# Efficient Extraction for Mobile Web Access Log with Caching Strategy

Lifeng Gao[1], Min Zhu[1*], Mengying Li[1], Yu Cao[2], Weixue Zhang[1]
[1] College of Computer Science, Sichuan University, Chengdu, China.
[2] EMC Labs, Beijing, China.

* Corresponding author. Email: zhumin@scu.edu.cn
Manuscript submitted May 21, 2015; accepted July 27, 2015.
doi: 10.17706/jcp.11.2.99-108

**Abstract:** Mobile web access log file plays an important role in the analysis about demand of mobile terminal market or user behavior. However, the log file data is highly dimensional, disorganized and semi-structured, which heightens the difficulty of data extracting accuracy; while it generates and transmits continuously, which poses an extracting efficiency challenge. It is highly desirable to extract the information embedded in log files as disorder or hidden situation efficiently and accurately. This paper proposes an efficient extracting method for mobile web access log with cache strategy. Firstly of all, data dictionary sets are built for each kind of complex field before extracting. Then, data is extracted based on the dictionaries, and the dictionaries will be completed simultaneously. Furthermore, with the discovery of the distribution of some data in the log generally following Zipf-like distribution, cache strategy is considered to be an auxiliary way to reduce mapping time. In addition, the classical cache strategy LFU is chosen. Ultimately, the experiment shows that the data could be extracted from the log accurately, and the extracting efficiency speeds up remarkably with LFU cache strategy.

**Key words:** Data extraction, mobile web access log, data dictionary, cache strategy, Zipf-like distribution.

## 1. Introduction

More and more people are used to surfing the Internet to acquire knowledge. Simultaneously, web access log files grow explosively. WUM (Web Usage Mining) is a technology to mine data from web access data log, aiming to find and analyze usage patterns [1]. Moreover, data preprocessing occupies about 80% time of the WUM, which aims at making the log data structured [2]. However, data extraction is the foundation of data preprocessing. Therefore, the efficiency of data extraction becomes the bottleneck in the entire preprocessing and WUM.

Compared with other web access log, information in mobile web access log is more abundant (Time, Host, User-Agent, etc.) and heterogeneous, where disorder or lacks of information items occur frequently. Furthermore, some tokens in the log distribute asymmetrically, which follow Zipf-like distribution through statistical analysis. Finally, it has strong real-time performance as flow data, which leads to the sustained high growth of log data. Therefore, the key point is how to improve the efficiency of data extraction in a certain range of accuracy [3], and the difficulty is how to extract a variety of tokens from one field. However, the bulk of the existing methods focus on the HTML or XML files, into which the mobile web access log files are difficultly transformed. Additionally, traditional pattern recognition methods mostly need embedding the pattern in the programs, which makes it hard to update to meet requirement of accuracy flexibly.

In this paper, data dictionary set is built before data extraction and completed dynamically when the data is extracted from mobile web access log, which acts as the original method. At this foundation, importing LFU cache strategy to accelerate the matching and extracting speed extends this original method. The experiment shows that our method has a good extracting accuracy. Furthermore, the experiment is also done on the classical cache strategies, FIFO and LRU. And the result gives the evidence that the extended method has higher extracting efficiency than that using FIFO or LRU cache strategies or using no cache strategy.

## 2. Related Work

Data extraction can be used widely, from Electronic medical records (EMRs) [4] to web page, which is the foundation of relative research. There are many web data extraction tools [5] and methods.

The method proposed by paper [6] aims at extracting web geospatial data. DEBy [7] extracts data from web sources based on a small set of examples specified by the user and adopt a bottom-up procedure. FiVaTech [8] applies tree matching, tree alignment, and mining techniques to achieve data extraction. CTVS [9] automatically extracts data from query result pages by identifying, segmenting and aligning, which combines both tag and value similarity. ViDE [10] extracts structured data from deep web pages including data record extraction and data item extraction. However, these methods mentioning above extract data from web pages, based on analyzing the HTML source code or the tags of the web pages [10].

The method proposed by paper [11] designs a extraction system to extract tourism industry data based on XML. The method proposed by paper [12] parses a web document into an extended DOM tree and proposes a web information extraction algorithm based on Hidden Markov Model.

Whereas, there are not definitions of the relevant tags in the mobile web access log. The log data is disorder and mixed together. As a result, it is hard to convert the log files into HTML or XML files, so the above methods are not applicable. Therefor, some extracting methods focuses on the mobile web access log files are desirable urgently.

