 CMC, vol.59, no.1, pp.229-237, 2019

PMS-Sorting: A New Sorting Algorithm Based on Similarity

Hongbin Wang[1], Lianke Zhou[1], Guodong Zhao [1, *], Nianbin Wang[1], Jianguo Sun[1], Yue Zheng[2] and Lei Chen[3]

Abstract: Borda sorting algorithm is a kind of improvement algorithm based on weighted position sorting algorithm, it is mainly suitable for the high duplication of search results, for the independent search results, the effect is not very good and the computing method of relative score in Borda sorting algorithm is according to the rule of the linear regressive, but position relationship cannot fully represent the correlation changes. aimed at this drawback, the new sorting algorithm is proposed in this paper, named PMS-Sorting algorithm, firstly the position score of the returned results is standardized processing, and the similarity retrieval word string with the query results is combined into the algorithm, the similarity calculation method is also improved, through the experiment, the improved algorithm is superior to traditional sorting algorithm.

Keywords: Meta search engine, result sorting, query similarity, Borda sorting algorithm, position relationship.

1 Introduction

Meta search engine [Zhang, Yu, Liao et al. (2004); Smyth and Boydell (2010)] is aimed to increase the precision and recall rate of in-dependent search engine, so there is no need to set data base for search and retrieval mechanism [Yamamoto, Fujii, Toyofuku et al. (2001); Yang (2005); Chen and Xu (2016)]. It achieve its search behavior by integrating a search engine that best meets user needs in accordance with the users' interest or excel-lent degree of search engines, and its search interface is the same as the tradition-al search engines. For the search results returned, the Meta search engine will integrate a mechanism in accordance with results to remove-duplicate web pages and complete mix, then sort them according to a certain algorithm, finally return to user a process. Therefore, the ranking of results for the Meta search engine is of vital importance. There are a lot of researches for algorithm of ranking results nowadays. In this paper, the author studies and makes improvement on the classic Borda ranking algorithm. As a traditional ranking algorithm in a weighed way, Borda ranking algorithm is first applied in vote [Yang (2005); Yong and Zulin (2011)], which is a method for voters to choose candidates.

[1] College of Computer Science and Technology, Harbin Engineering University, Harbin, 150001, China.

[2] Liren College of Yanshan University, Yanshan University, Qinhuangdao, 066004, China.

[3] College of Engineering and Computing, Georgia Southern University, Georgia, 30458, USA.

[*] Corresponding Author: Guodong Zhao. Email: zhaoguodong@hrbeu.edu.cn.

Because of its good availability, it has been widely used.

2 Traditional borda sorting algorithm model

The traditional Borda algorithm [Cho, Brand, Bordawekar et al. (2015)] is a kind of improvement based on the weighted sorting algorithm. The algorithm is described as follows:

We define the set of tested search engines in the meta search engine as $S = \{s_1, s_2, \ldots, s_n\}$. $R = \{r_1, r_2, \ldots, r_m\}$ is the set for all query results to query word q. Each query result r_k is composed of four parts: URL, title, abstract, relevance score, which are represented by array assi_Url[k]，si_Title[k]，si_Abs[k]，si_Score[k] on condition of $k = 1,2,\ldots,m$, $i = 1,2,\ldots,n$.

The Borda sorting algorithm [Katsirelos, Walsh, Davies et al. (2011)] in meta search engine [Lawrence and Giles (1998)] is voted by the results returned by tested search engines. It establishes the preference relationship in position relationship among tested search engines according to the returned results after inputting query word [García-Lapresta and Martínez-Panero (2002)]. If the result is independent, we regard related score in other search engines as zero. Finally we put all the scores of each result to be summed to obtain the final score, and sort it in descending order. The mathematical model for the algorithm is as follows: the number of tested search engine is n, which means $S = \{s_1, s_2, \ldots, s_n\}$; the set of query results is $R = \{r_1, r_2, \ldots, r_m\}$. For S_k, we build matrix of preference relationship to R_k as Eq. (1).

$$R_k = \begin{bmatrix} b_{11}^k & b_{11}^k & \ldots & b_{1m}^k \\ b_{21}^k & b_{22}^k & \ldots & b_{2m}^k \\ \ldots & \ldots & \ldots & \ldots \\ b_{m1}^k & b_{m1}^k & \ldots & b_{mm}^k \end{bmatrix} \tag{1}$$

when r_i is ranked before r_j by k, $b_{ij}^k = 1$; otherwise it is 0. The score for ri from Sk is shown in Eq. (2).

