Front. Electr. Electron. Eng. China 2011, 6(2): 388–396
DOI 10.1007/s11460-011-0137-z

RESEARCH ARTICLE

Weining QIAN, Aoying ZHOU, Minqi ZHOU

A best-effort approach to an infrastructure for Chinese Web related research

Abstract The design of the infrastructure for Chinese Web (CWI), a prototype system aimed at forum data analysis, is introduced. CWI takes a best effort approach. 1) It tries its best to extract or annotate semantics over the web data. 2) It provides flexible schemes for users to transform the web data into eXtensible Markup Language (XML) forms with more semantic annotations that are more friendly for further analytical tasks. 3) A distributed graph repository, called DISGR is used as backend for management of web data. The paper introduces the design issues, reports the progress of the implementation, and discusses the research issues that are under study.

Keywords Chinese Web infrastructure, semantic entity, graph data model, distributed storage

1 Introduction

The World Wide Web (WWW, or the Web) contains huge volumes of data published by different organizations or people, which can be treated as the largest distributed database in the world. However, utilization of data on the Web is difficult since the data is unstructured, heterogeneous, and fast changing. We propose to develop an infrastructure for Chinese Web (CWI) related research to serve the requirements of analysis on data from the Chinese Web [1].

One focus of CWI is the analysis of Chinese forum data. It is reported that, in 2009, more than 30% of people with Internet experience in China accessed forums [2]. Forum data are fast changing and can provide on-time information about news events. Furthermore, forums usually contain rich information about users' interests, opinions, and social networking activities. However, though many forums provide local search functions, there are few public services that can be used for analyzing inter-sites forum data.

In this paper, the design of CWI is introduced. The backend of CWI, including a crawler for forum data, a semantic annotator that utilizes both internal features of web pages and external resources, and a distributed graph repository, are presented. One characteristic of CWI that distinguishes itself from other systems is that it takes a *best-effort* approach, especially from the semantic extraction perspective:

1) CWI only crawls part of the Chinese Web. However, the crawled web pages are not treated as a closed world. External links are preserved. Furthermore, while needed, external resources are retrieved, parsed, and analyzed, so that related information are extracted for further analysis of the crawled data.

2) CWI does not try to construct a database fitting to pre-defined schema. However, it accepts templates that are used to model any piece of information that can be extracted from crawled web data. Thus, CWI organizes the data in a bottom-up manner. Though CWI may only provide a database with incomplete, and sometimes even inconsistent information, we believe that this is a practical approach to supporting analytical web applications.

3) CWI provides flexible operators that are effective for web data retrieval and analysis, while being friendly for efficient implementation in large-scale clusters.

The rest of the paper is organized as follows. The architecture of the system is introduced in Sect. 2, followed with, in Sects. 3, 4, and 5, the three most important parts in CWI, i.e., the data collector, the semantic annotator, and the graph storage and indexing engine. After the description of related works in Sect. 6, the last section is for discussion and concluding remarks.

Received July 22, 2010; accepted January 26, 2011

Weining QIAN (✉), Aoying ZHOU, Minqi ZHOU
Institute of Massive Computing, Software Engineering Institute, East China Normal University, Shanghai 200241, China
E-mail: wnqian@sei.ecnu.edu.cn

2 Architecture

The information processing workflow of CWI is shown

in Fig. 1. The *crawler* retrieves web pages continuously. The web pages are retrieved and parsed, so that its internal structures are obtained. Then, some basic information is extracted based on descriptive extraction rules by the *extractor*, after which the results are represented in eXtensible Markup Language (XML) forms. These internal XML data may be further analyzed by *annotators*, which analyze content and structure of the XML data, and add annotations to semantic entities in these data.

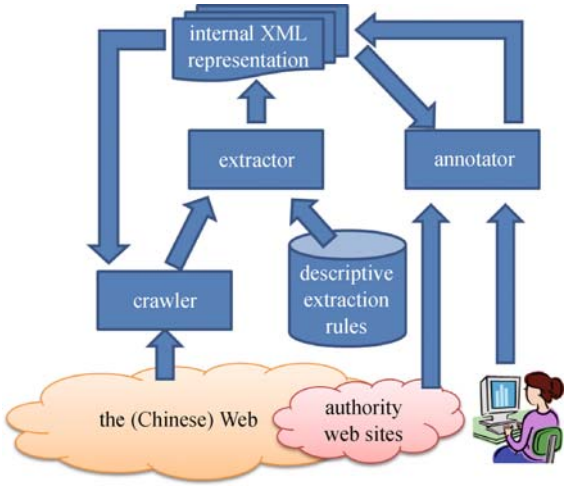


Fig. 1 Architecture of the system

XML data are modeled as graphs of the Tagged Graph Model (TGM), a graph data model designed for web data management in CWI. The data are then stored in a distributed repository that supports efficient retrieval, transformation, and aggregation of subgraphs. The operators with parallel processing are implemented by using Map/Reduce programming model [3].