## 3. Data Analysis

When a mobile terminal device accesses web servers, the access information will be stored as text format like a HTTP request message to form the web access log. The mobile web access log files on which this paper focusing are collected from a region of city in China.

### 3.1. Data Structure

The original log file is composed of data records separated by specific delimiters. Moreover, each record contains several fields and each field records a part of information about the mobile terminal accessing the web server. In addition, each field can be divided into two parts: field name and field value; the former represents the type of this field and the latter is a string made up of various tokens, the detailed descriptions (as shown in Fig. 1).

There are two kinds of field:

*Simple field:* the field value only contains a single token, such as the host field.

*Complex field:* the field value contains a series of tokens with noise. In other words, the field value is a token vector. For example, the User-Agent field value is a token vector including terminal brand, operating system and browser.

Furthermore, the log file data is semi-structured. Some records contain only one field, while some contain more than twenty fields. However, there are more than 300 types of fields and different unions of fields usually appear in different records. And each kind of field in the log appears with different frequency. For example, the proportion of Time, Host and User-Agent field respectively is 100%, 81%, and 76%.

```
----------------------------------------
1333967945 [10.99.29.15:2157->10.99.192.13:2158] 10.138.150.32->10.0.0.172
GET /comm/v2/result.jsp?sid=AeJBkJkfNNG5_0KWIOb5v-CE&activeId=12&aId=5678&answer=1 HTTP/1.1
accept:
application/vnd.wap.xhtml+xml,application/xml,text/vnd.wap.wml,text/html,application/xhtml+xml,image/jpeg;q=0.5,image/png;q=0.5,image/gif;q=0.5,image/*;q=0.6,video/*,audio/*,*/*;q=0.6
user-agent: MQQBrowser/3.1/Adr (Linux; U; 2.1-update1; zh-cn; GT-I5503 Build/ERE27;240*320)
referer:
http://sq8.3g.qq.com/comm/v2/result.jsp?sid=AeJBkJkfNNG5_0KWIOb5v-CE&activeId=12&aId=5582&answer=3
cookie:
sd_userid=48481328000295684;sd_cookie_crttime=1328000295684;qq_mb_adv_special=-344946998|1328187419173;pt=2;mtt_cache_ck=20120215191046;stock_uin=wi+MQ0+2xt47xQXiDr0YnyZ6ECM2EYfG;3g_lastLoginQq=907995;3g_csp=1333254835;info_lau=907995;appsd_mid=1904;softdown_mid=1904;match_mid=-1;softdown_pid=14;g_ut=2;info_index_att=1;icfa=content_rela;
----------------------------------------
```

Fig. 1. Example of a record in log files.

### 3.2. Data Feature

In complex fields, tokens in the same kind of field are distributed chaotically and emerged randomly, such as the terminal brand in the User-Agent field. Fig. 2. shows the frequency-rank distributions of the brands though browsing 4, 211,828 User-Agent fields, where 1,487,039 fields are devoid of brand tokens.

Based on Fig. 2., the product of rank of brands and their frequency is roughly a constant, which belongs to the Zipf distribution [13].

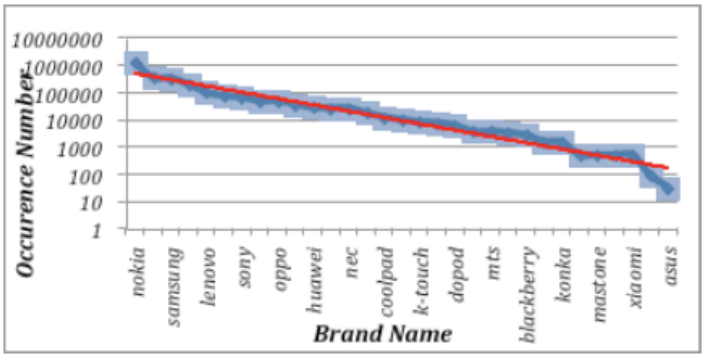



Fig. 2. Frequency-rank distributions in brands.

## 4. Extraction Method

The data extraction can be divided into three steps, just as the Fig. 3.

STEP 1: Data Segmentation. The log file data is extracted record by record, and the extracting method is based on fields. So the log file should be divided into records and the record should be split into fields with the specific delimiter. Then, the field name will be gained.

