Text Content Filtering Based on Chinese Character Reconstruction from Radicals

Wenlei He[1], Gongshen Liu[1], Jun Luo[2], and Jiuchuan Lin[2]

[1] School of Information Security Engineering
Shanghai Jiao Tong University
[2] Key Lab of Information Network Security of Ministry of Public Security
The Third Research Institute of Ministry of Public Security

Abstract. Content filtering through keyword matching is widely adopted in network censoring, and proven to be successful. However, a technique to bypass this kind of censorship by decomposing Chinese characters appears recently. Chinese characters are combinations of radicals, and splitting characters into radicals pose a big obstacle to keyword filtering. To tackle this challenge, we proposed the first filtering technology based on combination of Chinese character radicals. We use a modified Rabin-Karp algorithm to reconstruct characters from radicals according to Chinese character structure library. Then we use another modified Rabin-Karp algorithm to filter keywords among massive text content. Experiment shows that our approach can identify most of the keywords in the form of combination of radicals and yields a visible improvement in the filtering result compared to traditional keyword filtering.

Keywords: Chinese character radical, multi-pattern matching, text filtering.

1 Introduction

In the past decades, Internet has evolved from an emerging technology to a ubiquitous service. The Internet can fulfill people's need for knowledge in today's information society by its quick spread of all kinds of information. However, due to its virtuality and arbitrariness nature, Internet conveys fruitful information as well as harmful information. The uncontrolled spread of harmful information may have bad influence on social stability. Thus, it's important to effectively manage the information resources of web media, which is also a big technical challenge due to the massive amount of information on the web.

Various kinds of information are available on the web, text, image, video, etc. Text is the dominant among all of them. Netizens are accustomed to negotiating through e-mails, participating in discussion on forums or BBS, recording seeing or feeling on blogs. Since everyone can participate in those activities and create shared text content on the web, it's quite easy for evils to create and share harmful texts. To keep a healthy network environment, it's essential to censor and filter text content on the web so as to keep netizens away from the infestation of harmful information.

The most prominent feature of harmful information is that they are always closely related to several keywords. Thus, keyword filtering is widely adopted to filter text content [1], and proven to be quite successful. While the priest climbs a post, the devil climbs ten, keyword filtering are not always effective. Since Chinese characters are combinations of character radicals [2], many characters can be decomposed into radicals, some characters are themselves radicals. This made it possible to bypass keyword filtering without affecting understanding the meaning of those keywords by replacing one or more characters in keyword with combination of character radicals. E.g. use "运云力" to represent "运动".

Traditionally, we can filter harmful document related to "法轮功" by matching keyword, but some evil sites replaced "法轮功" with "三去车仑工力", causing the current filtering mechanism to fail. Even worse, since the filtering mechanism has failed, people can search for harmful keywords like "三去车仑工力" in commodity search engines, and get plenty of harmful documents from search result. Many evil sites are now aware of this weakness of the current filtering mechanism, and the trick mentioned above to bypass keyword filtering is becoming more and more popular. We analyzed a sample of harmful documents collected by National Engineering Laboratory of Content Analysis. Our analysis shows that:

- A visible portion of harmful documents has adopted the decomposing trick to bypass the filtering mechanism, see Table 1.
- Most of the documents involving decomposed characters contain harmful information.

Table 1. Statistic of sampled harmful documents

Category	Proportion	Sample Size
Reactionary	9%	893
Adult	8%	2781
Political Criticism	10%	1470
Public Hazard	6%	1322

The second column in the table shows the proportion of harmful documents containing intentionally decomposed Chinese characters in a category (number of harmful documents containing decomposed characters / number of harmful documents).

Decomposing Chinese characters into radicals is a new phenomenon on the web. The idea behind this trick is simple, but it can completely fail the traditional keyword filtering. Filtering against this trick is a new research topic without much attention now. In this paper, we proposed the first filtering technology against those intentionally decomposed characters. We first set up a Chinese character decomposing structure library. Section 2 gives an overview on the principles of how to decompose Chinese characters. Section 3 gives an overview of our filtering system. We used a modified Rabin-Karp [3] multi-pattern matching algorithm to reconstruct characters from radicals before applying keyword filtering. After reconstruction, we used another modified Rabin-Karp algorithm to filter keywords. We described our modification to Rabin-Karp in Section 3.1, 3.2. In Section 4, we compared our filtering results with traditional filtering, and also showed the efficiency improvement of our modified Rabin-Karp algorithm in reconstruction. We gave a conclusion of our work in Section 5.