3 Data collecting

There are two tasks for data collecting in CWI. First, data should be *crawled* from remote web sites. This is a task that most crawlers are able to do. The second one is parsing and basic information extraction. Though most crawlers are able to parse a web page and identify hyperlinks in it, they are not able to identify more detailed information, such as the author of a post in a web forum. Apparently, it is a difficult and even impossible mission to build a general extractor for any kind of information. However, we argue that extraction should be a native function of a web data collector for the following reasons:

1) The atomic information blocks are usually not web pages. For example, one web page of a web forum may contain multiple posts, each from a different author with different content. Thus, taking a web page as the finest information block is inappropriate.

2) Related information may or may not appear in the same web page. Again, the list of posts of a web forum may span over a series of web pages, which are highly related in terms of content and time.

3) The crawler should schedule the crawling tasks based on those basic information. These kinds of relationships should be taken into consideration in crawler scheduling and crawled data storage. For example, the same post in a web forum may appear in two different web pages crawled at different times. Crawlers without extractors have difficulty in identifying such duplication.

The data collector of CWI integrates web page crawling with structure-based information extraction. It is different to other crawlers in that web pages crawled are fed into a rule-based structure extractor, which transforms original web pages in hyper text markup language (HTML) representation into XML representation. Basic information blocks, such as post (in a web forum), authors, timestamp of publishing, title, content, etc. may be tagged based on rules. The rules can be modified and extended, so that different structures and modifications on structures of web sites are handled.

3.1 Rule-based structure extraction

An *XPath* [4] is a series of *steps*, in which each step is evaluated in a specific *context*, i.e., a node in XML document tree or a list of nodes. Each step is of the form $axis :: node-test[P_1][P_2]\cdots[P_n]$. Here, $axis$ denotes the directions of a step, $node-test$ denotes the type of nodes to be selected, and P_i are predicates of any type of XPath expressions that returns a boolean value.

An *XPath pattern* is an XPath in which any element name, attribute name, axis, or node-test function name can be a variable (started with a '$'). A *CWI rule* is of the form $head :- body$, in which both head and body are a set of XPath patterns. It is required that there is only direct parent or child axis appearing in head, and there is no free variables in *head*. Furthermore, all XPath patterns in body should be evaluated only on source XML documents. Each rule is interpreted as an action that *whenever all XPath patterns in the body is matched, XPath patterns in head are inserted into the target XML document.*

The CWI rule actually is a subset of eXtensible Stylesheet Language Transformation (XSLT) [5], and it is currently implemented by using an XSLT engine. Each CWI rule is transformed to a set of XSLT templates, which is introduced briefly as follows:

1) Each CWI rule is transformed into an XSLT template, in which the body of the template corresponds to the head of the rule;

2) For each pattern in the rule body, if it contains no variable, then it is translated to the pattern in the template directly;

3) Else, for patterns joined by some common variable(s), each join is translated to a new XSLT template, which matches one pattern, and call the template of next pattern or the one corresponding to the rule.

There are several advantages of this rule-based information extraction.

1) First, it is flexible and extensible. Rules may be modified or added, so that different structures or changes of structures can be handled.

2) Second, it is appropriate for extraction tasks based on replicated structures. It is rather useful for extraction of basic information from Web 2.0 web sites. Currently, this scheme is used for extraction of information from two web forums, and three twitter-style sites.

3) Last but not the least, it is friendly to machine learning methods, since some important machine learning or data mining tasks, such as association rule mining and rule-based learning, use rules as models.

4 Semantic annotation

The basic information extracted by the data collector is coarse, and contain few semantic information. The annotators are responsible for the task of semantic information extraction. Currently, CWI focuses on extracting semantic entities of type time, location, people, and events. Information extraction is a hot topic in both the industrial and research community. There are a few important technologies available for building an effective information extraction system [6].