STEP 2: Data Extraction. In terms of the field name, the field type is distinguished to choose corresponding method to extract data, such as direct extraction, split and transformation, or extraction based on dictionaries.

STEP 3: Data Storage. The extracted data belonging to the same record is emerged into one token vector. Finally, this vector is recorded in the data table with the specific form.

For different field types, extracting methods will be introduced as follows.

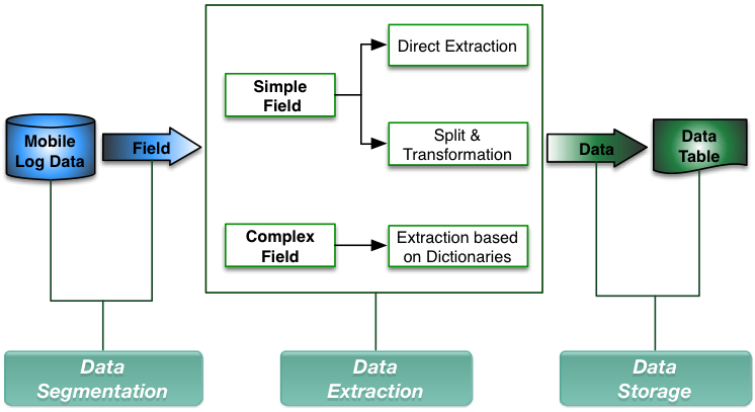



Fig. 3. Data extraction chart flow.

## 4.1. Extraction in Simple Field

In simple field, the location of token in field is explicit, or there is explicit boundary. If the field contains a single token, the token should be extracted directly. Otherwise, if the field contains other various tokens or noise, the right token should be filtered through splitting the field value. If the extracted token is difficult to recognize, this unclear token should be transformed into normal form. For instance, the timestamp is an integer of 10 digital, 1333967945, which is extracted from field "1333967945 [10.99.29.15:2157-> 10.99.192.13:2158] 10.138.150.32->10.0.0.172". And it will be converted to 2012-04-09 18:39:05.

## 4.2. Extraction in Complex Fields

Complex field contains much information and there is lack of fixed structure to tell different tokens, which makes the extraction more difficult. There are two stages to solve it, building data dictionary and extracting data. Before extraction, we should build dictionaries for each kind of token in this field.

1) Building Data Dictionary Set

*Input: sample files of mobile web access log*
*Output: corresponding data dictionary set of each complex field*

$N$ = the number of genus of complex field by travelling through the sample files;
$FieldNames[N]$://preserve the field names
$Field\_set[N]$://gather fields into one set within the same field name in $FieldNames$
$Dict\_set[N]$://store the dictionary sets for each genus of complex field
*for*(int $i = 0; i < N; i++$)
$M$ = the number of species of tokens in the field named $FieldNames[i]$;
$List$://record the species name
Dictionary[M]://the dictionary set for $FieldNames[i]$, each named by the token in List
*for*($T$ *in List*){
*for*(*field in* $Field\_set[i]$){
*if*($A$ *is in field belonging to* $T$ *but not in* $Dictionary_T$){
$Dictionary_T.add(A)$;
}
}//a dictionary relative $T$ was founded
$Dict\_set[i].add(Dictionary_T)$://build the dictionary set for correlative $FieldNames[i]$
}//a dictionary set relative $FieldNames[i]$ was founded
}
return $Dict\_set$://the whole data dictionary set is built

Some log files are chosen randomly from the mobile web access log files, which are going to be handled as sample files to build data dictionary set. Generally, a genus of complex field comprises several tokens.

Therefore, a list should be built to corresponding field through travelling the whole sample files, which records the species of tokens appearing in this genus of field. Finally, a set of dictionaries is built based on the list. For example, the User-Agent field composes the brand, operating system and browser of the terminal. So the list of User-Agent is defined as $List_{user\text{-}agent} = \{Brand, OS, Browser\}$, and its dictionary set is defined as $Dict\_set_{user\text{-}agent} = \{Dictionary_{brand}, Dictionary_{OS}, Dictionary_{browser}\}$ The process of building data dictionary sets as above.

