TELKOMNIKA, Vol. 13, No. 4, December 2015, pp. 1495~1504
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v13i4.2736

The Analysis of Rank Fusion Techniques to Improve Query Relevance

Diyah Puspitaningrum*[1], Jeri Apriansyah Pagua[2], Aan Erlansari[3], Fauzi[4], Rusdi Efendi[5], Desi Andreswari[6], I.S.W.B. Prasetya[7]
[1,2,3,4,5,6]Department of Computer Science, Faculty of Engineering, University of Bengkulu
WR Supratman Street, Kandang Limun, Bengkulu 38371, Sumatera, Indonesia
[7]Department of Information and Computing Sciences, Utrecht University
PO Box 80.089, 3508 TB Utrecht, The Netherlands
*Corresponding author, e-mails: diyahpuspitaningrum@gmail.com[1], jeri.apriansyahpagua@unib.ac.id[2], aan.erlansari@unib.ac.id[3], fauzi.faisal@unib.ac.id[4], rusdi.efendi@unib.ac.id[5], desi.andreswari@unib.ac.id[6], S.W.B.Prasetya@uu.nl[7]

Abstract

Rank fusion meta-search engine algorithms can be used to merge web search results of multiple search engines. In this paper we introduce two variants of the Weighted Borda-Fuse algorithm. The first variant retrieves documents based on popularities of component engines. The second one is based on k user-defined toplist of component engines. In this research, experiments were performed on k={50,100,200} toplist with AND/OR combinations implemented on 'UNIB Meta Fusion' meta-search engine prototype which employed 3 out of 5 popular search engines. Both of our two algorithms outperformed other rank fusion algorithms (relevance score is upto 0.76 *compare to Google that is 0.27, at P@10). The pseudo-relevance automatic judgement techniques involved are Reciprocal Rank, Borda Count, and Condorcet. The optimal setting was reached for queries with operator "AND" (degree 1) or "AND ... AND" (degree 2) with k=200. The 'UNIB Meta Fusion' meta-search engine system was built correctly.*

Keywords*: Weighted Borda-Fuse, rank fusion, meta-search engine, pseudo-relevance automatic judgement, query relevance*

1. Introduction

There are many proposals for a meta-search engine (MSE). Given a query (a set of keywords), typically an MSE system retrieves web pages that are relevant to the query by exploiting all its underlying search engines. It sends the query to these engines; the results obtained are then merged and ranked. It returns final web documents ranked by relevance. In the Helios architecture [1] the MSE system uses standard merger and ranker modules. To achieve high performance it utilizes async I/O and parallel TCP connections with the remote search engines. In the Tadpole architecture [2, 3], the rank fusion algorithms are based on a variety of parameters, such as the rank order and the number of times an URL appears in the results of each of its search engines components, to compute a weight for each collected results [2-4]. There is also the concern of user specific needs. For example, an MSE should ideally let the user choose his favourite search engines from an available list, and do query modifications, as well as explore available rank fusion techniques [2].

In general, rank fusion algorithms offer improvement of the relevance scores of the returned documents of multiple search engines. Dwork, Kumar, Naor, and Sivakumar propose the use of rank aggregation methods for MSEs viz. the Borda's method, Footrule and Scaled Footrule, and Markov Chain methods [5]. Lam and Leung propose a complete directed graph viz. MST Algorithm [6]. Supervised rank aggregation methods such as Borda-Fuse and supervised Markov Chain based methods are investigated in [7]. KE algorithm [8] and its variants [4] exploit the ranking on the results that an MSE receives from its component engines, by considering the number of documents appearances in the component engines' lists with equal reliability assumption of those engines. Another rank fusion MSE algorithm named Count Function [9] defines web documents ranking as summing ranks as per positions of a URL

Received August 13, 2014; Revised November 20, 2015; Accepted November 30, 2015

divided by the count of URL documents. Aslam and Montague introduces a voting fusion method named *Borda-Fuse* which is an adaptation of the Borda Count election process [10]. Borda-Fuse, tested in two of the five tests using TREC test data, performed better than the best component IR system in the election results [10, 11]. Borda-Fuse that is extended to a weighted variant is called Weighted Borda-Fuse algorithm; it multiplies the points in which a retrieval system S_i assigns to a candidate URL with a system weight W_i. By using improved performance weights, Weighted Borda-Fuse has the potential of outperforming CombMNZ [12].

The coverage of each search engine is limited, only about 1% of billion pages are in the surface web while the rest are in the deep web. Therefore it is interesting to know how to merge different search engines and how deep we should crawl the web to potentially retrieved still relevant documents. Getting less search engines report ranking scores, we can convert the local ranks into local ranking scores. The KE algorithm [13] is a score-based method that exploits ranking of the search results of the component engines where all those engines are treated equally reliable. Consider a document *x*. In KE, the local ranks (r_i) of *x* as returned from all component engines of an MSE are summed and converted to a single weight score (W_{ke}) using this formula:

$$W_{ke} = \frac{\sum_{i=1}^{m} r_i}{((n)^m * (\frac{k}{10} + 1)^n)} \tag{1}$$

where $\sum_{i=1}^{m} r_i$ is the sum of all rankings from each search engines that the document appears, *n* is the number of component engines where the document appears in their results, *m* is the total number of component engines exploited, and *k* is the number of toplist documents crawled from each component engine. The lesser the weight, the better the ranking score is.