$$r_i^k = \sum_{j=1}^{m} r_{ij}^k \tag{2}$$

so the matrix of all query results from R_k (k=1,2,…,n) is shown in Eq. (3).

$$R_k = \begin{bmatrix} r_1^1 & b_1^2 & \cdots & b_1^n \\ r_2^1 & r_2^2 & \cdots & r_2^n \\ \vdots & \vdots & & \vdots \\ r_m^1 & r_m^2 & \cdots & r_m^n \end{bmatrix} \tag{3}$$

the final score is shown in Eq. (4).

$$Borda(r_i) = \sum_{k=1}^{m} r_i^k \tag{4}$$

sorting $Borda(r_i)$ according to relevance score of Borda and return to users.

3 PMS-sorting

The PMS-sorting algorithm based on the query result position, multiplicity and query similarity as is proposed in this paper, not only considers the relevancy and repeated read information of the query result position, but also combines the similarity of the query term and query result, which improves considerably the result of independent searching [Ping (2003)].

Besides, we plan to use global similarity for computing the similarity between query term and query result, because the methods of relevance algorithm of each independent search engine are not public, or are to be compared directly. In addition, on the problem of malpractice of the current global relevance algorithm, the similarity algorithm on the titles and abstracts of the return result would be more effective and accurate.

The research in the paper is about one user's query string q, the score of the query result can be ultimately represented as a Borda score, and can be sorted and showed to users. We are going to make discussions on the algorithms in the following part.

3.1 The position standardization of the query result in search engine

Result list of independent search engines is sorted according to relevance of search words, therefore, the position of results can reflect its relevance with the query word enormously. Under common cases, the first ones of the results are most relevant to the users, making it very necessary to consider the position information of independent search engine. To make the position score more accurate, we improved the algorithm as follows.

N search engine members $S_1, S_2, \ldots, S_n$ search a certain query word q, and m results are returned by search engine S_j, and the relevance of the result r_k located in k and the users query position is represented by $pos(q, S_j, r_k)$, which is shown in Eq. (5).

$$pos\left(q, S_j, r_k\right) = \frac{m-k+1}{m} \tag{5}$$

where $pos(q, S_j, r_k) \in [0,1]$, if the query result rk is the first result in the result collection of some search engine, then the score of $pos(q, S_j, r_k)$ is 1, which means that the first result of all member search engines are equally important. But if the numbers of results list documents that returned are different, the smaller is the number, the higher is the score. It means that having a good position in a list that has more results is more valuable than in a list that has fewer results. Thus the relationship between query results and query words is unified, i.e., the latter the position is, the smaller $pos(q, S_j, r_k)$ is, the fewer its relationship with the query word is, and the less the influence on the sorting.

3.2 The global similarity between query results and user query

Suppose query string q has n feature items $t_1, t_2, \ldots, t_n$ and two documents d_1, d_2, if a certain feature occurred n times in d_1 while other feature items haven't occurred, but in d_2, n feature items have occurred once, in this case, although the word frequency is the same in the two documents, d_2 is obviously more relevant and has the most

comprehensive covered features. For example, the query string "central People's Government" divides q into three feature items, $t_1 =$ "Central", $t_2 =$ "People's", $t_3 =$ "Government" , if feature item $t_1 =$ "Central" occurred many times in document one with other features not occurring, while all the three features occurred once in document two, apparently document two is more relevant to query string q. Under such cases, higher weight should be put when the query string matches the query word more comprehensively.

The matching degree between query string q and the r_k abstract. If the feature term matches the abstract rather comprehensively, it should have higher weight. The matching level of feature item ti and abstract is represented by $pg(\mathrm{t_i}, \mathrm{S_j}, \mathrm{r_k} \cdot \mathrm{abs})$, and the computing method is shown in Eq. (6).

$$pg\left(t_i, S_j, r_k \cdot abs\right) = \begin{cases} w\left(t_i\right) & t_i \in r_k \cdot abs \\ 0 & \text{other} \end{cases} \tag{6}$$

in the above method, $w(t_i)$ represent the weight of each feature item given by query string.

