patterns are correct, non-comment blocks may be proposed wrongly. We present three filtering strategies, i.e., tag pattern loop filtering (M1), rule overlap filtering (M2) and the longest rule first (M3), to eliminate non-comment blocks. M1 and M2 are independent of each other, and M3 must be performed after M1 and M2. Figures 3-5 list an example

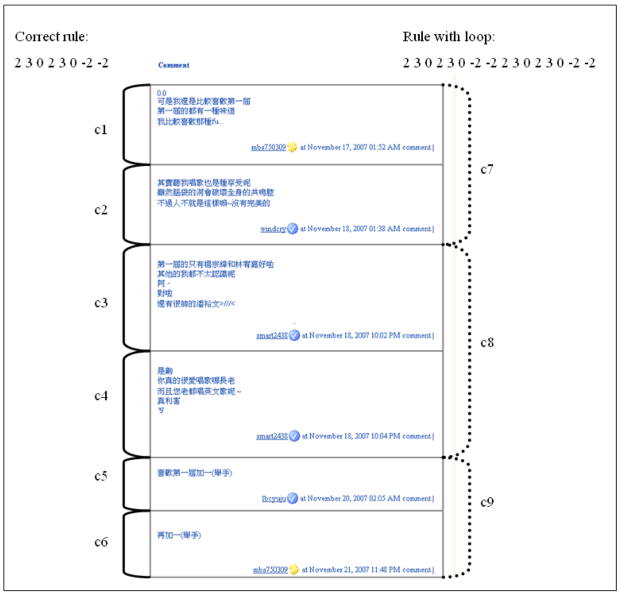

Figure 3: Correct Rule vs. Rule with Loop

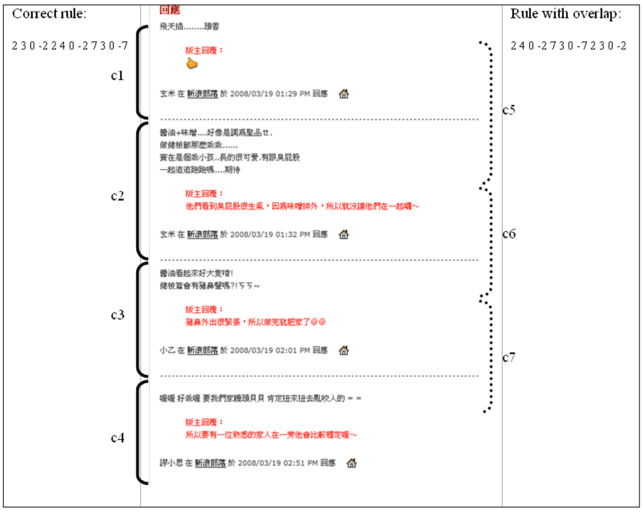

Figure 4: Correct Rule vs. Rule with Overlap

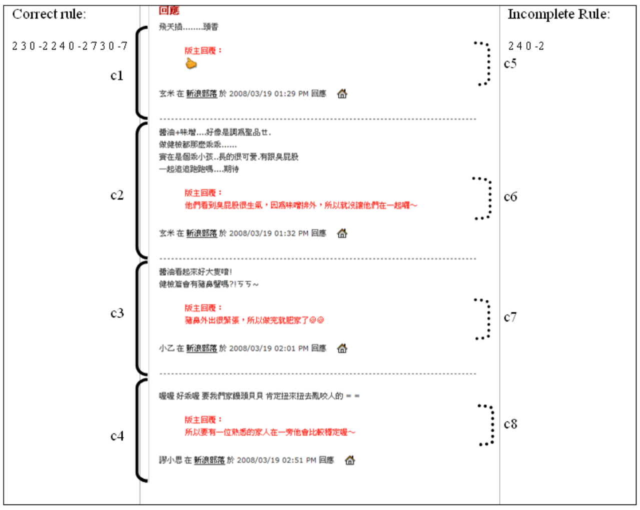

Figure 5: The Longest Rule First

for each case. The left braces list the correct boundaries of comments in the post shown in the middle. Some incorrect blocks shown on the right braces are too large (Figure 3), overlap (Figure 4) or too small (Figure 5).