Patel and Shah propose to simply use the Count Function to compute the ranking of an MSE document [9]. The ranking of the document *x* is computed as follows:

$$Rank\ (x) = \frac{\sum_{i=1}^{n} P_i(x)}{count\ (x)} \tag{2}$$

where $P_i(x)$ is the local rank of document *x* returned from a component engine. $P_i(x)$ in (2) is same with r_i in (1). Unlike the KE algorithm, the documents are here ranked in descending order (the higher the weight the better the ranking score).

Borda Count [10] is a voting-based data fusion that is adopted to a meta-search engine environment in the Weighted Borda-Fuse (WBF) algorithm [14, 15]. Different from the KE and Count Function algorithms, in WBF, component search engines do not have to be treated as equally realiable. The votes for a web document that lays on the local rank *i* of the component search engine *j* are

$$V(r_{i,j}) = w_j * (\max{}_k (r_k) - i + 1) \tag{3}$$

where w_j is the weight of *j* , and $\max_k(r_k)$ is the number of toplist documents crawled from component search engine *k*. Retrieved web documents that appear in more than one search engines receive the sum of their votes. The documents are ranked in descending order of the total votes they receive (the higher the vote the better the ranking score) [3, 7, 10, 16].

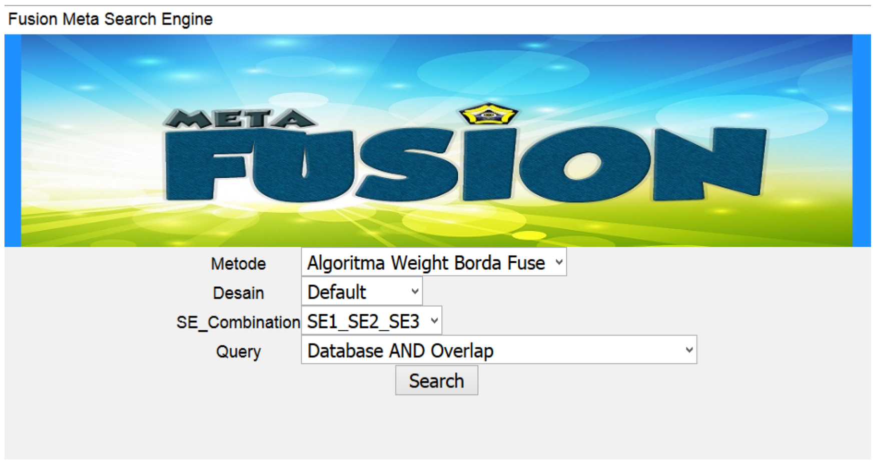
Evaluation key parameters for ranking strategies in an MSE can be viewed (optionally) by its algorithmic complexity, rank aggregation time, overlap across search engines, and the precisions from the user's perspective [3, 17]. For situations where a search engine joins a meta-search engine on the fly [18], some rank fusion algorithms can not be implemented due to estimating a search engine score usually requires enough sample data from each component engine. In our work, we use several *k*-toplist values of component engines.

Research of query operators' utilization has reported that about 10% of search engine users use advanced query operators such as Boolean AND/OR (and the rest only use simple queries) [19]. We are interested in examining the effect of complexity degree of query, in particular degree one and two (using respectively one and two operators). According to [20], the performance of queries of complexity one outperform that of complexity two, in all cases; but is the decrease in relevance significant enough? All of our query experiments in this research are implemented on searching web documents using 3 search engines: Google, Altavista, and Fast search engine. It would be interesting to see whether the results are consistent using different component search engines.

This paper introduces two variants of Weight Borda Fuse algorithm that aims to improve query relevance of web search results in a meta-search engine environment. The rest of this paper is organized as follows: Section 2 describes our prototype of meta-search engine system and the proposed Weighted Boda-Fuse variants, Section 3 reports our experimental results, and finally Section 4 gives some conclusions.

2. The 'UNIB Meta Fusion' Prototype and The Algorithms

To overcome issues defined in Section 1, we built a prototype of a user adaptive MSE called 'UNIB Meta Fusion' that allows a user to choose his favourite search engines and *k*-toplist set up of retrieved web documents (*k* = 50, 100, 200). In this research we experiment with our two proposed variants of the WBF algorithm [14, 15], KE algorithm [8], and Count Function algorithm [9]. Relevance is computed using two IR metrics: precisions and MRR. All is measured from 10-toplist of MSE against 10-toplist of *pseudo-relevance* technique.