2 Principles for Chinese Character Decomposing

Chinese character is structured two-dimension character. Every Chinese character is composed of several character radicals. Chinese Linguistics and Language Administration gave an official definition for character radical in <GB13000.1 Chinese Character Specification for Information Processing>: composing unit of Chinese characters that is made up of strokes [4]. Character radical has a hierarchy structure. A character radical can be made up of several smaller character radicals. E.g. Chinese character "想" is composed of "相" and "心", these two are level 1 radicals for "想". "相" is made up of "木" and "目", and these two are level 2 radicals for "想". Since level 1 decomposing is more intuitive than level 2 decomposing, e.g." 木目" looks like "相", but it's hard for people to think of "想" when looking at "木目心", in order to make words understandable, usually only level 1 decomposing is used in bypass filtering. We see no level 2 decomposing in the harmful document collection from National Engineering Laboratory of Content Analysis. Accordingly, we consider only level 1 decomposing.

The structure of a Chinese character usually falls into the following categories: left-right, up-down, left-center-right, up-center-down, surrounded, half-surrounded, monolith. Intuitively, left-right, left-center-right structure characters are more understandable after decomposing. Statistics [5] shows that left-right structure counts for over 60 percent of all Chinese characters; up-down structure counts for over 20 percent. We summarize these observations as the following conclusions:

- Level 1 decomposing is more intuitive
- Left-right, left-center-right decomposing is more intuitive

We manually decomposed some Chinese characters defined in GB2312 charset which are easily understandable after decomposing. Based on the above conclusions, most of the characters we choose to decompose are left-right characters, and we use only level 1 decomposing. The outcome of our decomposing work is a Chinese character decomposing structure library (character structure library for short) in the form of character-structure-radical triplets, as shown in Table 2.

Table 2. Sample of Chinese character decomposing structure library

Character	Structure	Radicals
权	Left-right	木又
格	Left-right	木各
就	Left-right	京尤
动	Left-right	云力
树	Left-center-right	木又寸
起	Half-surrounded	走己

Some radicals are variants of characters, some are not. Take "河" for example, if we decompose it into "水" and "可", it would be confusing and not understandable. Instead, we choose to decompose it into "三" and "可", which is more meaningful.

3 Keyword Filtering Based on Chinese Character Reconstruction

Figure 1 gives an overview on our filtering system. HTML files shown in Figure 1 are collected via collectors in network. Preprocessing will remove all HTML tags, punctuations, and white-spaces. If punctuations and white-spaces are not removed, those punctuations in between characters of a keyword may cause keyword matching to fail. Next, we take the decomposed characters in character structure library as patterns, and use multi-pattern matching algorithm to find out and recombine all intentionally decomposed characters. After character reconstruction, we use another multi-pattern matching algorithm to search for keywords, and filter out all documents that contain any keywords.

In the above process, we used two multi-pattern matching algorithms, and the efficiency of the two algorithms is vital to the performance of the whole filtering system. We carefully selected the two algorithms. We modified Rabin-Karp [3] algorithm to better fit our scenario of character reconstruction and keyword filtering. We describe our modification to Rabin-Karp algorithm in Section 3.1 and 3.2.