The matching level of query string q and abstract can be represented as $pg(\mathrm{t_i}, \mathrm{S_j}, \mathrm{r_k} \cdot \mathrm{abs})$, and the computing formula is shown in Eq. (7).

$$PG\left(q, S_j, abs\right) = \sum_{i=1}^{n} pg\left(t_i, S_j, r_k \cdot abs\right) \tag{7}$$

3.1.1 The computing of the similarity of feature item t_i and r_k abstract

Now let's compute the similarity between every feature item of the query string and the result r_k , then the similarity between each feature item t_i and rk abstract can be represented by $sim(t_i, S_j, r_k, \cdot abs)$, which is shown in Eq. (8).

$$sim\left(t_i, S_j, r_k \cdot abs\right) = \begin{cases} \sum_{x=1}^{N(t_i, abs)} \left(1 - \frac{location\left(t_i, k\right)}{length\left(abs\right)}\right) & N\left(t_i, abs\right) > 0 \\ 0 & N\left(t_i, abs\right) = 0 \end{cases} \tag{8}$$

in the above method, $N(t_i, abs)$ represents the times that feature item t_i of the query string occurred in the query result r_k, and length(abs) represents the length of query result r_k abstract, $position(t_i, k)$ represents the k times that feature item t_i occurred in the abstract. Then the computing method of the similarity between query string q and abstract is shown in Eq. (9).

$$sim\left(q, S_j, abs\right) = \sum_{i=1}^{n} sim\left(t_i, S_j, abs\right) \tag{9}$$

3.1.2 Computing of the similarity of query string q and r_k

If the similarity of query string q and abstract is represented as $corr(q, S_j, r_k.abs)$, the computing method is shown in Eq. (10).

$$corr\left(q, S_j, r_k \cdot abs\right) = sim\left(q, S_j, r_k \cdot abs\right) \times pg\left(q, S_j, r_k \cdot abs\right) \tag{10}$$

and the query result r_k can be represented as Eq. (11).

$$corr\left(q, S_j, r_k \cdot tit\right) = sim\left(q, S_j, r_k \cdot tit\right) \times pg\left(q, S_j, r_k \cdot tit\right) \tag{11}$$

the computing of the similarity of query string q and r_k query result. The computing can be more scientific by making weighted summation of the similarity of query string q and title r_k and abstract. The two weights are represented as α and β, and the ultimate similarity as $corr(q, S_j, r_k)$, the formula is as Eq. (12).

$$corr\left(q, S_j, r_k\right) = \alpha * corr\left(q, S_j, r_k \cdot tit\right) + \beta * corr\left(q, S_j, r_k \cdot abs\right) \tag{12}$$

where $\alpha + \beta = 1$.

3.2 Computing of relevant score of the query result

The Borda sorting idea is the accumulated ultimate score of every result, whose query result is voted by search engine, and the score has considered the position of query result. In this paper, the ultimate relevant score of query result rk is represented as the above weighted summation of position relevance $pos(q, S_j, r_k)$ and the similarity $corr(q, S_j, r_k)$ between query word string and query result rk. The computing method is as Eq. (13).

$$Score\left(q, S_j, r_k\right) = \omega * pos\left(q, S_j, r_k\right) + \theta * corr\left(q, S_j, r_k\right) \tag{13}$$

where ω and θ are weight factors, and $\omega + \theta = 1$.

3.3 Score computing of the ultimate borda of the query result

Through the above steps we have computed the relevant score of the query result r_k, the score of the results searched by many member search engines is the sum of each one of them. Hence, for n member search engines, the Borda score of the query result r_k is represented by $Borda(q, S_j, r_k)$ as Eq. (14).

$$Borda\left(q, S_j, r_k\right) = \sum_{j=1}^{n} Score\left(q, S_j, r_k\right) \tag{14}$$

in the end, descending the query result according to the score of $Borda(q, S_j, r_k)$, and display it to users.

4 Experiment results and analysis

Authors should discuss the results and how they can be interpreted in perspective of previous studies and of the working hypotheses. The findings and their implications should be discussed in the broadest context possible. Future research directions may also be highlighted.

4.1 Selection of dataset

To analyze and exam the algorithm with experiment, we build a prototype system of search engine, whose member search engines are Baidu, Yahoo, Bing and Sogou. We do experiments on representative retrieval topics, and each time of the search concludes the

first 30 results of their member queries.

The query dataset uses the top 100 query words on the search ranking list of search engines in 2004. In this experiment, we use query words of different topics. In the end, we compare them concerning effect of algorithm.

4.2 Evaluation method

Common evaluation methods in search engine domain are recall, precision, system response time, etc. Because of the principle of element search engine, normally they all can get pretty high recall ratio, and the formula we use in this paper to evaluate the efficiency of algorithm by precision is demonstrated in Eq. (15).

$$precision = \frac{the\ number\ of\ \ query\ correlation}{total\ \ number\ of\ \ search\ results} \tag{15}$$

4.3 Result and analyze

4.3.1 The influence on the algorithm by weight factors ω and θ

In the algorithm in the paper, the weight factors ω and θ influence the weight of position and similarity factors, hence their dereferencing have great influence on the algorithm. In the experiment, the dereferencing of ω vary from 0.1 to 0.9, and the variety of the average precision on different dereferencing is demonstrated in Fig. 1.

