Fusing blog opinion retrieval results for better effectiveness

Shengli Wu
[a]*School of Computer Science and Telecommunication Engineering*
Jiangsu University, Zhenjiang, China
[b]*School of Computing and Mathematics*
University of Ulster, Newtownabbey, UK

***Abstract*—In recent years, blogs have been very popular on the Web as a grassroots publishing platform. Some research has been conducted on them and blog opinion retrieval is one of the key issues. In this paper, we investigate if data fusion can be useful for improvement of effectiveness of blog opinion retrieval. Extensive experimentation with the runs submitted to the blog opinion retrieval task in TREC 2008 is carried out and a few data fusion methods including CombSum, CombMNZ, Borda count, and the linear combination method are investigated. We observe that generally speaking, all data fusion methods involved are very competitive compared with the best component retrieval system. Especially, the linear combination method with proper training is superior to other data fusion methods and it is able to beat the best component retrieval system by a clear margin. This study demonstrates that data fusion can be an effective technique for blog opinion retrieval if proper fusion methods are applied.**

***Keywords*-Blog system; Opinion retrieval; Data fusion; Linear combination**

I. Introduction

In recent years, blogs have been very popular on the Web as a grassroots publishing platform. There are a large number of them and they cover many different aspects of people's life. A blog may be owned by an individual or a company. Posts on events, opinions, and so on, can be published by the owner and comments can be made by the readers accordingly.

A lot of research has been conducted on blogging systems and some related issues, such as blog search engines, notification mechanisms, detecting splogs (spam blogs), facet-based opinion retrieval, and so on, are discussed in [1]–[5] and others. opinion retrieval is one of the important issues addressed in blogging systems. Usually, an opinion retrieval system is implemented by enhancing an ordinary information retrieval system with an opinion finding mechanism, which may rely on a lexicon of subjective words and phrases, gathered from a variety of manually or automatically built lexical resources.

In information retrieval, the data fusion technique has been used to combine results from different retrieval models, different document representations, different query representations, and so on, to improve effectiveness. Generally speaking, previous investigation on data fusion methods, for example, in [6]–[10] and others, demonstrates that they can improve retrieval effectiveness if carefully arranged. In addition, many recently developed information retrieval systems or toolkits, such as Indri [1], terrier [2], Lucene [3], and others, have also integrated different techniques and components. They can be regarded as fusions of different techniques or components.

In a sense, blog opinion retrieval systems are more complicated than conventional information retrieval systems, and many different kinds of techniques can be used together in any individual blog opinion retrieval systems. In such a scenario, we hypothesize that the data fusion technique is very likely a useful technique for blog opinion retrieval. To our knowledge, data fusion has not been investigated for blog retrieval systems before. In this paper, we empirically investigate this issue by extensive experimentation.

In 2006, TREC [4] introduced the blog retrieval track. At first, only the opinion finding task was carried out. In the following three years, polarity opinion finding and distillation tasks had been added. For each of those tasks, dozens, even hundreds of runs were submitted for evaluation. This provide us a very good benchmark to test all sorts of systems, techniques, and so on. Especially, in the TREC 2008 opinion finding task, a total of 191 runs were submitted from 19 groups [11], and each of them includes a ranked list of up to 1000 post documents for each of a total of 150 queries. Since the number of runs submitted and the number of queries used are large, we consider this is the best data set for the evaluation of the data fusion technique.

II. Data Fusion Methods

In this section, we discuss the data fusion methods used in the experiment. CombSum, CombMNZ, and the linear combination method with performance level weighting (LCP), performance square weighting (LCP2), and weights decided by multiple regression (LCR). The score normalization methods used include Borda and the fitting linear score normalization method,

[1]http://lemurproject.org/indri.php
[2]http://terrier.org/
[3]http://lucene.apache.org/
[4]http://trec.nist.gov/

Suppose for a given query Q, n component retrieval systems are used to search the same document collection C. For document d_j, s_{ij} is the score calculated for d_j from component system ir_i. Then for CombSum [12], we calculate the total score t_j for every document d_j using

$$t_j(CombSum) = \sum_{i=1}^{n} s_{ij}$$

while for CombMNZ [12], we use the equation

$$t_j(CombMNZ) = m * \sum_{i=1}^{n} s_{ij}$$

to calculate scores. Here m is the number of s_{ij} whose value is above zero. For the linear combination method [9], [13], we use the equation

$$t_j(LN) = \sum_{i=1}^{n} (w_i * s_{ij})$$

to calculate scores. Here w_i is the weight predefined for component system ir_i. After the process of calculating scores, all the documents are ranked according to their total scores calculated.

For the linear combination method, a related issue is how to assign weights to component systems. One policy is to connect weight with performance. One straightforward method is the performance-level weighting [6], [14]. That is, if the average performance of system ir_i over a group of training queries is a, then we set a as ir_i's weight. However, it is found that using a power function of performance (such as a^2, a^3, etc.) is more effective than the simple performance-level weighting [9]. Another option is to use multiple linear regression to obtain suitable weights, which can be derived from the geometric probabilistic framework [15].