4.1 Semantic entities

We define a *semantic entity* as a quadruple: $\langle T, L, P, K\rangle$, where T is a timestamp or a time period, L is a location, P is a set of people, and K is a set of terms. Thus, a semantic entity contains the information about *when* and *where* an entity is evolved, *who* it is related, and a brief description about the entity given by the terms. All four elements in the quadruple can be *null* or empty, which means the attribute is unknown or not applicable.

Our quadruple-based definition of semantic entity is more general than traditional keyword-based representation of semantic entities, since it not only gives the description of the entity, but also provides the semantic information in a multi-attribute scheme. There are several ways to analyze the data based on this scheme:

1) First, semantic entities can be joined to each other based on elements in the quadruple.

2) Second, semantic entities can be projected to subspaces based on their attributes so that further aggregation analysis can be conducted.

3) Third, similarities between semantic entities can be measured.

4) Last but not the least, complicated ontology can be constructed based on this representation scheme.

4.2 Internal features for identifying entity candidates

The features that are used in annotators are listed in Table 1. A two-stage training method that combines conditional random fields (CRFs) with support vector machines (SVMs) [7] are used in annotators.

The first stage uses CRF models to identify terms that may be a naming entity. Then, the statistical features are computed. For terms or term combinations that have high *accessor variety* and *inverse document frequency*, even if they have not been seen in training corpus before, they may be naming entities that emerge. The details of using internal features to identify entity candidates is introduced in Ref. [7].

4.3 External features for filling in templates

Though internal features are proved to be useful for naming entity identification in traditional information extraction scenario, it is not sufficient in semantic entity identification. CWI focuses on data analysis on user-created content. However, user-created content are often of low-quality. For example, tweets from micro-blogs are very short, while posts from online forums are full of spoken language. Therefore, it is difficult to find semantic entities without context.

It should be noted that context includes both information from web pages we crawled and information beyond

Table 1 Features that are used by annotators

feature	type	notes
pattern	lexicon and syntax	regular expressions
begin-of and end-of	context	features of the begin- or end-of a semantic entity
mutual information	statistics	feature of the relationship of two terms that may belong to the same entity
accessor variety	statistics	feature of the relationship of two terms in terms of their independence to context
inverse document frequency	statistics	feature of the distinguishing ability of terms
dictionary lookup	external	features of terms of previous known types
template match	external	features of terms within previous known context
similarity	external	features of terms related to previous known semantic entities or entity types

the scope of crawlers. Though there are researches on converting external resources, such as wikipedia data, into local features, e.g., dictionaries, we argue that it is not a feasible approach. On one side, the volume of external resources is huge. The cost, in terms of storage, bandwidth, and time, is expensive to crawl all external resources. On the other side, since user-created content is fast evolving along with external resources, local features may not catch up with the pace of the requirements of data analysis. Thus, emerging semantic entities may be omitted if only internal and local features are used.

The authority web sites that are used as external features in different kinds of semantic entity identification are as follows:

1) *Time*: No external authority web sites are used.

2) *Location*: Google Maps (or called Google Ditu in China) is used.

3) *People*: Wikipedia and news web sites are used.

4) *Event*: Wikipedia and news web sites are used.

The terms representing naming entities are fed into search boxes of external resources. Top results are obtained and parsed. There are two strategies for feeding terms to external resources:

1) Eager: The eager strategy feeds all related terms and their combinations to external resources. After all results are collected, they are parsed together to fill in the template of semantic entities.

2) Lazy: The lazy strategy just feeds terms one by one to external features. While the template is filled in, it will not continue anymore.

Different strategies are used for different kinds of elements. For example, since locations are usually unique, the lazy strategy is used. However, for semantic entity description, i.e., terms in the quadruple, the eager strategy is used.

The results returned by external resources are represented in term vectors. Attribute values on times, locations, people names, and terms are parsed. The similarity between the vector with the information for semantic entity identification is calculated. Attribute values with highest scores in terms of similarity are used to fill in the template of semantic entities. The process is shown in pseudocode of Algorithm 1.

Algorithm 1 Filling in the template

Input: candidates of semantic entity C
Output: semantic entities S, which is initially empty
for all candidate $c : \langle T_c, L_c, P_c, K_c \rangle \in C$ **do**
  determine R_c; {R_c is the set of external resources}
  for all $r \in R_c$ **do**
    generate K_c^r; {K_c^r is the set of terms for r}
    $R \leftarrow$ query r by using K_c^r;
    $c \leftarrow$ combine c and R;
  end for
  $S \leftarrow S \cup \{c\}$;
end for

4.4 Put them all together

The overall process for semantic entity identification is shown in Fig. 2.