Metode	Algoritma Weight Borda Fuse
Desain	Default
SE_Combination	SE1_SE2_SE3
Query	Database AND Overlap

Figure 1. The web interface of 'UNIB Meta Fusion'. A user can choose his preferred rank fusion algorithm, *k*-toplist, combination of component search engines as well as query

Our 'UNIB Meta Fusion' (Figure 1) is a meta-search engine prototype that supports a choice of 3 over 5 well-known search engines. Related to Figure 1, $SE = \{SE_1, \ldots, SE_5\}$ is a list of component search engines: Google, Bing, Ask.com, Lycos, and Exalead, respectively. Since we intended this project for research purpose, processes of retrieving, parsing, merging, ranking, and reporting the results of the search engines are done separately in the off-line mode. The prototype will show only the list of URL results of the best rank fusion method. By 'UNIB Meta Fusion' we especially want to investigate which rank fusion method outperforms the others, in different toplist retrieved documents' setup. We modify the Weighted Borda-Fuse algorithm into 2 variants, called '*Default*' WBF and '*MyOwn*' WBF. '*Default*' sets up the number of retrieved toplist documents based on the popularity of a search engine whereas in '*MyOwn*' the user is free in defining the combination of toplist of multiple components search engines.

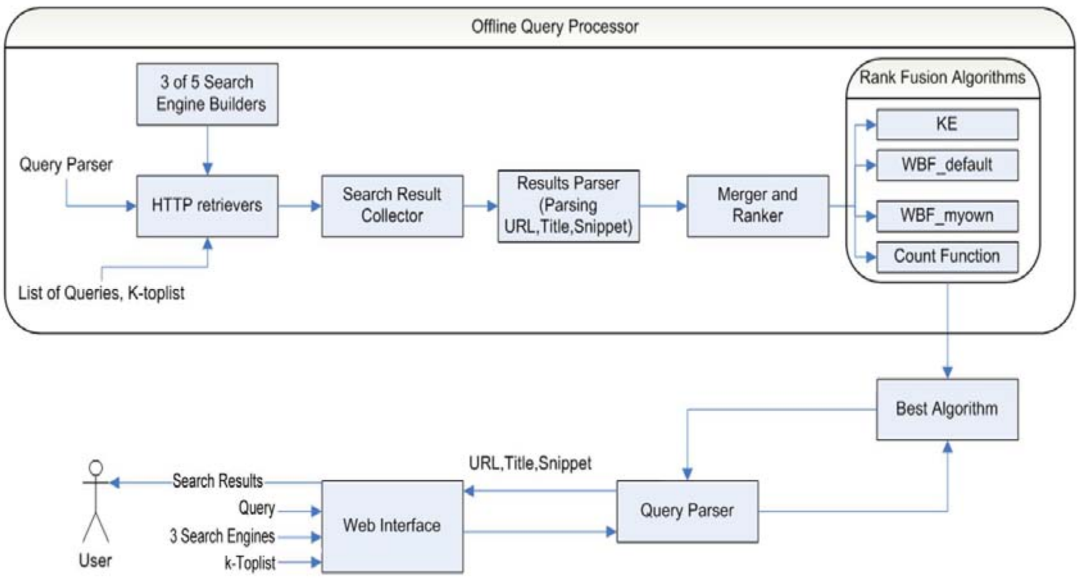


Figure 2. The architecture of 'UNIB Meta Fusion'

Figure 2 shows the architecture of 'UNIB Meta Fusion' system. The *Web Interface* allows a user to submit a query, specify a choice of three search engines, and a number *k* for how many toplist documents will be retrieved from each of the specified search engine. The *Query Parser* creates an appropriate format for the query and passes the information of *k*–toplist URLs and choices of search engines to *Best Algorithm*. *Best Algorithm* employs only best performed rank fusion algorithm viz. the algorithm that has the highest documents retrieval's relevance score to a set of gold standard retrieval relevant documents generated by either Rank Relevance, Borda Count, or Condorcet technique such as suggested by Nuray and Can [21]. *Best Algorithm* then returns the merged and ranked list of documents to *Query Parser* that in turn will return sets of [URL, title, snippet] as query search results into user.

The *Offline Query Processor* is the most time consuming part of this research. Given a list of queries, sets of *k*-toplist of component engines (*k* = 50, 100, 200), and sets of combinations of 3 out of 5 search engines, the *HTTP Retrievers* handles the network communications, and the *Search Results Collector* stores separately each *k*-toplist documents of the search engines. *Results Parser* parses the lists into URLs, titles, and snippets. These are then combined in *Merger and Ranker* and then ranked by one of the *Rank Fusion Algorithms*: KE, '*Default*' WBF, '*MyOwn*' WBF, and Count Function algorithm. We then carefully investigate and decide which algorithm that work best to be employed in *Best Algorithm*.