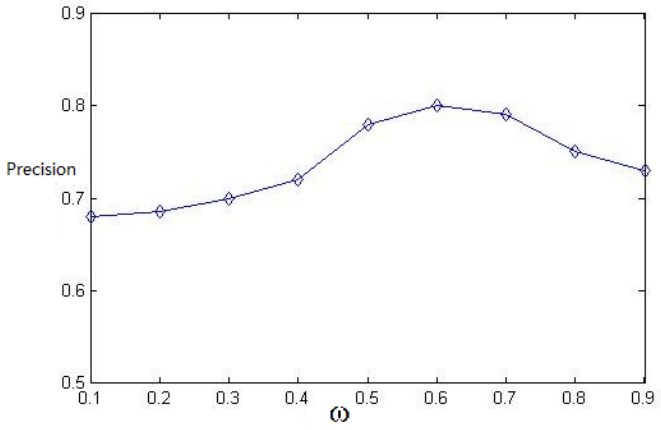

Figure 1: The relationship between the dereferencing (ω) and the average precision

As we can see from Fig. 1, when ω<0.4, the variety remains barely changed, but when the dereferencing is around 0.6, the precision reaches its highest point, and then in the downward trend. Hence, in the following experiment, the dereferencing of weight factor is ω=0.6, which means the great value of the result permutation position in the return results collection of its search engine.

4.3.2 Comparison between the algorithm in the paper and independent search engine

To verify the effectiveness of the algorithm in the paper, we will compare the element search engine NMSE of the algorithm with the average precision rate and recall rate of its

element search engine. Different search engine will have different effect in accordance with different query subject, for example, among the search engines, the precision rate of searching "Ebola virus" of Baidu is 0.75, of Yahoo is 0.68, of Bing is 0.59, and of Sogou is 0.67. And when searching other words, we receive different results. In the following section, we will search with every independent search engine and the element search engine using the algorithm in the paper, and the effect of average value comparison is demonstrated in Fig. 2.

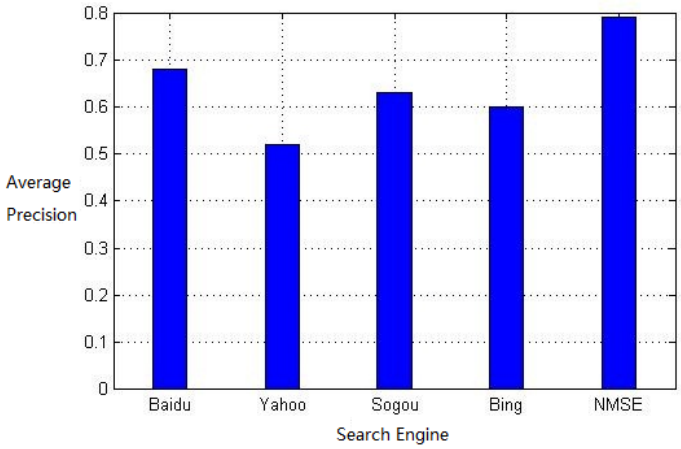

Figure 2: Average precision comparison

As we can see from Fig. 2, Baidu remains the leading role in Chinese searching, while the element search engine used the algorithm in the paper has higher average precision rate than Baidu when searching different subjects.

4.3.3 Comparison between the improved algorithm in the paper and classic element search engine sorting algorithm

The algorithm in the paper is improved based on the Borda sorting algorithm in element search engine. To verify the efficiency of the algorithm, we now choose several classic sorting algorithms as comparison object, which are Borda sorting algorithm, Round-Robin algorithm and Comb SUM algorithm.

The Round-Robin algorithm adopts the idea of polling, and its algorithm method is to first arrange the member search engines in a certain order, and when the search engine does results merging, get the first result of its member search engine, then the second, and so on. Comb SUM algorithm is a relevance score method, because local similarity of different search engines cannot be compared but to composed directly, we can get normalized relevance score by mapping the position of search result to [0,1]. Com SUM algorithm is to add all the relevance scores that occur in different search engines as the ultimate relevance score, and sorting in this order.

We now select query key words of different subjects from the dataset, and do search experiment for ten consecutive years under the Web environment, in the end, we extract the average value. The comparison effect of the four algorithms is demonstrated in Fig. 3.

