TELKOMNIKA Indonesian Journal of Electrical Engineering
Vol.12, No.3, March 2014, pp. 2271 ~ 2277
DOI: http://dx.doi.org/10.11591/telkomnika.v12i3.4497

A Real-time Search Structure and Classification Algorithm of Microblog Based on Partial Indexing

Jingbo Yuan*[1], Bairong Wang[2], Shunli Ding[3]
Northeastern University at Qinhuangdao, China
School of Business Administration Northeastern University, China
Environmental Management College of China, China
*Corresponding author, e-mail: jingboyuan@hotmail.com[1], nihaoalison@163.com[2], dingsl@163.com[3]

Abstract

With the popularity and the rapid development of social networks, microblog has also obtained great attention and gotten wide application. Microblog may produce a large amount of information per second. The real-time search is a critical technology to timely get the latest high-quality information from microblog. A new real-time search framework based on meta-search engine and a query classification algorithm of microblog was put forward. The classification and indexing process is based on the partial indexing mechanism to improve retrieval efficiency. The classification algorithm divides queries into the candidate queries and the popular queries, and takes separate storage strategy. Test results show that the classification algorithm can reduce real-time search time and improve the efficiency of retrieval.

Keywords: *social networks, microblog, real-time search, classification algorithm*

1. Introduction

As a new media platform, microblog has obtained great attention in recent years. In addition to information sharing and exchange, microblog also derives other applications, such as micro-blog marketing. Applications of microblog need to implement information sharing with the technology support of real-time search. Real-time search is a kind of real-time and quick search on the Internet, to get the latest information the moment you hand in the query.

Due to the high efficiency and large amount of latest information, microblog is destined to be the main information source of real-time search. At present, the real-time web search represented by Twitter has become one of the most popular applications in the Internet [1, 2]. Microblog produce a large amount of information every minute, and the information is disorder and short life cycle [3]. According to the statistics, the number of Twitters sent by users every day is as high as 130 million. How to get the latest and high-quality information from this huge, messy information has become a major challenge of the real-time search technology, which had been researched and received great attention from academia [4-7].

Providing real-time search service is indeed very challenge for large-scale microblog systems[8]. In order to guarantee good quality and high efficiency, information identification and index should be carried out as quickly as possible. The web crawler technology for traditional search engine cannot meet the demand of real-time retrieval and its structure is not suitable for real-time search based on microblog.

In the paper, a new framework for microblog real-time search has been designed on the foundation of meta-search engine, taking the retrieval mechanism of a microblog system itself as information filtering interface to save the cost of information filtering of the real-time search system and improve the microblog's information quality. The microblog classification process is based on the partial indexing mechanism. At the same time, a microblog classification algorithm is proposed, and take stored separately strategy from the popular questions and the candidate question. The algorithm reduces real-time search time and improves the efficiency of retrieval.

2. The Structure of Microblog Real-time Search

Hundreds million microblog users use the service every day and produce a great deal of information. To ensure the system’s efficiency, the real-time search can be modeled as meta

Received August 24, 2013; Revised September 28, 2013; Accepted October 21, 2013

search engines and establish the search service on the basis of various microblog search, in this way it can avoid filtering and indexing all the microblogs. The real-time search system of microblogs will use the retrieve ranking mechanism and only store those microblogs higher ranked by the ranking function. The structure framework of microblog real-time search besed on meta-search engine is designed in this paper, as shown in Figure 1.

Request submitting agent is responsible for implementing the users' personalized searching demands, such as calling which microblog search engines, setting the limits of searching time and the number of results, etc. Moreover, it is in charge of distributing the requests to independent microblog systems.

Cooperation websites are mainly the microblog sites, such as Sina microblog, Tencent microblog, etc. Of course, the real-time search can also adopt other websites' information in order to enrich the sources of information. The cooperation websites provide the real-time system with latest data and improve the instantaneity of real-time search.

Data formats submitted by each cooperation website are very different, therefore, these data will be cut and unified and the useful information will be extracted afterwards in the converter. All the work is prepared for ranking calculation.