2.4 Binary Classification

A comment/non-comment classifier is based on features selected from both block-level and rule-level listed below.

Block-Level Features

(1) **Block length without tags.** Comments are shorter than post contents in blog posts on average.
(2) **Block length with tags.** HTML tags are considered in the determination of block length.
(3) **Number of words.** We consider block length in words instead of characters.
(4) **Frequency of "comment" word.** The word "comment" often appears in comment blocks.
(5) **Ratio of anchor tags.** Anchor tag contains a hyperlink to a page. This feature measures the anchor tag ratio in a block as (number of anchor tags) / (number of tags).
(6) **Number of anchor tags.** This feature measures the number of anchor tags instead of ratio in a block.
(7) **Ratio of stop words.** We postulate that stop word ratio may be lower in blogroll, categories, advertisements and other possible templates. In contrast, stop word ratio tends to be higher in comment blocks.
(8) **Number of stop words.** This feature measures the number of stop words instead of ratio in a block.
(9) **Ratio of punctuation marks.** We postulate that the punctuation ratio is higher in comment blocks than that in other blocks.
(10) **Number of punctuation marks.** This feature measures the number of punctuation marks instead of ratio in a block.
(11) **Block start position.** Comment region always appears after the post content. It seldom occurs in the top of the page.
(12) **Block end position.** This feature is defined as the end position of a block divided by the length of whole post.
(13) **Number of date and time expressions.** Reader responses always accompany with time and date expressions. This feature counts the occurrences of date and time expressions in a block.
(14) **Occurrence of date and time expressions.** This feature equals to 1 if date and time expressions occur in a block, 0 otherwise.

Rule-Level Features

(15) **Rule start position.** This feature captures the starting position of a rule used to divide a blog post into a post region and a comment region.
(16) **Rule end position.** Rule end position models the end position of a comment region.
(17) **Density.** Density measures the ratio of total length of comment blocks divided by the length of a rule.
(18) **Coverage.** The length of a region compared to the whole blog post may provide information about whether it is a comment region.
(19) **Regularity.** The space between each adjacent comment block is almost the same.

Support Vector Machine (SVM) is adopted to learn a comment/non-comment classifier with the selected features.

3. Experiments

3.1 Experimental Setup

Total 12 blog service providers listed in Table 2 are used to collect a corpus *CommentExtract* 1.0. Total 50 blog posts are selected from each service provider. We manually labeled each comment in a given blog post by considering two comment styles: comment blocks without or with author reply. For the former, besides comment content, some related information such as commenter name, date and time are included. For the latter, we regard a comment block as a composite of both a reader comment and an author reply. A labeled comment block must include comment contents of readers and author. Table 3 lists the statistics of the *CommentExtract* 1.0 corpus.

3.2 Evaluation

A 12-fold cross validation is conducted. Each fold comes from a blog site. The data from 11 blog sites are used for training, and the remaining site is for testing. Each fold contains 50 posts from the same site. Table 4 compares different combinations of the three filtering strategies. Employing all strategies together achieves the best.

Provider	URL
Wretch	http://www.wretch.cc/blog
Yam	http://blog.yam.com
Pixnet	http://www.pixnet.net/blg
Roodo	http://blog.roodo.com
Blogspot	http://www.blogger.com/home
Xuite	http://blog.xuite.net
Sina	http://blog.sina.com.tw
Yahoo	http://tw.blog.yahoo.com
China Times	http://blog.chinatimes.com
Udn	http://blog.udn.com
MSN	http://home.services.spaces.live.com
Oui	http://www.oui-blog.com

Table 2: Blog Service Providers

Number of blog posts	600
Number of blog posts with comments	482
Number of comments	3,505
Mean # comments per blog post	5.8
Comment Length (with tag)	
Mean	906.1
Maximum	7,593
Minimum	140
Median	756
Comment Length (without tag)	
Mean	290.4
Maximum	6,905
Minimum	41
Median	206