There are two characteristics that differentiate CWI annotators from their information extraction systems:

1) First, annotators are not built upon the fixed corpus of collected web data. Information from authority web sites are crawled, parsed, and analyzed when needed. These information are used to verify and correct the identified potential semantic entities.

2) Second, the annotation subsystem is flexible. Templates are used for pre-defined types of semantic entities. Previously identified entities are used as training data, so that features for the same type may be updated.

5 Graph storage and indexing

A subsystem named DISGR for distributed graph repository, is used for storage, indexing, and retrieval of data in CWI. It is designed specifically for user-created

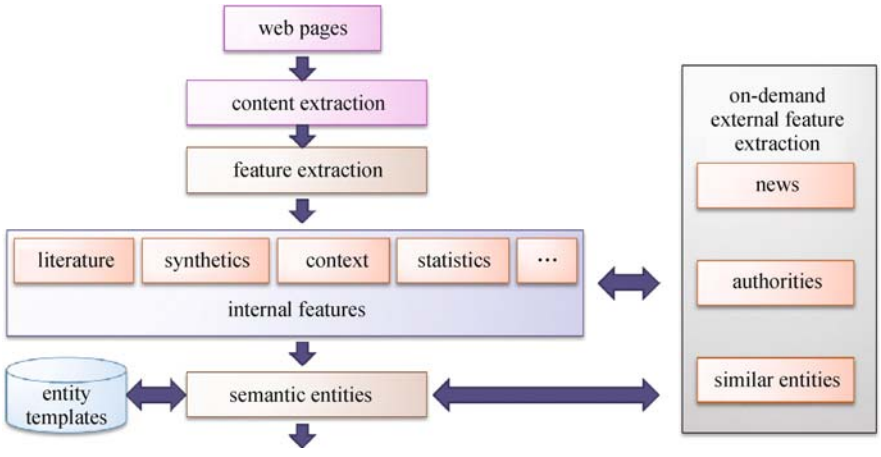


Fig. 2 Semantic entity identification

content analysis. For this purpose, DISGR is designed to be flexible for complex analytical tasks over data without fixed schema, and scalability for handling huge volume of data and large number of concurrent tasks. It has several characteristics that differentiate it from other systems, which are listed as follows:

1) DISGR is designed to manage and analyze web data. Therefore, a graph data model, TGM is used. TGM allows attributes on both vertices and directed edges. Predefined attribute data types include text, so that not only structures but also content of forum data can be represented.

2) Aiming at an infrastructure for a wide-range of users with different analytical requests, DISGR is designed to be flexible for user specific analytical processes. Users may define their own data types and procedures. Furthermore, users may also plug in their own signatures and indexing schemes for efficiency purposes.

3) DISGR is designed for deployment on shared-nothing clusters for scalability and availability purposes. All metadata, data and indices are distributed over a set of nodes with local storage. Furthermore, queries are to be evaluated in parallel.

4) TGM graphs along with attributes on vertices and edges are stored in a column-based manner. Signatures and indices on variant granularities are designed, so that unnecessary joins can be avoided in query processing.

5) Signature-based processing is used heavily in DISGR. Signatures are used for filtering unnecessary node and disk block access. They are also used for clustering and de-clustering data so that data localization can be achieved along with parallelization.

5.1 Requirements

To support variant requirements for forum data analysis, a set of characteristics is needed by DISGR. First, it should be flexible for modeling complex forum data. Forum sites have different strategies for management of posts. For example, in Google Groups (http://groups.google.com/), posts (messages) are organized in *discussions*, while each discussion belongs to a group. Groups are organized by categories. And categories are organized in hierarchies. However, in OS News (http://www.osnews.com/), posts are organized in threads, while there is no additional classification on threads. In either example above, a post may cite previous posts in post body. Also, a post may contain anything an HTML document may have. In some forums, a post may even have additional tags. Therefore, DISGR's data model should be able to represent all these kinds of data without burdening system administrators with too much management workload.