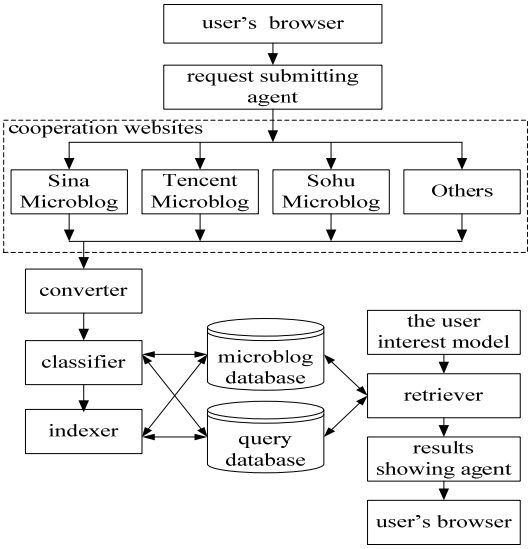

Figure1. The Structure of the Real-time Search

The partial indexing mechanism has to be established on the classification of query and microblog. Partial indexing, namely classified index, is also used to improve retrieval efficiency of the system.The classifier is mainly responsible for classifying microblog. Meanwhile, it can also call the data in the database and conduct updating as well as maintenance constantly.

The indexer mainly adopts different indexing strategy according to the results of the classifier. At the same time, the indexer will update the database regularly, clear out the data more than a certain time range in order to keep the system running at high speed.

The retriever can search the database according to the customers' demands and output the qualified results. In addition, the retriever will learn automatically users' interest models in accordance to the statistics. In this way it can provide users with real-time search suggestions and help users find useful information more quickly.

The results showing agent will classify the results and push the popular information to users, and the results will be shown on the homepage and updated continually.

3. Classification and Indexing Process

The idea of partial indexing emerged as early as the 19th century. In short, partial indexing will not index the whole database, instead it just index microblogs that are very likely to

be the results of a certain query. In this paper the query is divided into hot query and candidate query as shown in definition 1 and 2, and microblog is divided into effective microblog and noisy microblog respectively. Queries and microblogs are both composed of keywords.

Definition 1: the candidate query. For microblog $t = \{k1, k2, \ldots, kn\}$ and query $q = \{k1', k2', \ldots km'\}$, if there is $ki \in t \rightarrow kj' \in q \wedge kj' = ki$, then the query can be called candidate query, and the microblog is called effective microblog.

Definition 2: the popular queries. For any candidate query q, if the number of its effective microblog $Cq >= M$, then q is a popular query. M is threshold number and $M=C*R$, where, R is the number of records of each page of the search results. $C=5$, and the systems can also make adjustments factor C as needed.

For the effective microblogs of hot queries, the system will do real-time indexing and for the noisy ones based on the partial indexing batch indexing with offline mode. All microblogs of candidate queries are with batch indexing so as to improve the system response time and efficiency. Partial indexing mechanism should take query and microblog classification as foundation. Therefore, classification should be done before partial indexing, the whole process is mainly divided into the following six steps, and its work flow is shown in Figure 2.