Table 3: Statistics of *CommentExtract* 1.0 Corpus

Strategy	Recall	Precision	F-measure
No Filter	0.607	0.166	0.221
M1	0.652	0.453	0.493
M2	0.663	0.206	0.256
M3	0.695	0.347	0.387
M1+M2	0.646	0.520	0.526
M2+M3	0.694	0.416	0.452
M1+M3	0.660	0.687	0.640
M1+M2+M3	**0.717**	**0.793**	**0.715**

Table 4: Comparisons of Different Filtering Strategies

Recall that we propose 14 block-level features and 5 rule-level features to discriminate comment blocks from non-comment blocks. To examine which features are critical, we remove a feature from the feature set one at a time, repeat the same training and testing procedure, and tell out the performance differences. In total, there are 19 experiments on 12-fold cross validation. Table 5 shows F-measure of classifiers after a feature being removed. Except that features 5 and 15 do not result in clear performance difference, removing features 4, 13, 14, 16 or 17 lower the average performance, and removing features 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 18, or 19 increases the average performance. The former features may be important for improving the performance because the performance decreases when these features are removed. When only they are used, the F-measure is improved from 0.715 to 0.781 compared to using all features.

We also employ feature scores for feature selection (Chen and Lin, 2005). Feature score measures the discrimination of two sets of real numbers. For each feature, its values of positive and negative instances in training data can be used to assess if this feature is discriminative. Given a training vector x_i, its elements are all values extracted by the i-th feature. Now we have 19 features, so that i can be 1 to 19. Feature score for i-th feature $F(i)$ is defined as follows.

$$F(i) = \frac{\left(\bar{x}_i^{p} - \bar{x}_i\right) + \left(\bar{x}_i^{n} - \bar{x}_i\right)}{\frac{1}{n_p - 1}\sum_{k=1}^{n_p}\left(x_{k,i}^{p} - \bar{x}_i^{p}\right)^2 + \frac{1}{n_n - 1}\sum_{k=1}^{n_n}\left(x_{k,i}^{n} - \bar{x}_i^{n}\right)^2}$$

where n_p and n_n denote the number of positive and negative examples, respectively; $\bar{x}_i$, $\bar{x}_i^p$, $\bar{x}_i^n$ is the average of the i-th feature of the whole, positive, and negative data sets, respectively; $x_{k,i}^p$ is the i-th feature of the k-th positive example, and $x_{k,i}^n$ is the i-th feature of the k-th negative example.

Table 6 lists the feature score of each feature and the corresponding rank. The top-3 discriminative features are *occurrence of date and time expressions*, *number of date and time expressions*, and *density*. They also belong to the positive feature set selected by the approach of removing one feature at a time. The next top two features are *ratio of stop words* and *frequency of "comment" word*. Table 7 presents the performance of overall system and comment/non-comment classifier in the same 12-fold cross validation with different feature sets. When features 14, 13, 17, 7 and 4 (i.e., *occurrence of date and time expressions*, *number of date and time expressions*, *density*, *ratio of stop words*, and *frequency of "comment" word*) are adopted, F-measure is improved further to 0.801.