Second, DISGR should be scalable to both the volume of data and the number of analytical tasks. According to the volume of data, it is not only the concern of storage capability, but also the problem on index update, management overhead, and incremental processing for continuous analytical tasks. On the other hand, since there will exist concurrent analytical tasks, while some of them may be very time consuming, batch scheduling is apparently insufficient. Therefore, a finer scheduling strategy is needed.

Another issue is the query language. A pure descriptive language may be easy for users to learn and use, but it also limits the functionalities that users may use for complex analytical tasks. Therefore, the system should provide a descriptive language along with a flexible mechanism for plug-in user defined types and functions (UDTs and UDFs). Furthermore, for efficiency purpose, user defined indices should also be allowed to serve UDFs.

On the other hand, the query language should support on-demand queries, continuous queries, and batch processing. DISGR should apply different scheduling strategies to different kinds of queries, so that all users may have a satisfying experience.

5.2 Data model

A graph model named TGM, for Tagged Graph Model, is used to represent forum data. TGM is the basis of DISGR. TGM was first introduced in Ref. [8], and then revised in Ref. [9]. For completeness, the data model, query operators and a brief description about design choices is given here.

A TGM graph is a binary (V, E) in which V is a set of vertices and E is a set of edges. Each vertex v in V is of the form $(id, \{attr_i\})$, in which id is the identifier of the vertex, and $attr_i$'s are attributes. Each $attr_i$ is a triple $(name, type, value)$, where $name$ is the attribute name, $type$ is the data type, and $value$ is the attribute value. Each edge e in E is of the form $(id_{\mathrm{s}}, id_{\mathrm{d}}, \{attr_i\})$, where id_{s} and id_{d} are source and destination of the edge, and $attr_i$'s are attributes as those of vertices.

Figure 3 shows part of a TGM graph of a forum post from Mail Archives. It is shown that attributes can be attached on both vertices and nodes. The number of attributes on each vertex or edge may not be fixed. Thus, different properties of objects and relationships between objects can be represented.

DISGR has six predefined attribute types that are listed in Table 2. DISGR also allows users to define their own data types.