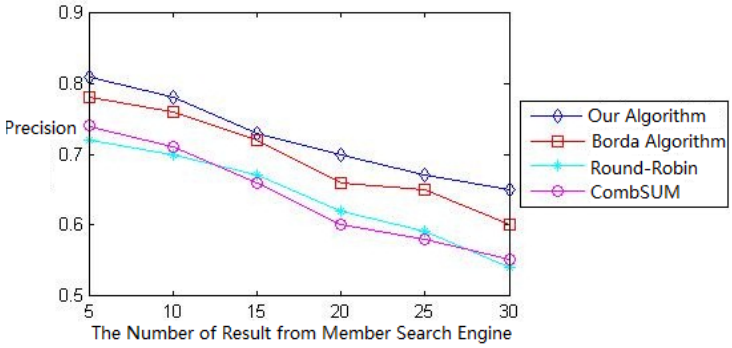

Figure 3: Precision comparison diagrams between our algorithm and traditional algorithm

As we can see from Fig. 3, along with the increase of results, the precision is declining. The algorithm in the paper has better precision than traditional Borda sorting algorithm, and also higher than the other two traditional sorting algorithm, which means that the improved algorithm is very effective.

5 Conclusions

The improved algorithm has made the following improvements on the basis of traditional Borda sorting algorithm. (1) Normalize the sorting position of the query results, and replace the position score with position relevance. We cannot directly compare the query result position in the search engine, because the results returned from each search engine are few yet different, which is why it is not accurate to represent the position score by the quantity, whereas position relevance can better represent the relevance between position and query word. (2) Considering the current relevance algorithm is to first download the original document, then compute in unification the global similarity, which waste a lot of time and network resource thus cannot be accepted by users. According to research, the title and abstract of search results centralized the main information of the websites, so in the paper, we compute global relevance with information extracted from titles and abstracts returned by websites. (3) When computing the similarity with titles and abstracts, we combined the matching weight of query words and results, which makes the computing more accurate. However, there exist some shortcomings in time efficiency. Besides, it does not take individualized needs of different users into consideration. Element search engine will be more personalize, professionalize, and intellectualize, which is also a hotspot for future element search engine research.

Acknowledgement: This work was funded by the National Natural Science Foundation of China under Grant (No. 61772152 and No. 61502037), the Basic Research Project (Nos. JCKY2016206B001, JCKY2014206C002 and JCKY2017604C010), and the Technical Foundation Project (No. JSQB2017206C002).

References

Chen, X.; Xu, L. (2016): An educational resource retrieval mechanism based on lucene and topic index. *13th Web Information Systems and Applications Conference*, pp. 125-128.

Cho, M.; Brand, D.; Bordawekar, R.; Finkler, U.; Kulandaisamy, V. et al. (2015): PARADIS: an efficient parallel algorithm for in-place radix sort. *Proceedings of the Vldb Endowment*, vol. 8, pp. 1518-1529.

García-Lapresta, J. L.; Martínez-Panero, M. (2002): Borda count versus approval voting: a fuzzy approach. *Public Choice*, vol. 112, pp. 167-184.

Katsirelos, G.; Walsh, T.; Davies, J.; Narodytska, N. (2011): *Complexity of and Algorithms for Borda Manipulation*. San-Francisco, United States.

Lawrence, S.; Giles, C. L. (1998): Inquirus, the NECI meta search engine. *Computer Networks & Isdn Systems*, vol. 30, pp 95-105.

Ping, W. U. (2003): The study on large scale duplicated web pages of chinesefast deletion algorithm based on string of feature code. *Journal of Chinese Information Processing*, vol. 2, pp 28-35.

Smyth, B.; Boydell, O. (2010): Meta Search Engine. US: freshpatents.

Yamamoto, K.; Fujii, R.; Toyofuku, Y.; Saito, T.; Koseki, H. et al. (2001): The KDEL receptor mediates a retrieval mechanism that contributes to quality control at the endoplasmic reticulum. *Embo Journal*, vol. 20, pp. 3082-3091.

Yang, K. (2005): Information retrieval on the web. *ACM Computing Surveys*, vol. 39, pp. 33-80.

Yong, L.; Zulin, W. (2011): Rank aggregation performance analysis for borda and local search algorithm. *International Symposium on Information Science and Engineering*, pp. 125-128.

Zhang, Q. G.; Yu, G. B.; Liao, H. S.; Sui, S. L. (2004): A Processing model for the query results obtained by meta-search engines. *Journal of South China University of Technology*, vol. 32, pp 47-51, 57.