feature	Roodo	Wretch	Yahoo	Xuite	China	Yam	Pixnet	Sina	Oui	Blogspot	Udn	MSN	Average
All	0.790	0.639	0.751	0.459	0.965	0.783	0.581	0.900	0.662	0.841	0.338	0.871	0.715
Removed Feature													
1	0.866	0.956	0.802	0.533	**0.626**	0.811	0.591	**0.865**	0.742	**0.797**	0.363	**0.800**	0.729
2	0.806	0.724	0.774	**0.448**	**0.956**	0.795	**0.573**	**0.885**	0.717	**0.804**	0.401	**0.848**	0.728
3	0.810	0.746	0.763	**0.444**	**0.956**	0.796	**0.572**	**0.893**	0.726	**0.819**	0.342	**0.846**	0.726
4	**0.779**	0.798	0.792	0.604	**0.579**	0.804	**0.553**	**0.825**	0.740	**0.598**	**0.317**	**0.741**	**0.677**
5	0.856	0.909	0.789	0.492	**0.621**	0.811	0.585	**0.871**	0.746	**0.765**	0.400	**0.762**	0.717
6	0.808	0.787	0.763	**0.446**	**0.959**	0.798	**0.573**	0.899	0.726	**0.817**	0.338	**0.841**	0.729
7	**0.639**	0.972	0.757	0.458	**0.924**	0.801	0.587	**0.879**	0.733	**0.736**	0.471	0.901	0.738
8	0.807	0.746	0.763	**0.444**	**0.959**	0.798	**0.572**	0.899	0.724	**0.820**	**0.331**	**0.846**	0.726
9	**0.785**	0.746	0.762	**0.446**	**0.953**	0.796	**0.572**	0.899	0.721	**0.818**	0.357	**0.839**	0.724
10	0.810	0.787	0.763	**0.446**	**0.959**	0.798	**0.572**	**0.893**	0.724	**0.821**	0.339	**0.841**	0.729
11	0.804	0.853	0.769	**0.447**	**0.959**	0.796	**0.573**	0.896	0.725	**0.817**	0.340	**0.848**	0.735
12	0.813	0.825	0.769	**0.448**	**0.959**	0.798	**0.572**	0.898	0.724	**0.809**	0.345	**0.848**	0.734
13	**0.413**	**0.356**	**0.700**	**0.208**	**0.458**	**0.440**	**0.574**	**0.435**	**0.596**	**0.576**	**0.104**	**0.316**	**0.431**
14	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
15	**0.762**	0.816	0.770	**0.430**	**0.953**	**0.774**	**0.570**	**0.886**	0.716	**0.788**	**0.316**	**0.808**	0.716
16	**0.722**	**0.337**	**0.716**	**0.436**	0.969	0.792	**0.558**	**0.891**	0.726	**0.646**	**0.331**	**0.813**	**0.661**
17	**0.618**	**0.318**	**0.712**	**0.413**	**0.890**	**0.706**	0.628	**0.723**	0.663	**0.769**	**0.320**	**0.685**	**0.620**
18	**0.712**	0.862	0.758	**0.431**	**0.960**	0.783	**0.563**	0.920	0.736	**0.776**	**0.328**	**0.846**	0.723
19	**0.557**	0.904	0.770	0.547	**0.946**	0.790	**0.561**	0.953	0.741	**0.813**	**0.305**	**0.825**	0.726

Table 5: Comparisons of Different Features Using Removing One Feature at a Time

Rank	Id	Feature	Feature Score
1	14	Occurrence of date and time expressions	2.908
2	13	Number of date and time expressions	0.630
3	17	Density	0.287
4	7	Ratio of stop words	0.250
5	4	Frequency of "comment" word	0.190
6	15	Rule start position	0.113
7	8	Number of stop words	0.102
8	3	Number of words	0.092
9	10	Number of punctuation marks	0.089
10	19	Regularity	0.087
11	1	Block length without tags	0.083
12	18	Coverage	0.076
13	5	Ratio of anchor tags	0.052
14	9	Ratio of punctuation marks	0.027
15	12	Block end position	0.022
16	11	Block start position	0.019
17	2	Block length with tags	0.014
18	6	Number of anchor tags	0.011
19	16	Rule end position	0.003