The 'UNIB Meta Fusion' returns documents processed by the off-line query processor using 10 queries of two terms and three terms length (see Table 1). Those queries are extended further using operators AND/OR. For example, for three terms query of "Java applet programming" is combined further to form 4 different new queries: "Java AND applet AND

Table 1. The Multi Domain Queries [20]

Two Terms Queries	**Three Terms Queries**
database overlap	comparative education methodology
multilingual OPACs	java applet programming
programming algorithm	indexing AND digital libraries
road-map plan	geographical stroke incidence
adolescent alcoholism	culturally responsive teaching

programming", "Java AND applet OR programming", "Java OR applet AND programming", and "Java OR applet OR programming". Two terms length queries get a similar treatment. At the end, the 10 queries are extended into 30 different queries.

2.1. The User Defined '*MyOwn*' WBF Algorithm

The user defined '*MyOwn*' WBF algorithm allows a user to define his own search engine weights. By considering (3), we set the r_k of each search engine with an equal value of *k*-toplist; but with different weights as suggested by the user. The weights are usually proportionate with the user's trust of relevancy of the corresponding search engines. In other words, by using the user defined '*MyOwn*' WBF algorithm we would like to know whether the MSE system will produce a good relevance score if we treat the system with different weights, given a user defined *k* values of *k*-toplist crawling document for all of component search engines.

The '*MyOwn*' WBF's processes for meta-search 3 of 5 search engines are as follow:

Step 1. The user specifies *k*; this determines the number of documents of all of component search engines will later retrieve (*k*-toplist), e.g. $k = 200$.

Step 2. Define the set of search engines $SE = \{SE_1, SE_2, \ldots, SE_n\}$ that are available for meta-searching. // In our case $n = 5$.

Step 3. The user set the weight w_j ($j = 1, \ldots, n$) for each search engine in Step 2. // For example $w_j = \{50, 30, 20, 25, 15\}$.

Step 4. The user selects three out of *n* search engines to be used.

Step 5. For each three engines from Step 4, set $\max_k(r_k)$ to be the value of the *constant k* from Step 1 for each component engine.

Step 6. For each document found in the *k*-toplist returned by each component engine chosen in Step 4:

Step 6a. Compute $V(r_{i,j})$, using equation (3), where *i* is the ranking of the document in engine *j*.

Step 6b. Compute the document's WBF ranking score:

$$WBF\_ranking\_score = (\sum_{j=1}^{3} V(r_{i,j})) * total\_SE$$

Consider a document *x*. The *total_SE* is the number of search engines where document *x* is found.

Step 7. Order the found documents descendingly by their WBF ranking scores.

Step 8. Presents the top 10 documents obtained from Step 7 to the user.

Example 1:

Consider a meta-search engine system built using '*MyOwn*' WBF algorithm. Let $n = 5$, and the chosen search engines are SE_1, SE_2, and SE_3, with $k = 200$ for all of them (this determines the *k*-toplist to be retrieved). Suppose the user specifies {50, 30, 20} as the weights of the component engines respectively. Assume we have 3 documents: Doc_1, Doc_2, and Doc_3 and several facts:

- Doc_1 is found respectively at rank 8,9, and 11 in the toplists of SE_1, SE_2, and SE_3.
- Doc_2 is found at rank 9 and 13 in the toplists of SE_1 and SE_3; it is not found by SE_2.
- Doc_3 is found respectively at rank 3, 5, and 4.

The WBF scoring of these documents is then shown in Table 2.

Table 2. The Scoring of WBF Algorithm by Considering (3)

Query	SE_1 (50%)	SE_2 (30%)	SE_3 (20%)	WBF Ranking Score
Doc_1	50*(200-8+1) = 9650	30*(200-9+1) = 5760	20*(200-11+1)= 3800	(9650+5760+3800)*3 = 57630
Doc_2	50*(200-9+1) = 9600	Not found	20*(200-13+1)= 3760	(9600+0+3760)*2 = 26720
Doc_3	50*(200-3+1) = 9900	30*(200-5+1) = 5880	20*(200-4+1)= 3940	(9900+5880+3940)*3 = 59160

Table 2 shows the ordering: $Doc_3 > Doc_1 > Doc_2$.

2.2. The '*Default*' WBF Algorithm

The '*Default*' WBF algorithm is a special instance of the '*MyOwn*' WBF but with the differences in Step 1 and Step 5 of the '*MyOwn*' algorithm. In '*Default*' WBF, the *k* values for *k*-toplist of each component engine are influenced by each component engine's weight. In the '*Default*' WBF, consider 3 component search engines and *k* is an element of {50, 100, 200}, then for equation (3) we set $\max_k(r_k)$ to be 200 for the engine with the highest w_j; 100 for the engine with the second highest w_j, and 50 for the third engine.