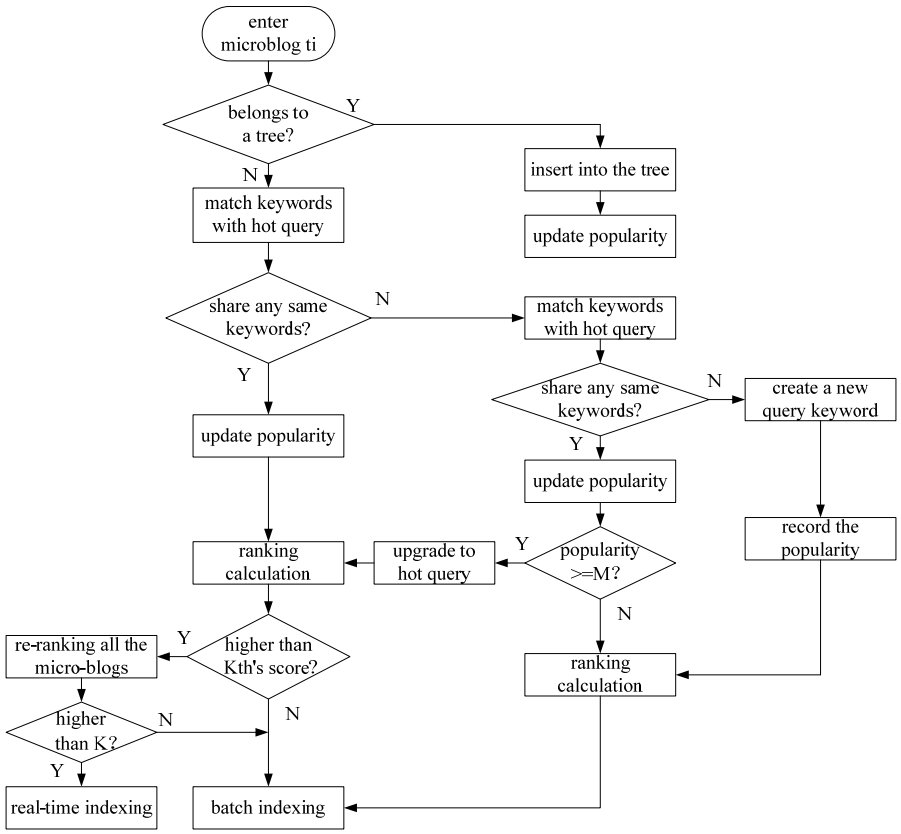

Figure 2. The Microblog Classification and Indexing Process

Step 1: Once a microblog *ti* entered, the system will first judge the type of *ti*. If the microblog belongs to a tree structure, then it will be stored in a tree structure with an increase of the tree's popularity afterwards. If it is a single node structure, then go to Step 2.

Step 2: The system will match the keywords of ti with hot queries, if it does not contain the keywords of any hot query, then enter the Step 3. If it contains the keywords of a hot query, popularity of this query will be updated and go to Step 5.

Step 3: The system will match the keywords of ti with candidate queries, if it does not contain candidate queries' keywords, then a new query keyword will be created. Meanwhile, the

system will record the query's popularity as well as the microblogs' ranking results, and conduct batch indexing on *ti*. If it contains the keywords of this candidate query, then go to Step 4.

Step 4: The system will updated the popularity of this candidate query and make a comparison of popularity *Cq* with system threshold M. If *Cq* < *M*, the query is still a candidate one, the system will conduct ranking calculation and batch indexing afterwards. If *Cq>M*, this candidate query will upgrade for hot query and go to Step 5.

Step 5: The system will calculate *ti's* ranking. If *score(ti)<score(tk)*, then *ti* is noisy and batch indexed. If the score is higher than that of the *Kth* microblog in the query's set, namely score *(ti)* ≧*score (tk)*, then *ti* belongs to effective ones and go to Step 6.

Step 6: The system will re-rank all effective microblogs belong to this hot query, if *ti* rank higher than *K*, then do real-time indexing, or batch indexing.

4. Microblog Classification Algorithm

Queries of the search engine follow the Pareto law, that is, 20% of the queries represent 80% of the users' demand. Therefore we store the data separately, namely 20% of the queries are recorded in memory, and the remaining 80% of the queries are recorded in logs of the hard disk. After a query is classified, the address of its related microblog will be stored, in this way some microblogs are classified with the classification of queries. May believe that microblogs related with popular queries are useful and effective ones. In order to keep a real-time performance, classification results of queries and microblog will be updated constantly.

There are two keywords lists, which set the microblogs and queries into vectors. Table 1 and Table 2 illustrate the matrix structures of microblogs and queries. *Ak* is an *m* by *n* matrix, where *m* represents the number of keywords in the system, and *n* represents the number of microblog. *Bk* is a *m* by *u* matrix, where *m* represents the number of keywords in the system, and *u* represents the number of queries. Each line of the matrix represents a microblog or a query, and each column represents a keyword. If the *jth* keyword appears in the *nth* microblog or the case of the *ith* query, then set the value of *Ak[n][j]* or *Bk[i][j]* to 1, otherwise, set to 0.