Table 6: Feature Scores and Rank of Each Feature

Number of Features	Overall			Classifier		
	Recall	Precision	F-measure	Recall	Precision	F-measure
19	0.668	0.793	0.682	0.717	0.793	0.715
18	0.597	0.793	0.627	0.636	0.793	0.656
17	0.604	0.793	0.632	0.643	0.793	0.661
16	0.622	0.795	0.644	0.662	0.795	0.674
15	0.640	0.797	0.661	0.683	0.797	0.692
14	0.644	0.797	0.663	0.687	0.797	0.693
13	0.661	0.799	0.677	0.706	0.799	0.709
12	0.650	0.795	0.669	0.693	0.795	0.699
11	0.686	0.798	0.703	0.736	0.798	0.737
10	0.696	**0.800**	0.712	0.748	**0.800**	0.746
9	0.628	0.784	0.629	0.671	0.784	0.657
8	0.639	0.785	0.640	0.682	0.785	0.668
7	0.648	0.786	0.651	0.693	0.786	0.681
6	0.656	0.786	0.656	0.701	0.786	0.686
5	**0.793**	0.780	**0.766**	**0.855**	0.780	**0.801**
4	0.774	0.737	0.730	0.836	0.737	0.763
3	0.782	0.758	0.743	0.845	0.758	0.776
2	0	0	0	0	0	0
1	0	0	0	0	0	0

Table 7: Comparison of Different Number of Features Using Feature Scores

4. Application on Opinion Mining

After comment extraction, the opinions in each comment and the amount of comments which indicates the polarity tendency in each post can be presented to users. Typical opinionated information contains three basic components: an opinion holder, an opinion, and a target. The author of a blog post is the opinion holder of the post content and the reader who writes a comment is the opinion holder of this comment. The opinions are actually viewpoints or attitudes expressed in post content and each comment. A target can be a product, an event, a person, an organization, *etc*. It is usually specified in blog post.

We adopted opinion mining algorithms proposed by Ku and Chen (2007) to determine the opinion tendency of post content and the accompanying comments. They are categorized into positive, negative, or neutral for further applications. Figure 6 shows a user interface of blog search. The search results are categorized into positive, negative, and neutral, and the numbers of positive, negative, and neutral blog posts are also presented. For a blog post, the result shows its link, title, and snippets. Besides, the numbers of positive, negative, and neutral comments in a blog post are also summarized. Figure 7 shows a blog post with opinion information. The left side of this figure lists the original blog post and its right side the opinions of the post content and each comment. We can easily tell out the opinions of both the author and the readers by the up and down symbols, i.e., supportive and not supportive.

5. Conclusion

This paper presents a prototyped system to identify comments from blog posts and applies them to opinion mining. The best F-measure, recall, and precision of comment extraction are 0.801, 0.855, and 0.780, respectively, by using all of the three filtering strategies together with some selected features. After comment extraction, each comment is also categorized into positive, negative, or neutral comment by the same opinion mining algorithm as the one used in post content.

Many kinds of irrelevant comments are posted. For example, spam comments may carry advertisements with few links. Besides, commenter may just leave a message for greeting. Identifying relevant comments is an important and challenging issue for correctly fining the opinion of readers.

6. Acknowledgments

Research of this paper was partially supported by National Science Council, Taiwan, under the contract 96-2628-E-002-240-MY3.

7. References

Cao, D.L., Liao, X.W., Xu, H.B. and Bai, S. (2008). Blog Post and Comment Extraction Using Information Quantity of Web Format. In *Proceedings of 2008 AIRS*. LNCS 4993, pp. 298--309.

Hu, G., Sun, A., and Lim, E.P. (2007). Comment-Oriented Blog Summarization by Sentence Extraction. In *Proceedings of 6th ACM CIKM*. pp. 901--904.

Liu, Y., Huang, X., An, A., and Yu, X. (2007). ARSA: A Sentence-Aware Model for Predicting Sales Performance Using Blogs. In *Proceedings of SIGIR 2007*. pp. 607--614.

Chen, Y.W. and Lin, C.J. (2005). Combining SVMs with Various Feature Selection Strategies. http://www.csie.ntu.edu.tw/~cjlin/papers/features.pdf

Ku, L.W. and Chen, H.H. (2007). Mining Opinions from the Web: Beyond Relevance Retrieval. *JASIST*. Vol. 58, No. 12, pp. 1838--1850.