Example 2:

Consider a meta-search engine system built using '*Default*' WBF algorithm. Let n = 5, and the chosen search engines are SE_1, SE_2, and SE_3. Suppose the user specifies {50, 30, 20} as the respective weigth of those engines. Assume we have 3 documents: Doc_1, Doc_2, and Doc_3 and several facts:

- Doc_1 is found on SE_1 in rank 8 from SE_1 toplist, on SE_2 in rank 9 from SE_2 toplist, and on SE_3 in rank 11 from SE_3 toplist.
- Doc_2 is found on SE_1 in rank 9 from SE_1 toplist, not found on SE_2, and on SE_3 in rank 13 from SE_3 toplist.
- Doc_3 is found on SE_1 in rank 3 from SE_1 toplist, on SE_2 in rank 5 from SE_2 toplist, and on SE_3 in rank 4 from SE_3 toplist.

Then the scoring of '*Default*' WBF as on Table 3.

Table 3. The Scoring of WBF Algorithm by considering (3)

Query	SE_1 (50%)	SE_2 (30%)	SE_3 (20%)	WBF Ranking Score
Doc_1	50*(200-8+1) = 9650	30*(100-9+1) = 2760	20*(50-11+1) = 800	(9650+2760+800)*3 = 39630
Doc_2	50*(200-9+1) = 9600	Not found	20*(50-13+1) = 760	(9600+0+760)*2 = 20720
Doc_3	50*(200-3+1) = 9900	30*(100-5+1) = 2880	20*(50-4+1) = 940	(9900+2880+940)*3 = 41160

From Table 3 we have $Doc_3 > Doc_1 > Doc_2$ in the rank order of the MSE system. This rank order is influenced by WBF scores of each documents. The more relevant a document, the WBF will put it into higher position of web search retrieval of the MSE.

3. Results and Analysis

We have described two variants for ranking in WBF meta-search. We would like to compare them with other existing rank fusion algorithms: the KE algorithm [8] and the Count Function algorithm [9]. Queries are sent to each search engine, retrieving toplists until *k* {*k* = 50, 100, 200} URLs have been crawled from each component search engine and merged by the four algorithms ('*Default*' WBF, '*MyOwn*' WBF, KE, and Count Function). For evaluation, as the queries are multi domain (not limited such as TREC datasets; these multi domain queries are for simulating real world situations) and also since using human judgment is expensive, we evaluate our system using three different gold standards: Reciprocal Rank (RR), Borda Count (BC), and Condorcet methods. The lattests are known as "Pseudo-Relevance" datasets as suggested in [21].

In this research, all experiments are executed on an Acer 4741 machine with an Intel core i3 and 5GB RAM. All prototyping processes from retrieval, parsing, merging, ranking, until presenting the query results to user, are implemented in Python. The language is efficient and a fast Python module, named *webpy*, helps in providing a user friendly interface of the MSE prototype. For the evaluation of the tasks in all of our experiments, we adopted two metrics that capture the relevance at different aspects [22]:

- Precision at rank *n* (P@*n*): Precision at rank *n* is defined as the proportion of retrieved documents that is relevant with the gold standard, averaged over all documents.
- Mean Reciprocal Rank (MRR): MRR measures where in the ranking the first relevant document (with the gold standard) is returned by the system, averaged over all the

documents. This measure provides insight in the ability of the system to return a relevant document at the top of the ranking.

Table 4 to Table 6 shows the results in terms of both P@*n* and MRR of different *k*-toplist compared to gold standards (Pseudo-Relevance). From these results, we can see that our proposed method (‘*Default*’ and ‘*MyOwn*’ WBF) achieves the best results in terms of both P@*n* and MRR. In general, these verifies the effectiveness of our proposed method for rank aggregation.

For both WBF variants in all of experiments in this research we set up weights = {30, 20,15,25,10} respectively for Google, Bing, AskJeeves, Lycos (powered by Yahoo!), and Exalead. All relevance scores is obtained using best P@10 of each rank fusion algorithm compare to best P@10 of Google. Google has chosen as a benchmark since Google shows best performance of any individual search engines. We order individual component engines by their weights for convenience (Table 4 to Table 6).

Table 4. Results of Different Methods for MSE, compared to Pseudo-Relevance Sets at *k*=50

System	P@10_RR	P@10_BC	P@10_Condorcet	MRR_RR	MRR_BC	MRR_Condorcet
MSE Rank Fusion Performance						
MyOwn WBF	0.6563	0.6950	0.5577	0.9853	0.9627	0.4339
Default WBF	0.6530	0.7300	0.6300	0.8877	0.7543	0.4453
KE	0.6650	0.6613	0.5763	0.9132	0.8747	0.4009
Count Function	0.2687	0.2483	0.3050	0.3238	0.1373	0.1417
Individual Component Engines Performance						
Google	0.3267	0.3437	0.2933	0.4113	0.4356	0.2949
Lycos	0.2893	0.3073	0.2643	0.4088	0.4432	0.2899
Bing	0.1783	0.1800	0.1630	0.3385	0.3299	0.2015
Ask.com	0.2010	0.2013	0.1907	0.3625	0.3676	0.2053
Exalead	0.0680	0.0627	0.0727	0.1188	0.1227	0.1135

Table 5. Results of Different Methods for MSE, compared to Pseudo-Relevance Sets at *k*=100