The dictionaries are relative small because of different log files focus on specific spatial-temporal information. For instance, new ones unremittingly about mobile terminals replace old versions. Additionally, among thousands of kinds of browsers, $Dictionary_{browser}$ contains only dozens of that to present this whole mobile web access log files. What's more, the dictionaries are updated easily, which can adapt the changing of mobile terminal market and relative software, etc. Consequently, it is possible to put the data dictionaries into memory to achieve high efficiency. Meanwhile, the data dictionaries could be stored as files outside the programmer, which is easy to update, with strong flexibility.

2) Data Extraction

When extracting, the complex field should be mapped to the relative dictionary set. Fig. 4 shows the process that tokens are extracted from the field and merged into a token vector as the result for further investigation. Finally, this process is defined as the original method.

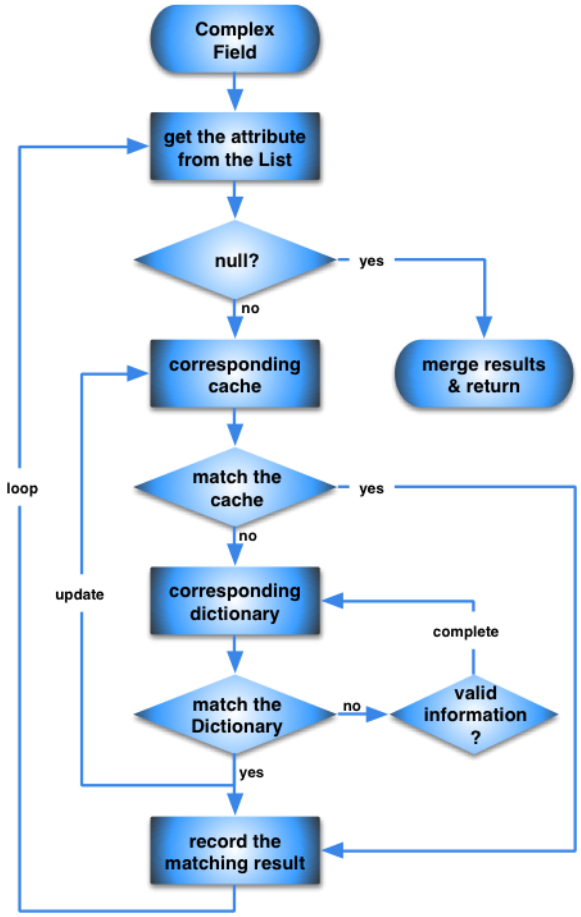



Fig. 4. Extraction in complex.

To each complex field, there are some disorder information items, lack of information items, or various appearance items, which raise the difficulties of extraction. For instance, Moto and Motorola mean the same brand, as well as Samsung, GT-S5820, and GT-C3050C.

*T* presents a species of tokens in field *F*, and $\text{Dictionary}_T$ is the relative dictionary about *T*. When the field *F* matches nothing in $\text{Dictionary}_T$, there are some reasons that the $\text{Dictionary}_T$ is inadequate or the mobile terminal market has changed. So $\text{Dictionary}_T$ should be updated. Therefore, some information may be added into the $\text{Dictionary}_T$ to complete this dictionary as follows.

STEP 1: Judge whether there is a token in this field belongs to *T*. If it is, mark the corresponding information as *A*, and go to STEP 2; otherwise, ignore this field.

STEP 2: If there are no relative tokens indicating the same value of *A* in $\text{Dictionary}_T$, then put it into the $\text{Dictionary}_T$ directly; else, go to STEP 3.

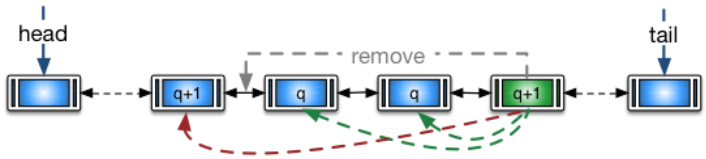
STEP 3: There must be at least one token $A'$ in $\text{Dictionary}_T$ presenting the same value of *A* in another way. If we can abstract a rule by regular expression from *A* and $A'$, then replace $A'$ with this rule; otherwise, just put *A* into $\text{Dictionary}_T$ as another form of this attribute value.

For example, there are Samsung and GT-S7568 in $\text{Dictionary}_{\text{brand}}$, but GT-C3050C can't be matched. So we abstract regular expression "\bgt-\w?\d+" as a rule and put it into $\text{Dictionary}_{\text{brand}}$ to replace GT-S7568.