Figure 6: User Interface to Blog Opinion Search

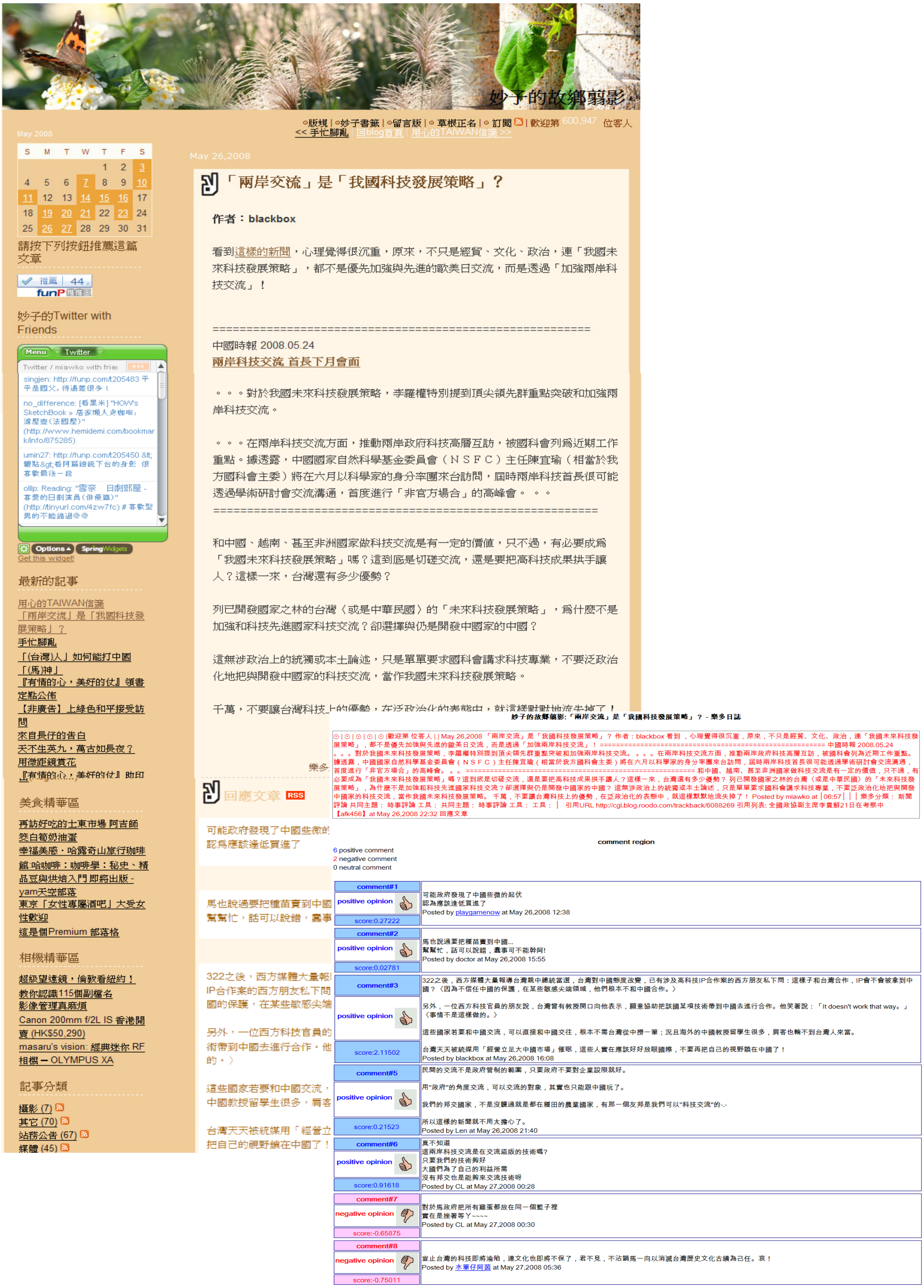

Figure 7: A Blog Post with Opinion Information