Table 1. Keywords Table of Microblogs *Ak*

	k1	k2	k3	k4	……	Km
t1	1	0	0	0	……	0
t2	1	0	1	0	……	1
t3	0	1	0	0	……	0
t4	1	0	0	0	……	0
……	……	……	……	……	……	……
tn	0	0	0	0		1

Table 2. Keywords Table of Queries *Bk*

	k1	k2	k3	k4	……	km
q1	1	0	0	0	……	0
q2	1	0	1	0	……	1
q3	0	1	0	0	……	0
q4	1	0	0	0	……	0
……	……	……	……	……	……	……
Qu	0	0	0	0		1

To improve working efficiency, it need to batch the calculation process. The evaluation vector on microblog and queries is expressed as Ve and calculated according to the Equation (1).

$$V_e = A_k \times B_k^T \quad (1)$$

Bk^T is the transposed matrix of *Bk*. Ve is an *i×i* matrix. *Ve[i][j]* present the number of same key words in microblog *ti* and query *qj*. If *Ve[i][j] ≠0*, then *qj* is a candidate query, and *ti* is an effective microblog to query *qj*. In addition, need a count vector *Cq* to record the number of effective microblogs to each query, and *Cq* is expressed as Equation (2).

$$Cq[j] = \sum_{j=1}^{n} Ve[i][j] \quad (2)$$

By applying the matrix vector model, can convert the process of looking for candidate queries into the process of matrix calculation. The evaluation vector can be calculated rapidly because both *Ak* and *Bk* are sparse matrixes.

It has been reported that 62% of the users will click on the first page of search results, and more than 90% of the users will not browse pages after the third page. So only part of the effective microblogs to each query are stored. The microblog screening function adopted in the process of query classification is shown as Equation (3).

$$E(q_j, t_i) = \alpha * t_i.\text{timrstpam} + \beta * Ve[i][j] \quad (3)$$

Where *ti.timestamp* is the submit time of microblog *ti*. *Ve[i][j]* is the number of same keywords in microblog *i* and query *j*. *α* and *β* are used to control the weight of different factors. This fuction is just used when there are many effective microblog of a query at the time that the query is initially established. When the query is stable, another classification function will be used.

The query classification algorithm contains three major parts. The first is to get *Ve*. The algorithm for getting *Ve* is shown as follows:

```
1  int[ ][ ] getVe( )
2  {
3            for(int i=0;i<n;i++)
4       {
5                  for(int j=0;j<u;j++)
6            {
7                     int sum = 0;
8                     for(int k=0;k<m;k++)
9               {
10                          sum+=ak[i][k]*bk[j][k];
11              }
12                    ve[i][j]=sum;
13               }
14         }
15         return ve;
16       }
```

Where *i* is the subscript of microblogs *t*, and *n* is the number of microblogs calculated, and *j* is the subscript of query *q*, and *u* is the number of queries calculated and *k* is the subscript of keywords *t*, and *m* is the number of keywords in the search engine. Three loop calculations are used in this algorithm. The first one (line 3) is to ensure that all microblog are calculated, the second one (line 5) is to make the microblog is caclulated with every query, and the third one (line 8) is to calculate every key word of the query. The value of sum is the product of microblog ti and query qj with every keyword (line 10).

The second is to get *Cq*. The algorithm for getting *Cq* is shown as follows:

```
1  int[ ] getCq( )
2  {
3           for(int j=0;j<u;j++)
4       {
5                  int sum = 0;
6                  for(int i=0;i<n;i++)
7            {
8                    sum+=ve[i][j];
9                   }
10                 Cq[j] = sum;
11         }
12         return Cq;
13  }
```

From query *1* to *u*, calculate *Cq* (line 1), the process is from microblog *1* to *n* (line 4-7), then get the *Cq* from the sum of *ve[i][j]* (line 8).

The third is the algorithm for query classification and shown as follows:

```
1  CategoryQuestion( )
2  {
3     int[] Cq = getCq();
4     if(Cq[j]==0)
5      {
6            return new query;
7           }
8      else if(Cq[j]>=M)
9       {
10                  return hot query");
11      }
12       else
13        {
14                  return candidate query
15        }
16   }
```

If *Cq[j]=0*, it means that a microblog has not the same keyword with the query, then the query is totally new (line 4-7). If *Cq[j]>=M*, it means that the query has more than *M* effective microblog, it can be the popular query (line 8-11), in the other situations, the query will be an candidate query.

5. Verifying of the Classification Algorithm

In the validation phase, grab a total of 43769 microblog published at different time from Sina microblog. Before testing, the value of users' authorities has been calculated off-line. A keyword database has been built based on the keywords from these microblog. The searching process with 130000 queries is simulated.