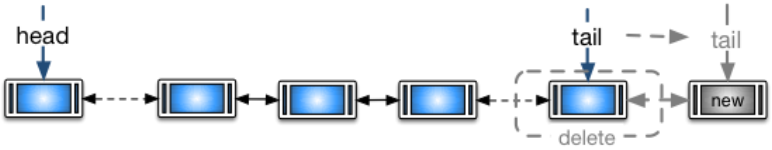
However, if data dictionaries are updated immediately with the failure of matching, the workload will be high. To lessen it, the updating will triggered when the extracting rate is less than the standard, which will be introduced in detail in Section 5.3.

Just as mentioned in Section 3.2, the mobile web access log data fits Zipf distribution. At this time, adopting cache strategy will promote the operating [14]. Based on the original method mentioning above, the LFU (Least Frequently Used) cache strategy is considered to boost the efficiency. The essential point is to find an excellent balance the extracting time and the memory size [15], [16].

LFU is optimization in the whole processing while is bad performance when data accesses abruptly. Fig. 5 explains how to update the cache.




(a) Update with matching




(b) Update without matching

Fig. 5. Update cache with LFU.

## 5. Results

### 5.1. Extraction in Complex Field

The method in this paper can extract data accurately from complex fields, and transform the information to tokens easily recognized. Table 1 shows the extracting results.

### 5.2. Extraction in Mobile Web Access Log

The method proposed by this paper is applied to extract data from mobile web access logs. Table 2 shows the target data table with the attributes of Time, Host and User-Agent fields.

Table 1. Extracting Results of Complex Fields

| Raw Field | Extracting Results | | |
|---|---|---|---|
| User-Agent | Brand | OS | Browser |
| MQQBrowser/2.0 (Nokia5233;SymbianOS/9.1 Series60/3.0) | nokia | symbian | mqqbrowser |
| GT-S5820_2.3.6_weibo_2.8.1_android | sumsung | android | weibo |
| E63/SymbianOS/9.1 Series60/3.0 | nokia | symbian | null |
| Mozilla/5.0 (Linux; U; Android 2.3.5; zh-cn; TCL_A919 Build/GRJ90) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1 | htc | android | safari |
| MAUI_WAP_Browser | null | null | maui_wap_browser |

Table 2. Data Table

| Time | Host | Brand | OS | Browser |
|---|---|---|---|---|
| 4/9/2012 18:52 | baidu | htc | android | mozilla |
| 4/9/2012 18:52 | qpic | symbian | symbianos | mqqbrowser |
| 4/9/2012 18:52 | weibo | iphone | iphone | null |
| 4/9/2012 18:52 | sina | iphone | iphone | mozilla |
| 4/9/2012 18:52 | soso | null | null | maui |

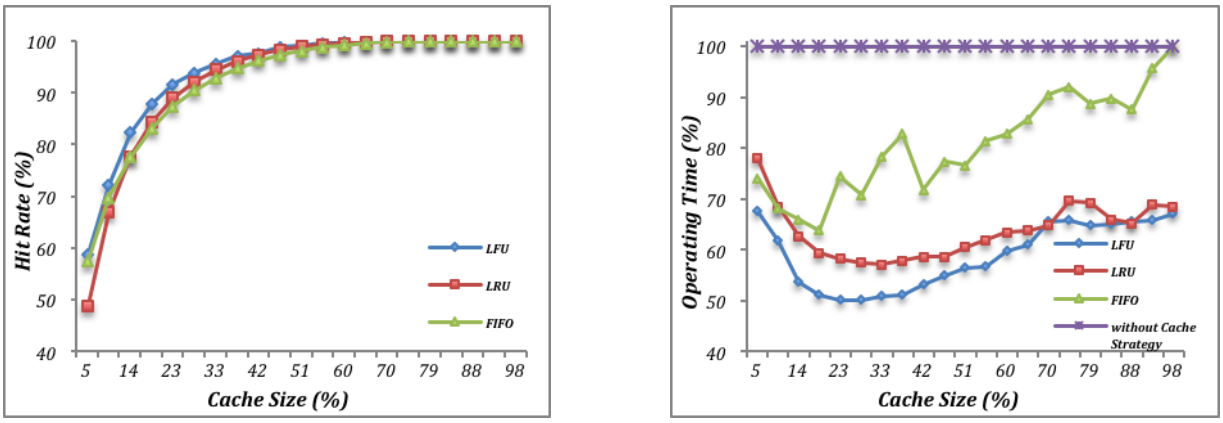
## 5.3. Performance Comparison

Theoretically, the accuracy of extraction can achieve 100% if the dictionaries are adequate. However, there are lacks of information items in complex fields. Table 3 shows the extracting rate of tokens in the complex field User-Agent.