System	P@10_RR	P@10_BC	P@10_Condorcet	MRR_RR	MRR_BC	MRR_Condorcet
MSE Rank Fusion Performance						
MyOwn WBF	0.7103	0.7470	0.6023	0.9764	0.9340	0.4097
Default WBF	0.6683	0.6953	0.6447	0.9294	0.9285	0.4378
KE	0.6970	0.6993	0.5957	0.9214	0.8870	0.3875
Count Function	0.2533	0.2273	0.3000	0.3205	0.1277	0.1409
Individual Component Engines Performance						
Google	0.2987	0.3097	0.2863	0.3909	0.4264	0.3015
Lycos	0.2717	0.2853	0.2567	0.3829	0.4091	0.2715
Bing	0.1663	0.1693	0.1583	0.3242	0.3255	0.2006
Ask.com	0.1960	0.1980	0.1827	0.3468	0.3574	0.2036
Exalead	0.0633	0.0597	0.0643	0.1334	0.1406	0.0995

Table 6. Results of Different Methods for MSE, compared to Pseudo-Relevance Sets at *k*=200

System	P@10_RR	P@10_BC	P@10_Condorcet	MRR_RR	MRR_BC	MRR_Condorcet
MSE Rank Fusion Performance						
MyOwn WBF	0.7253	0.7630	0.5953	0.9683	0.8880	0.3979
Default WBF	0.6200	0.6377	0.6280	0.9153	0.9376	0.4463
KE	0.7050	0.7213	0.5887	0.9131	0.8541	0.3777
Count Function	0.2503	0.2203	0.3003	0.3218	0.1270	0.1450
Individual Component Engines Performance						
Google	0.2653	0.2703	0.2743	0.3843	0.4300	0.3116
Lycos	0.2370	0.2437	0.2417	0.3506	0.4027	0.2789
Bing	0.1497	0.1527	0.1497	0.3245	0.3361	0.2179
Ask.com	0.1884	0.1887	0.1747	0.3489	0.3613	0.2234
Exalead	0.0580	0.0563	0.0573	0.1418	0.1404	0.0980

The data in Table 4 are that of *k*-toplist with *k* = 50. Table 4 clearly shows that the highest relevance score (P@10) for '*MyOwn*' WBF is 0.6950 for Borda Count, two times higher than that of Google (2.02 times); that of '*Default*' WBF is 0.7300 for Borda Count, also two times higher than that of Google (2.12 times); and that of KE is 0.6650 for Reciprocal Rank that almost two times higher than that of Google (1.93 times). From Table 4, the best P@10 for Count Function is only 0.3050 for Condorcet that below the P@10 of Google (lesser, 0.89 times). From the MRR perspective, in general at *k* = 50 the two WBF methods outperform the others (WBFs do quite well in getting the first correct position).

Table 5 shows the same data, but for *k* = 100. It also shows improvement in P@10 with highest relevance score is for '*MyOwn*' WBF with 0.7470 for Borda Count (2.41 times higher than P@10 of Google). The highest score of '*Default*' WBF is now 0.6953 for Borda Count pseudo-relevance sets (2.25 times); that of KE is now 0.6993 for Borda Count (2.26 times); and Count Function is 0.300 for Condorcet (lesser, only 0.97 times). All against best P@10 of Google. In terms of MRR, the two WBF algorithms still perform better than the others.

Table 6 shows the results for *k*-toplist with *k* = 200. From Table 6 the highest improvement in relevance score (P@10) against best P@10 of Google for '*MyOwn*' WBF is 0.7630 for Borda Count pseudo-relevance sets (2.78 times than that of Google); for '*Default*' WBF the highest is 0.6377 for Borda Count pseudo-relevance sets (2.32 times); for KE method, the highest is 0.7213 for Borda Count pseudo-relevance sets (2.63 times); and for Count Function the highest is 0.3003 for Condorcet (1.09 times). All against P@10 of Google. In general as others *k*-toplist, at *k* = 200 the MRR of WBFs also stable outperform other rank fusion methods.

As a conclusion, both the '*Default*' WBF and the user defined '*MyOwn*' WBF produce best results. They outperform other algorithms such as KE [13] and Count Function [9]. The relevance of Count Function algorithm is far below the WBFs, this is due to the simplicity of the algorithm that only computes the sum of local rank of document *x* returned from each component engines divides by total number of occurence of *x* in all meta-search engine components. The Count Function algorithm does not consider neither popularities of component engines nor the number of crawled toplist documents. KE algorithm concerns about how many number of toplist documents crawled from each component engines but popularities are missed.

From experiments also we found that the best *k*-toplist of each component search engines (in terms of precision) is reached for *k* = 200 ('*MyOwn*' WBF with precision of 2.78 times higher than that of Google), followed by 2.41 times higher for '*MyOwn*' WBF with *k* = 100, and 2.12 times higher for '*Default*' WBF with *k* = 50, all compared to Google. Therefore the best methods found among rank fusion methods are the Weighted Borda-Fuse algorithms. The '*Default*' WBF is suitable for small datasets while the '*MyOwn*' WBF is suitable for bigger datasets (e.g. $\geq$100-toplist crawled from each component engine). The best gold standard is achieved by Borda Count technique.