5.3 Query operators

DISGR supports five categories of query operators,

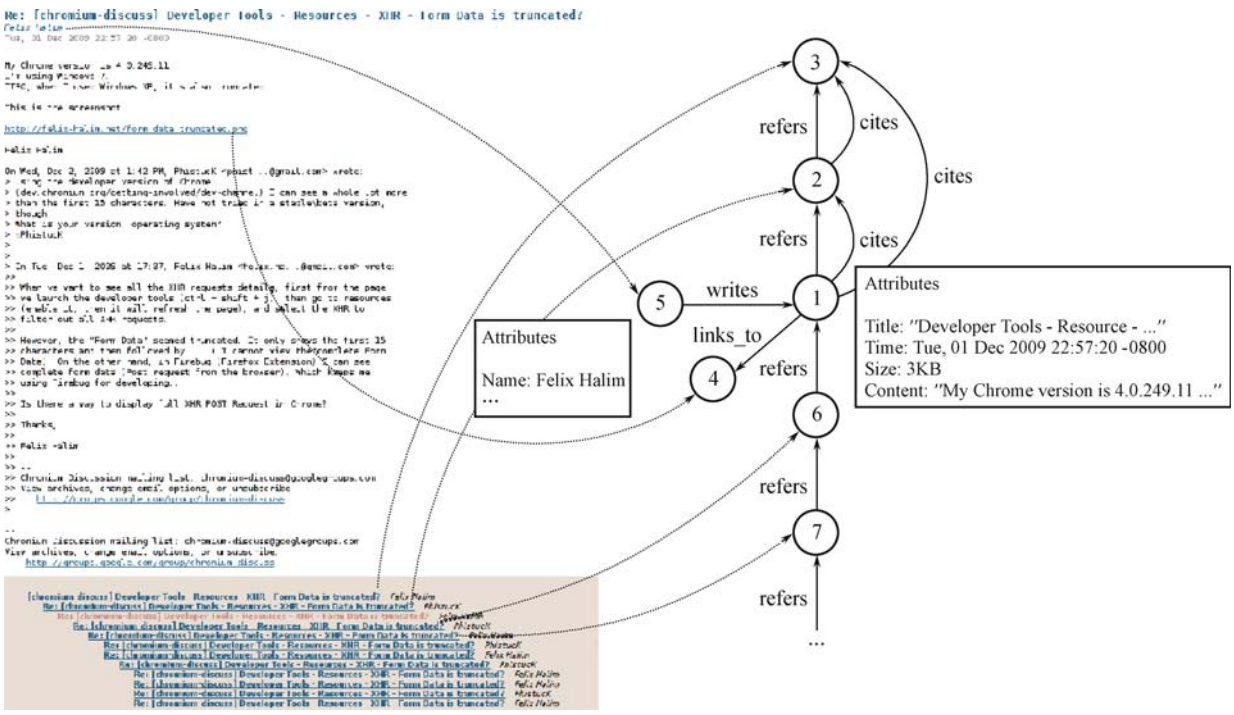


Fig. 3 Example of forum data

Table 2 Data types

data type	allowed operations	examples
numerical	comparison and arithmetic computation	*number_of_words*
categorical	equality comparison	*topic*
boolean	equality comparison and polling	*link_to*
time	comparison	*post_time*
text	keyword search and similarity search	*title, content*
undefined	equality comparison	—

which are introduced as follows:

1) *Wrappers.* A wrapper transforms a given TGM graph $G:(V,E)$ into another one $G':(V',E')$ based on certain rules. For example, given a rule set R with two rules:

- If $v \in V$, then $v \in V'$;
- For v_1, v_2, and $v_3 \in V$, if both edges (v_1, v_2) and $(v_2, v_3) \in E$, then $(v_1, v_2), (v_2, v_3)$ and $(v_1, v_3) \in E'$.

Then, the output of $wrapper_R(G)$ is the graph with the same vertices as G, while indirect linked vertices are connected with new edges with null attributes.

Note that in each rule, body can only contain conditions on vertices and edges in G, while heads can only contain propositions on those in G'. Therefore, there is no recursion in rules.

Rules are a kind of plugins in DISGR. Users may define their own rules, so that wrappers can be extended.

2) *Filters.* A filter $filter_{\text{cond}}(G)$ filters vertices and nodes in G that does not satisfy the condition *cond* out, and forms a new graph G'. There are two types of conditions, *inner conditions* and *outer conditions*. Both are of the form *attr_name op value*. The semantics of a filter is that, given a graph G, first, all vertices and edges satisfying the inner conditions are kept as a new graph G_0. Then. those vertices and edges that are directly connected to G_0 and satisfying the outer conditions are added to form G', which is the output of the filter operator.

3) *Merges.* A merge operator takes two graphs $G_1:(V_1,E_1)$ and $G_2:(V_2,E_2)$ and generates a new graph G. There are two kinds of merge operators, i.e., *union* and *join*.

Union is simple in that it just generates a new graph whose vertex and edge sets are unions of those from input graphs, correspondingly, i.e., $union(G_1,G_2) = G:(V_1 \cup V_2, E_1 \cup E_2)$. Join is actually the combination of a union operator with a wrapper.

4) *Groupings.* A grouping operator takes one graph $G:(V,E)$ and generates a set of subgraphs $\{G_i\}$. For each subgragh $G_i:(V_i,E_i)$, $V_i \subseteq V$, and $E_i \subseteq E$. Grouping is conducted based on attributes on vertices or edges, i.e., vertices or edges with common attribute value are assigned to the same subgraph.

5) *Aggregations.* Aggregations are conducted on a graph or a set of graphs (i.e., the result of a grouping operator). Each aggregation generates a set of statistics,

that can be represented as a new vertex's attributes. Typical aggregations including count (on number of vertices, number of edges, number of distinct value of a specific attribute, etc.), average (on value of a specific attribute for example), and sum (on value of a specific attribute, for example).

5.4 Implementation

Some implementation design of DISGR are introduced as follows:

1) *Column-based distributed storage*. Graphs, along with attributes, are stored in a column-store manner, which means:

• Each vertex is assigned a unique identifier;

• Each edge is assigned a unique identifier, and is stored in the form of (eid, vid', vid'');

• Each attribute value of a vertex is stored in the form of $(vid, attr\_name, value)$;

• Each attribute value of an edge is stored in the form of $(eid, attr\_name, value)$.

Currently, all the above four kinds of information are stored in separate files in HDFS of Hadoop.

2) *Multi-level index*. Indices are constructed in three levels:

• *Block index*. Data are not organized in rows, which is the default option in Hadoop. Within each block, there is a *block head* and a *block extent*. The block extent contains all data, while the block head contains the index of the data. This scheme reduces the cost of scanning in traditional Map/Reduce processing.