The processing time of popular queries are also compared under the two store policy respectively, the one is stored together and the other is stored separately. This kind of comparisons is also performed on candidate queries and the whole queries. The comparing results for popular and candidate queries are shown in Figure 3 and Figure 4. It can be seen form figures that processing time with separate storage is far less than queries stored together because the popular queries are searched often and the processing time in memory is much less than that on hard disk, the separate storage can reduce I/O time. However the processing time of candidate queries is not very different between the two policies, mainly because the candidate queries are searched quite often and the separate storage policy can not reduce too many I/Os. And then the test data show that nearly 78% of users can match their queries with popular queries in memory, and nearly 17% of the queries need to be processed in the candidate queries set.

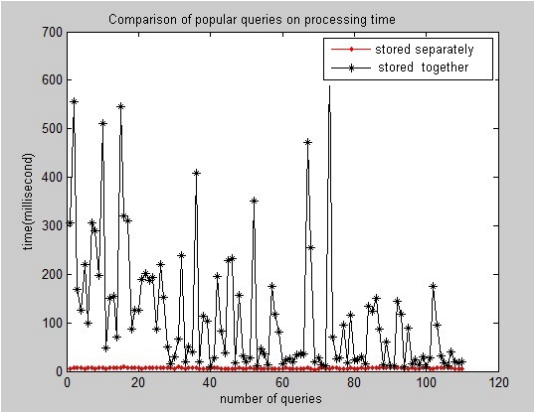

Figure 3. Comparison of Popular Queries on Processing Time

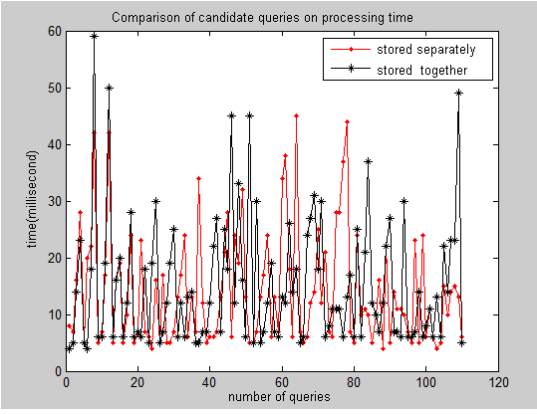

Figure 4. Comparison of Candidate Queries on Processing Time

6. Conclusion

As a new media platform, microbolg has a dramatically increasing demand of real-time information. Based on the meta-search engine and Sina microblog as for example, a framework of microblog real-time search is designed, and a microblog classification algorithm based the partial indexing mechanism is put forward. Taking cooperation microblog systems as the information filtering interface, the framework not only saves the cost of real-time search, but also gets microblog data with high quality.In the way based on query-classification, come up with the separate storage strategy to improve the real-time search efficiency. The test results show that the query-classification and separate storage strategy can improve the efficiency of information retrieval, especially for popular questions queries.

References

[1] Owen Phelan, K McCarthy, B Smyth. *Using twitter to recommend real-time topical news*. Proceedings of the third ACM conference on Recommender systems. New York. 2009: 385-388.

[2] John Hannon, Mike Bennett, Barry Smyth. *Recommending twitter users to follow using content and collaborative filtering approaches.* Proceedings of the 4th ACM Conference on Recommender Systems. Barcelona. 2010: 199-206.

[3] Li Yungming, Hsiao Hanwen. *Recommender service for social network based application*. Proceedings of the 11th International Conference on Electronic Commerce. New York. 2009: 378-381.

[4] Ostermaier B, Romer K, Mattern F, et al. *A real-time search engine for the web of things.* Internet of Things (IOT). Tokyo. 2010: 1-8.

[5] Jansen BJ, Liu Z, Weaver C, et al. Real time search on the web: Queries, topics, and economic value. *Information Processing & Management*. 2011; 47(4): 491-506.

[6] Soboroff I, Mccullough D, Lin J, et al. *Evaluating real-time search over tweets*. Proceedings of the 6th International AAAI Conference on Weblogs and social media (ICWSM). Dublin. 2012: 943-961.

[7] Efron M. Information search and retrieval in microblogs. *Journal of the American Society for Information Science and Technology*. 2011; 62(6): 996-1008.

[8] Chun Chen, Feng Li, Beng Chin Ooi, Sai Wu. *TI: an efficient indexing mechanism for real-time search on tweets*. SIGMOD'11. Athens. 2011: 649-660.