From Table 4 to Table 6, mostly in all cases the weight that has been set up influence the result. For example, high weight on Google will most probably give Google as the best search engine. Except for ask.com that always better than Bing. The Ask.com uses the ExpertRank algorithm that performs better than Bing that uses best trail finding algorithms. The ExpertRank algorithm is based on the HITS algorithm that uses a scheme in which every web page is assigned two scores: the hub score and the authority score. When compare to Google's PageRank algorithm, the Google search engine had more relevant top results, higher quantity of relevant results and that its results remained more stable than the ExpertRank algorithm [23]. While Google's search algorithms are very dependent on HTML text when it comes to indexing websites, multimedia contents (images, video, audio, Flash, and others) are far better with Bing [24]. In all of our experiments we ignore any multimedia contents and focus on text (structured information) this is the rationale for why ask.com always better than Bing. The Exalead search engine provides hybrid searching over typed information extracted from structured databases, as well as searching over unstructured text [25]. This semantic search engine is not too success in our experiments, again because we only focus on structured information.

Furthermore, considering the length of the queries, in Table 7 and Table 8 we focus on the use of two or three terms query since most query (97%) of all queries in World Wide Web having less than 6 terms [26]. In our experiments, we do not use queries from TREC since their length average are longer than common queries on internet. Table 7 shows the effect of length of query terms. From the results we know that using two or three terms are optional since there are no significant difference in the relevance scores (P@*n* and MRR). Table 7 supports result in [20] that performance of the query complexity 2 terms always outperforms the performance of the 3 terms, but in our case the difference is not significant thus we leave this as a choice to user.

Table 7. Performance of Borda Count Pseudo-Relevance based on Length of Terms

Length of Terms	P@1	P@3	P@5	P@10	MRR
Three terms	0.6100	0.5629	0.5053	0.3610	0.4909
Two terms	0.6470	0.5926	0.5404	0.3920	0.5257

Table 8. Performance of Borda Count Pseudo-Relevance based on Operators

Operators	P@1	P@3	P@5	P@10	MRR
AND	0.7074	0.6222	0.5806	0.4424	0.5725
OR	0.5867	0.5630	0.5001	0.3416	0.4789
AND ... AND	0.6378	0.6116	0.5630	0.4244	0.5495
AND ... OR	0.6151	0.5593	0.5053	0.3562	0.4879
OR ... AND	0.5945	0.5503	0.4860	0.3379	0.4696
OR ... OR	0.5926	0.5304	0.4670	0.3255	0.4568

To examine the effect of complexity degree of query length, we analyzed queries with degree 1 = 1 operator (two terms) and degree 2 = 2 operators (three terms). As operators we use AND/OR combinations. Table 8 shows relevance scores of Borda Count pseudo-relevance sets obtained from the 'UNIB Meta Fusion' MSE prototype. From Table 8 we suggest the use of operator "AND" for degree 1 and operators "AND ... AND" for degree 2 that stable in producing relevant results.

4. Conclusions

In this paper we briefly described two rank fusion algorithms: '*MyOwn*' WBF and '*Default*' WBF as well as their implementation on the 'UNIB Meta Fusion', a meta-search engine prototype. From experiments we showed that our variants of Weighted Borda-Fuse algorithms stable outperforms other MSE rank fusion methods. We showed that the weight that has been set up influence the result. The '*Default*' WBF is best for small datasets while the '*MyOwn*' WBF is best for larger datasets. The best value of *k*-toplist for crawling the web is achieved for *k* = 200. Furthermore, we suggest the use of operators "AND" or "AND ... AND" each for degree 1 and degree 2 queries to increase relevance with user needs. From experiments there are no significant difference in relevance if a user uses either two or three terms queries while browsing a search engine. As a general conclusion, our system, the MSE prototype 'UNIB Meta Fusion', was built correctly.

Acknowledgment

This work is supported by the University of Bengkulu's Excellent Research Grant #557/UN30.15/LT/2014 from the Research and Community Services Office of the University of Bengkulu (LPPM UNIB), Bengkulu Province, Sumatera, Indonesia.

References

[1] Gulli A, Signorini A. *Building an Open Source Meta Search Engine*. World Wide Web Conference. Chiba, Japan. 2005: 1004-1005.
[2] Meng W. Metasearch Engines. In: Liu L, Ozsu MT. *Editors*. Encyclopedia of Database Systems. 2009 edition. New York, USA: Springer; 2009: 1730-1734.
[3] Mahabhashyam MS, Singitham P. *Tadpole: A Metasearch Engine Evaluation of Meta Search Ranking Strategies*. Stanford University. Report number: CS276A. 2002.