• *Node index*. Data are distributed to blocks based on their content by using hashing. Thus, for a specific query, only part of the blocks on part of nodes are accessed. Thus, this hash scheme is used as node index.

• *Global index*. Not all vertices are stored in the same file, neither are edges, or attribute values. Each of them are partitioned into several files, based on some predefined conditions, such as their original source (web sites), or original graph identifier (for generated graphs of some operators). This scheme greatly reduces the cost for joins.

6 Related work

TGM is similar to WebSQL [10] and WebOQL [11] in that TGM accepts high-level descriptive queries as description on required services. The difference is that in TGM, attributes are used as query conditions.

Traditional centralized storage is not scalable enough for storing web data with huge volume. Historically, parallel database has been intensively studied [12,13]. Recently, large-scale distributed storage systems based on shared-nothing clusters are often chosen as a solution for this kind of tasks. Google file system (GFS) [14] is a system that uses centralized file lookup yet decentralized data access. GFS is designed for applications with huge amount of parallel access to large files, such as search engines. Also, it is reported to be very efficient and scalable [14]. DISGR focuses on the problem of how to take advantage of such kind of distributed storage to manage graph data. MapReduce [3] is a programming model for data intensive computing in large-scale distributed environments. It splits a computing task into two phases: the map phase and the reduce phase. In each phase, data are only processed locally within each node, so that the cost of data transmitting can be saved. Dryad [15] is similar to GFS/MapReduce except that it is developed based on Microsoft Windows. Web data management, analysis and mining tools, such as BigTable [16], Sawzall [17], Pig Latin [18,19], and WebStudio [20], are developed based on GFS, MapReduce or Dryad.

Graph data management is an important issue and is highly related to web data management. Existing research about graph data management focuses on structural queries and structure indexing [21–24]. However, in DISGR, we focus on how to provide queries on both structure and content, and scalability issues.

7 Discussion

CWI is a prototype system that is under development. This paper introduces the current design and implementation of the system. We believe that the best-effort approach taken by CWI is a practical one towards a usable infrastructure, while leaving space for advanced technology integration. Some research issues that we are working on include:

1) Rule learning for structure information extraction

Currently, the rules for extracting basic information from the crawled data are written by users directly. However, it is not trivial on writing such rules. We are studying learning methods for semi-automatically obtaining such rules.

2) Candidate and authority selection

In current implementation, strategies for selecting external resources are fixed, while strategies for feed terms to external web sites are simple. We are working on more intelligent candidate selection and authority selection policy.

3) Hash-based graph data distribution

Currently, vertices, edges, and attribute values are distributed to blocks by hashing just based on *values* or *neighboring vertices*. Though this approach filters out some unnecessary data access, it is far from efficient. We are working on hash-based structural indexing method that may improve the performance.

Acknowledgements This work was partially supported by the National Natural Science Foundation of China (Grant Nos. 60833003 and 61070051), and the National Basic Research Program of China (Grant No. 2010CB731402).