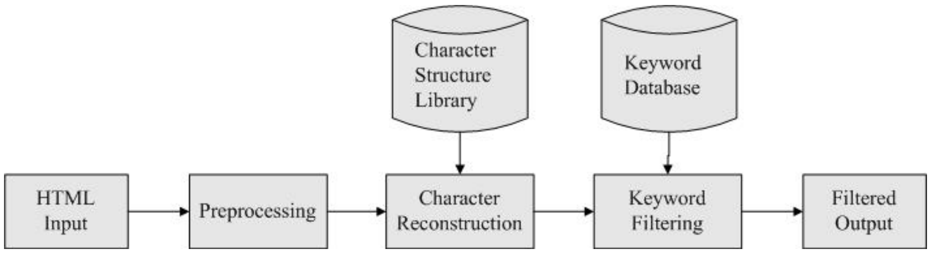

Fig. 1. Overview of filtering system

3.1 Chinese Character Reconstruction

Recombining Chinese character from character radicals is a multi-pattern matching problem in nature. Pattern matching [6] can be divided into single pattern matching and multi-pattern matching. Let P = {p_1, p_2,...,p_k} be a set of patterns, which are strings of characters from a fixed alphabet ∑. Let T=t_1, t_2,...,t_N be a large text, again consisting of characters from ∑. Multi-pattern matching is to find all occurrences of all the patterns of P in T. Single pattern matching is to find all occurrences of one pattern p_i in T. KMP (Knuth-Morris-Pratt) [7] and BM (Boyer-Moore) [8] are classical algorithms for single pattern matching. AC (Ano-Corasick) [9] and Wu-Manber [10] are algorithms for multi-pattern matching. AC is a state machine based algorithm, which requires large amount of memory resources. WM is an extension of BM, and it has the best performance in average case.

[11] proposed an improved WM algorithm. The algorithm eliminates the functional overlap of the table HASH and SHIFT, and computes the shift distances in an aggressive manner. After each test, the algorithm examines the character next to the scan window to maximize the shift distance. The idea behind this improvement is consistent with that of the quick-search (QS) algorithm [12].

From the observations in Section 2, we know that most patterns in character structure library are of length 2, few are of length 3. Since the prefix and suffix of WM algorithm overlaps a lot for patterns of length 2 and 3, it's not efficient to use WM. On the other hand, WM algorithm for such short patterns will act similar to Rabin-Karp algorithm, except that it's less efficient due to the tedious and duplicated computation and comparison of prefix and suffix hash. Rabin-Karp seems suitable for our purpose, but it requires the patterns to have a fixed length, thus we cannot use it directly.

Here we modified Rabin-Karp so that it can search for multi-patterns of both length 2 and 3. We replaced the set of hash values of pattern prefix with a hash map. The keys of hash map are hash values of pattern prefixes (prefix length is 2); the value of hash map is 0 for patterns of length 2; the one character following the prefix (the last character) for patterns of length 3. When current substring's hash equals any key in the hash map, we retrieve the corresponding value in the hash map. If a non-zero value is encountered, just compare the non-zero value (the third character in pattern) with the character following the prefix, a match (pattern of length 3) is found if the two equals. If the value we get is zero, a match is found immediately (pattern of length 2).

We further optimized Rabin-Karp by selecting a natural rolling hash function. In our modified version of RK, hash is calculated on two Chinese characters since prefix length is 2. The length of a Chinese character is two bytes in Unicode and many other encoding, thus the length of 2 Chinese characters equals the length a natural WORD (int) on 32-bit machines. Based on this observation, we take the four bytes code of 2 Chinese characters directly as its hash. This straightforward hashing has the following advantages:

- The hash value does not need any addition computation
- The probability of collision is zero.

Experiment shows that our modified RK outperforms the improved WM [11] in character reconstruction.

3.2 Keyword Filtering

Keyword filtering is also a problem of multi-pattern matching. Since the minimum length of all keywords is 2, WM is still not a good choice for keyword filtering due to the overlapping of prefix and suffix. We still choose to use RK. We need to modify RK further, since length of keywords (patterns) is mostly between 2 and 5 this time. We used the same straightforward rolling hash as in section 3.1, since prefix length is still 2. We still replace the set of hash values of pattern prefix with a hash map. We keep keys as the same in section 3.1, but use pointers pointing to the character following the prefix as values. When current substring's hash equals any key in the hash map, we retrieve the corresponding value in the hash map as before. Then compare the string pointed to by the retrieved pointer with the string starting from the character following the prefix to see if they are a match. Since most of our keywords are short, there won't be plenty of character comparisons, thus the algorithm is quite efficient.

4 Experiments

To demonstrate the effectiveness of our filtering system, we used the same harmful document collection from National Engineering Laboratory of Content Analysis as mentioned in Section 1, 2 as test data. We selected 752 words in all documents as the keywords to filter. These words show up 21973 times in all documents.