[4] Akritidis L, Katsaros D, Bozanis P. *Effective Ranking Fusion Methods for Personalized Metasearch Engines*. 12th Pan-Hellenic Conference on Informatics (PCI 2008). Samos Island, Greece. 2008: 39-43.

[5] Dwork C, Kumar R, Naor M, Sivakumar D. *Rank Aggregation Methods for the Web*. Proceedings of the 10th International World Wide Web Conference. Hong Kong. 2001: 613-622.

[6] Lam KW, Leung CH. *Rank Aggregation for Meta-search Engines*. Proceedings of the 13th International Conference on World Wide Web. New York. 2004: 384-385.

[7] Liu YT, Liu TY, Qin T, Ma ZM, Li H. *Supervised Rank Aggregation*. Proceedings of WWW 2007 Conference. Banff, Alberta, Canada. 2007: 481-489.

[8] Akritidis L, Voutsakelis G, Katsaros D, Bozanis P. *QuadSearch: A Novel Metasearch Engine*. Proceedings of the 11th Panhellenic Conference on Informatics (PCI 2007). Patras, Greece. 2007: 453-466.

[9] Patel B, Shah D. Ranking Algorithm for Meta Search Engine. *IJAERS International Journal of Advanced Engineering Research and Studies*. 2012; 2(1): 39-40.

[10] Aslam JA, Montague M. *Models for Metasearch*. Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 01). New York. 2001: 276-284.

[11] Christensen HU, Ortiz-Arroyo D. Applying Data Fusion Methods to Passage Retrieval in QAS. In: Haindl M, Kittler J, Roli F. *Editors*. Multiple Classifier Systems: 7th International Workshop, MCS 2007, Prague, Czech Republic, May 23-25, 2007, Proceedings. IEEE Computer Society Press. 2007. p. 82. (Lecture Notes in Computer Science, Vol. 4472). 1st ed. Berlin Heidelberg: Springer-Verlag; 2007: 82-92. Available from: 10.1007/978-3-540-72523-7_9.

[12] Montague M, Aslam J. *Condorcet Fusion for Improved Retrieval*. Proceedings of the 11th Annual ACM Conference on Information and Knowledge Management (CIKM 02). Tysons Corner, VA. 2002: 538-548.

[13] Renda ME, Straccia U. *Web Metasearch: Rank vs. Score based Rank Aggregation Methods*. Proceedings of the ACM Symposium on Applied Computing (SAC). Melbourne, FL. 2003: 841–846.

[14] Fagin R, Kumar R, Sivakumar D. Comparing Top-*k* Lists. *SIAM Journal on Discrete Mathematics*. 2003; 17(1): 134-160.

[15] Dorn J, Naz T. *Structuring Meta-Search Research by Design Patterns*. Proceedings of the International Computer Science and Technology Conference (ICSTC). San Diego, California, USA. 2008.

[16] van Erp M, Schomaker L. *Variants of the Borda Count Method for Combining Ranked Classifier Hypotheses*. Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition. Amsterdam. 2000: 443-452.

[17] Jadidoleslamy H. Search Result Merging and Ranking Strategies in Meta-Search Engines: A Survey. *IJCSI International Journal of Computer Science*. 2012; 9(4): 239-251.

[18] Wu Z, Raghavan V, Du C, Sai K, Meng W, He H, Yu C. *SE-LEGO: Creating Metasearch Engine on Demand*. ACM SIGIR Conference, Demo paper. Toronto, Canada. 2003: 464-464.

[19] Eastman CM, Jansen BJ. Coverage, Relevance, and Ranking: the Impact of Query Operators on Web Search Engine Results. *ACM Transactions on Information Systems*. 2003; 21(4): 383–411.

[20] Mohamed KA-E-F. Merging Multiple Search Results Approach for Meta-Search Engines. PhD Thesis. Pittsburgh, United States: Postgraduate School of Information Sciences, University of Pittsburgh; 2004.

[21] Nuray R, Can F. Automatic Ranking of Information Retrieval Systems using Data Fusion. *Journal of Information Processing and Management*. 2006; 42(3): 595-614.

[22] Sigurbjörnsson B, van Zwol R. *Flickr Tag Recommendation based on Collective Knowledge*. WWW 2008 Conference. Beijing, China. 2008: 327-336.

[23] Zeno G. PageRank vs ExpertRank: Analysis of Their Link Analysis Algorithms for Ranking Search Results. Thesis. Bayamon: Department of Computer Science, University of Puerto Rico; 2010.

[24] Singla A, White RW, Huang J. *Studying Trailfinding Algorithms for Enhanced Web Search*. Proceedings of SIGIR'10. Geneva, Switzerland. 2010: 443-450.

[25] Exalead. *Exalead CloudView Semantics White Paper*. Exalead. Report number: EN.140.001.0-V1.2. 2010.

[26] Jansen B, Spink A, Bateman J, Saracevic T. Real Life Information Retrieval: A Study of User Queries on the Web. *ACM SIGIR Forum*. 1998; 32(1): 5-17.