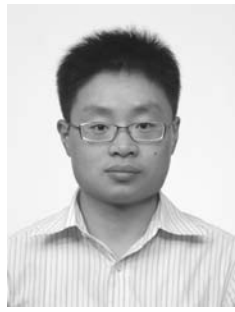
References

1. Qian W, Zhou A. Chinese Web infrastructure building: challenges and our roadmap. In: Proceedings of International Workshop on Information-Explosion and Next Generation Search. 2008, 4–11
2. China Internet Network Information Center. The 24th Statistical Report on the Development of the Chinese Internet. CNNIC, 2009
3. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. In: Proceedings of Operating Systems Design and Implementation. 2004, 137–150
4. Clark J, DeRose S. XML Path language (XPath) version 1.0. World Wide Web Consortium Recommendation, 1999. http://www.w3.org/TR/xpath.html
5. Clark J. XSL Transformations (XSLT). World Wide Web Consortium Recommendation, 1999. http://www.w3.org/TR/xslt
6. Sarawagi S. Information extraction. Foundations and Trends in Databases, 2008, 1(3): 261–377
7. Cai P, Luo H, Zhou A. Semantic entity detection by integrating CRF and SVM. In: Proceedings of the 11th International Conference on Web-Age Information Management. Lecture Notes in Computer Science, 2010, 6184: 483–494
8. Zhou A, Qian W, Tao D, Ma Q. DISG: a distributed graph repository for web infrastructure. In: Proceedings of the Second International Symposium on Universal Communication. 2008, 141–145
9. Qian W. Storage and index support for data intensive web applications. In: Proceedings of the 4th International Universal Communication Symposium. 2010, 62–68
10. Arocena G O, Mendelzon A O, Mihaila G A. Applications of a web query language. Computer Networks, 1997, 29(8–13): 1305–1315
11. Arocena G O, Mendelzon A O. WebOQL: restructuring documents, databases, and webs. In: Proceedings of the 14th International Conference on Data Engineering. 1998, 24–33
12. DeWitt D, Gray J. Parallel database systems: the future of high performance database systems. Communications of the ACM, 1992, 35(6): 85–98
13. Li J, Gao H, Luo J, Shi S, Zhang W. InfiniteDB: a PC-cluster based parallel massive database management system. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. 2007, 899–909
14. Ghemawat S, Gobioff H, Leung S T. The Google file system. In: Proceedings of the 9th ACM Symposium on Operating Systems Principles. 2003, 29–43
15. Isard M, Budiu M, Yu Y, Birrell A, Fetterly D. Dryad: distributed data-parallel programs from sequential building blocks. In: Proceedings of European Conference on Computer Systems. 2007, 59–72
16. Chang F, Dean J, Ghemawat S, Hsieh W C, Wallach D A, Burrows M, Chandra T, Fikes A, Gruber R E. Bigtable: a distributed storage system for structured data. ACM Transactions on Computer Systems, 2008, 26(2): 1–26
17. Pike R, Dorward S, Griesemer R, Quinlan S. Interpreting the data: parallel analysis with Sawzall. Scientific Programming, 2005, 13(4): 277–298
18. Olston C, Reed B, Srivastava U, Kumar R, Tomkins A. Pig Latin: a not-so-foreign language for data processing. In: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data. 2008, 1099–1110
19. Gates A, Natkovich O, Chopra S, Kamath P, Narayanam S, Olston C, Reed B, Srinivasan S, Srivastava U. Building a highlevel dataflow system on top of mapreduce: the Pig experience. Proceedings of the VLDB Endowment, 2009, 2(2): 1414–1425
20. Wen J R, Ma W Y. Webstudio: building infrastructure for web data management. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data. 2007, 875–876
21. Mendelzon A O, Wood P T. Finding regular simple paths in graph databases. SIAM Journal on Computing, 1995, 24(6): 1235–1258
22. Cheng J, Ke Y, Ng W. Efficient query processing on graph databases. ACM Transactions on Database Systems, 2009, 34(1): 1–48
23. Qun C, Lim A, Ong K W. D(k)-index: an adaptive structural summary for graph-structured data. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data. 2003, 134–144
24. Yan Y, Wang C, Zhou A, Qian W, Ma L, Pan Y. Efficient indices using graph partitioning in RDF triple stores. In: Proceedings of the 25th International Conference on Data Engineering. 2009, 1263–1266

Weining QIAN is currently an associate professor of Institute of Massive Computing, Software Engineering Institute, at East China Normal University, Shanghai, China. Before this he was an assistant professor at Department of Computer Science and Engineering at Fudan University. He received his M.S. and Ph.D. degrees in computer science from Fudan University in 2001 and 2004. His research interests include web data management, data mining for very large databases, data stream query processing and mining and peer-to-peer computing.

Aoying ZHOU is currently a professor in Computer Science at East China Normal University, Shanghai, China, where he is also chairing the Institute of Massive Computing. He received his Ph.D. degree from Fudan University in 1993, his master

and bachelor degrees from Sichuan University, Chengdu, in 1988 and 1985, respectively. He is now serving as the vice-director of ACM SIGMOD China and Database Technology Committee of China Computer Federation. He was invited to join the editorial boards of some prestigious academic journals, such as VLDB Journal, Journal of Computer Science and Technology (JCST), etc. He served or is serving as PC members of ACM SIGMOD07/08, WWW07/08, SIGIR07/08, EDBT06, VLDB05, ICDCS05, etc. He was the conference co-chair of ER'2004 and the PC co-chair of WAIM'2000.



Minqi ZHOU is currently a lecturer at Institute of Massive Computing, Software Engineering Institute, at East China Normal University, Shanghai, China. He received his Ph.D. degree in computer science from Fudan University in 2009. His research interests include massive data management, peer-to-peer computing and information retrieval.