Table 3. Extracting Rate of Tokens in User-Agent

| Token | Brand | OS | Browser |
|---|---|---|---|
| Extraction Rate | 65% | 51% | 54% |

According to the extracting rate in complex fields, the dictionaries should be updated selectively. Through the statistics results about collected complex fields without matching relative dictionaries periodically, the rate of collection about a certain token $T$ is marked as $\text{Rate}_{T\text{-collect}}$, and the current extracting rate about this token is marked as $\text{Rate}_{T\text{-collect}}$. If $\text{Rate}_{T\text{-collect}} > 1 - \text{Rate}_{T\text{-collect}} + T_{\text{threshould}}$, the $\text{Dictionary}_T$ will be updated to complete. $T_{\text{threshould}}$ is responsible to adjust the frequency of completing $\text{Dictionary}_T$. Therefore, it avoids updating the dictionaries at all times, which lowers the workload substantially.




(a) Hit rate (b) Operation time

Fig. 6. Comparisons of three cache strategies.

The efficiency of this method aiding with LFU cache strategy is higher than the original method without

any cache strategies or the original method with FIFO or LRU cache strategies. Fig. 6(a) shows the hit rates of LFU, LRU, and FIFO changing with the cache capacities. They are almost the same trend and LFU has the relative highest hit rate. In the figure, cache size means the proportion of the current size to the corresponding dictionary size.

Fig. 6(b) shows the efficiency of four methods – original method, original method with respective LFU, LRU and FIFO cache strategies. It is easy to see the original method's efficiency is the lowest. And with the increase of cache capacity, the efficiency of three extended methods roughly increases and then declines. That means when cache is too small, large amount of replacement will appear and when cache is too large, the cost raises acutely. However, the extended method with LFU cache strategy has the best performance.

To get the balance between time and cost, 23% cache size is chosen, which has the shortest operating time and relative small cache capacity. At this point, the extended method has the best efficiency.

## 6. Conclusion and Future Works

This paper proposes a method for extracting data from mobile terminal web access log. With distinguishing the field type, different extraction method is chosen to solve the difficult problem of processing the unstructured log data. For the extraction of complex fields, the establishment of data dictionaries as well as maintaining the corresponding caches is applied to improve extraction efficiency. Constantly completing data dictionary can ensure the extraction precision, better adaption and good robustness. This paper compares the extraction efficiency and hit rate of different cache strategies, and analyzes the results.

In the future work, dynamic cache capacity will be used to adapt to the data features to improve extraction efficiency and minimum cost. At the same time, the date dictionary will be completed by pattern recognition or other effective methods semi-automatically or automatically.

## Acknowledgment

This work is a part of the joint research project with EMC Labs China (the Office of CTO, EMC Corporate), which is funded by EMC China Centre of Excellence. It is also a part of Science and Technology Support Program of Sichuan Province, China (No. 2013GZ0015).

The authors gratefully acknowledge the helpful comments and suggestions of the reviewers. Our thanks would also give to Mr. Jun Wang for his advices.

## References

[1] Langhnoja, S., Barot, M., & Mehta, D. (2012). Pre-processing: Procedure on web log file for web usage mining. *International Journal of Emerging Technology and Advanced Engineering, 2(12),* 419-423.

[2] Losarwar, V., & Joshi, D. M. (2012). Data preprocessing in web usage mining. *Proceeding of International Conference on Artificial Intelligence and Embedded Systems* (pp. 15-16).

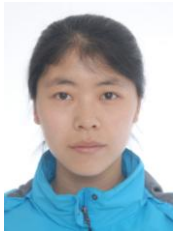
[3] Mayer-Schönberger, V., & Cukier, K. (2014). Big data: A revolution that will transform how we live, work, and think. *American Journal of Epidemiology, 17(1)*, 181-183.

[4] Martinell, M., Hallqvist, J., *et al.* (2012). Automated data extraction — a feasible way to construct patient registers of primary care utilization. *Ups. J. Med. Sci., 117(1)*, 52-56.