We input the document collection (6466 documents in all) into the filtering system. Our filtering system reconstructed those decomposed characters, and then applied keyword filtering on the processed text. A document is filtered out if it contains any keywords. The result in table 3 shows that our filtering system can recognize most of the keywords even if characters of these keywords are decomposed into radicals. As a comparison, we applied keyword filtering on the input without reconstructing characters from radicals.

Table 3. Effect of our filtering based on Character Reconstruction

	Keyword Matches	Filtered Documents
Filtering based on character reconstruction	99.57% (21878)	99.77% (6451)
Filtering without reconstruction	91.36% (20074)	92.11% (5956)

As shown in table 3, our approach can effective identify most of the keywords even in the form on combination of radicals. It yields a visible improvement in the filtering result compared to traditional filtering without character reconstruction. As more and more evil sites begin to use this trick, and the proportion of harmful documents containing intentionally decomposed characters increases, the improvement will be more significant in the future.

However, our approach also has its drawbacks. From table 3, we can see that there're still some keywords that cannot be identified with our approach (about 0.23%). Since the first radical of a character might be combined with character left to it mistakenly, some keywords cannot be identified. E.g. for "大小和卓木半反乱", "木" and "半" is combined into "柈" mistakenly, thus keyword "叛乱" cannot be identified after reconstruction. Our current approach cannot handle this kind of situations. To eliminate such kind of wrong combinations in future work, we can take semantic into consideration when recombining radicals.

We also tested the performance of our character reconstruction algorithm. It shows that our modified Rabin-Karp algorithm outperforms the improved Wu-Manber algorithm proposed in [11] by 35% on average in character reconstruction. To further improve the performance of the whole system, we can even consider combining character reconstruction and keyword filtering into one step in future work, using decomposed keywords as patterns. This would cause the hash table in Rabin-Karp to blow, since there might be several ways to decompose a single keyword. And it's trading space for speed.

5 Conclusions

Decomposing Chinese characters to bypass traditional keyword filtering has become a popular trick that many evil sites use now. In this paper we proposed a filtering technology against this kind of trick. We first use a modified Rabin-Karp algorithm to reconstruct Chinese characters from radicals. Then apply keyword filtering on the processed text. This is the first filtering system ever known against the trick. Experiment has showed the effectiveness and efficiency of our approach. In the future, we can further improve the filtering technology by taking semantic into consideration when recombining characters or even try to combine reconstruction and filtering into a single step.

Acknowledgement. The work described in this paper is fully supported by the National Natural Science Foundation of China (No.60703032), and the Opening Project of Key Lab of Information Network Security of Ministry of Public Security.

References

1. Oard, D.W.: The State of the Art in Text Filtering. User Modeling and User-Adapted Interaction 7(3) (1997)
2. Zhang, X.: Research of Chinese Character Structure of 20th Century. Language Research and Education (5), 75–79 (2004)
3. Karp, R.M., Rabin, M.O.: Efficient randomized pattern-matching algorithms. IBM Journal of Research and Development 31(2) (March 1987)
4. Chinese Linguistics and Language Administration. GB13000.1 Chinese Character Specification for Information Processing. Language and Literature Press, Beijing (1998)
5. Li, X.: Discussion and Opinion of the Evaluation Criterion of Chinese Calligraphy, `http://www.wenhuacn.com/`
6. Lee, R.J.: Analysis of Fundamental Exact and Inexact Pattern Matching Algorithms
7. Knuth, D.E.: Fast Pattern Matching in Strings. SIAM J. Comput. 6(2) (June 1977)
8. Boyer, R.S., Moore, J.S.: A Fast String Searching Algorithm. Communications of ACM 20(10) (October 1977)
9. Aho, A.V., Margaret, J.C.: Efficient string matching: An aid to bibliographic search. Communications of the ACM 18(6), 333–340 (1975)
10. Wu, S., Manber, U.: A Fast Algorithm for Multi-Pattern Searching. Technical Report TR 94-17, University of Arizona at Tuscon (May 1994)
11. Yang, D., Xu, K., Cui, Y.: An Improved Wu-Manber Multiple Patterns Matching Algorithm. IPCCC (April 2006)
12. Sunday, D.M.: A very fast substring search algorithm. Communications of the ACM 33(8), 132–142 (1990)