[5] Laender, A. H. F., Ribeiro-Neto, B. A., Silva, A. S. D., & Teixeira, J. S. (2002). A brief survey of web data extraction tools. *ACM SIGMOD Record, 31(2)*, 84-93.

[6] Shi, G., & Barker, K. (2011). Thematic data extraction from Web for GIS and applications. *Proceeding of 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services* (pp. 273-278).

[7] Laender, A. H. F., Ribeiro-Neto, B., & Da, S. A. S. (2002). Debye-data extraction by example. *Data & Knowledge Engineering, 40(1)*, 121-154.
[8] Kayed, M., & Chang, C. H. (2010). Fivatech: Page-level web data extraction from template pages. *IEEE Transactions on Knowledge & Data Engineering, 22(2)*, 249-263.
[9] Su, W., Wang, J., Lochovsky, F. H., & Liu, Y. (2012). Combining tag and value similarity for data extraction and alignment. *IEEE Transactions on Knowledge & Data Engineering, 24(7)*, 1186-1200.
[10] Liu, W., Meng, X., & Meng, W. (2009). Vide: a vision-based approach for deep web data extraction. *IEEE Transactions on Knowledge & Data Engineering, 22(3)*, 447-460.
[11] Jiang, H. C., Wang, D. L., Ban, X. J., & Jin-Xi, R. (2009). Web data sime-automatic extraction based on xml. *Computer Engineering, 35(21),* 51-53.
[12] Lai, J., Liu, Q., & Liu, Y. (2010). Web information extraction based on hidden Markov model. *Proceeding of 2010 14th International Conference on Computer Supported Cooperative Work in Design* (pp. 234-238).
[13] Li, W. (2003). Zipf's law everywhere. *Glottometrics, 5*, 14-21.
[14] Adamic, L. A., & Huberman, B. A. (2002). Zipf's law and the internet. *Glottometrics, 3,* 143-150.
[15] Chen, M. (2004). An analysis of some problems in grid cache. *Computer Science, 31(5)*, 15-17.
[16] Yong, L. I., Peng, Y. X., & Chen, F. J. (2000). Disk cache replacement algorithms for large scale video on demand system. *Journal of Computer Research & Development, 37(2)*, 207-212.



**Lifeng Gao** was born in Hebei, China in January 1988. She received the B.E. degree from Shijiazhuang Tiedao University, China, in 2012, and she is currently working toward the M.E. degree in College of Computer Science, Sichuan University, China. She is also a member of CCF. Her research interests mainly focus on visualization, vision-computing, and data mining.

**Min Zhu** received her B.Sc. degree in computer application from Hebei University of China, in 1993, and received her M.Sc. and Ph.D. from Sichuan University of China in computer application in 1996, in applied mathematics in 2004, respectively. She joined the College of Computer Science of Sichuan University as an assistant in 1996, and at present she is a professor and the vice dean, as well as the director of the Lab for Vision-Computing at College of Computer Science. She has wide research interests, mainly including visualization, image processing, intelligent information processing and wireless network. She is a member of IEEE and CCF (China Computer Federation) and the chair of the 2011-2012, 2012-2013 YOCSEF Chengdu separate forums of the CCF. She is an evaluation expert of of the NOSTA (National of Science & Technology Awards), the MOST (Ministry of Science and Technology), the MOE (Ministry of Education), and National Science Foundation of Beijing and Zhejiang Provinc.

**Mengying Li** was born in Hebei, China in November 1987. She received the B.E. degree from Sichuan University, China, in 2012, and she is currently working toward the M.E. degree in College of Computer Science, Sichuan University, China. She is also a member of CCF. Her research interests mainly focus on visualization, vision-computing, and data mining.

**Yu Cao** obtained his bachelor degree from University of Science and Technology of China in 2005, and his Ph.D degree from National University of Singapore in 2011, both in computer science major. Currently he is leading the applied research and solution innovation activities at EMC Labs of EMC Corporation, focusing on big data, cloud and trust.

**Weixue Zhang** was born in Shandong, China in September 1990. She received the B.E. degree from Ludong University, China, in 2012, and she is currently working toward the M.E. degree in College of Computer Science, Sichuan University, China. She is also a member of CCF. Her research interests mainly focus on image processing and visualization.