Suppose there are m queries, n information retrieval systems, and a total of r documents in a document collection C. For each query q^i, all information retrieval systems provides scores for all the documents in the collection. Therefore, we have $(s^i_{1k}, s^i_{2k}, ..., s^i_{nk}, y^i_k)$ for $\{i$=(1, 2, ..., m), k=(1, 2, ..., r)$\}$. Here s^i_{jk} stands for the score assigned by retrieval system ir_j to document d_k for query q^i; y^i_k is the judged relevance score of d_k for query q^i. If binary relevance judgment is used, then it is 1 for relevant documents and 0 otherwise.

Now we want to estimate

$$Y = \{y^i_k; i = (1, 2, ..., m), k = (1, 2, ..., r)\}$$

by a linear combination of scores from all component systems. The least squares estimates of the β's are the values $\hat{\beta}_0$, $\hat{\beta}_1$, $\hat{\beta}_2$, ..., and $\hat{\beta}_n$ for which the quantity

$$u = \sum_{i=1}^{m} \sum_{k=1}^{r} [y^i_k - (\hat{\beta}_0 + \hat{\beta}_1 s^i_{1k} + \hat{\beta}_2 s^i_{2k} + ... + + \hat{\beta}_n s^i_{nk})]^2$$

is a minimum. In the least squares sense the coefficients obtained by multiple linear regression can bring us the optimum fusion results by the linear combination method, since they can be used to make the most accurate estimation of the relevance scores of all the documents to all the queries as a whole.

Another related problem is how to obtain reliable scores for retrieved documents. One common linear score normalization method is: for any list of scores (associated with a ranked list of documents) for a given topic or query, we map the highest score into 1, the lowest score into 0, and any other scores into a value between 0 and 1 accordingly. This normalization method was used by Lee [8] and others in their experiments with TREC data sets. The above normalization method can be improved. In the TREC workshop, each system is usually required to submit 1000 documents for any given query. In such a situation, the top-ranked documents in the list are not always relevant, and the bottom-ranked documents are not always irrelevant. Therefore, [a, b]: ($0 < a < b < 1$) should be a more suitable range than [0,1] for score normalization [16]. This method is referred to as the fitting (linear score normalization) method later in this paper.

If only a ranked list of documents is provided without any scoring information, then we need to convert ranking information into scores. A common way of dealing with this is to assign a given score to documents at a particular rank. For example, Borda count [6] works like this: for a ranked list of t documents, the first document in the list is given a score of t, the second document in the list is given a score of $t - 1$, ..., the last document in the list is given a score of 1. Thus all documents are assigned corresponding scores based on their rank positions.

Table I
SUMMARY INFORMATION OF THE "BLOG06" TEST COLLECTION AND ITS CORRESPONDING STATISTICS

Quantity	Value
Number of Unique Blogs	100,649
RSS	62%
Atom First Feed Crawl Last Feed Crawl	21/02/2006
Number of Feeds Fetches	753,681
Number of Permalinks	3,215,171
Number of Homepages	324,880
Total Compressed Size	25GB
Total Uncompressed Size	148GB
Feeds (Uncompressed)	38.6GB
Permalinks (Uncompressed)	88.8GB
Homepages (Uncompressed)	20.8GB

III. EXPERIMENTAL SETTINGS AND RESULTS

In the TREC 2008 blog track, "Blog06" test collection was used. The summary information is shown in Table 1 [17].

Opinion retrieval is one of the tasks in the blog track. It is used to locate blog posts that express an opinion about a give target. A target can range from the name of a person or organization to a type of technology, a new product, or an event.

For most opinion retrieval systems, the opinion finding is a two-stage process. The first stage is to generate baseline ad hoc retrieval runs. 5 standard baselines were provided by NIST (National Institute of Standards and Technology, holder of TREC workshops) for the 2008 Blog track. Information about them can be found in [11]. Then, based on any of these baselines, the participants can submit their final runs by re-ranking a baseline run. 19 groups submitted a total of 191 runs to the opinion-finding task.

Each submitted run consists of up to 1000 retrieved documents for each topic. The retrieval units are the documents from the permalinks component of the Blog06 test collection. The content of a blog is defined as the content of the post itself and all the comments to the post.

Analogous to other TREC tracks, the blog track uses the pool policy for retrieval evaluation: pools were formed from the submitted runs of the participants. The two highest priority runs per group were pooled to depth 100. The remaining runs were pooled to depth 10. Only those documents in the pool are judged. All the documents that are not in the pool are treated as irrelevant documents.

In the experiment, two score normalization methods, Borda and the fitting method, are used. For the fitting method, 0.8987 and 0.0586 are used as the values for a and b, which are obtained from the observation of all the runs submitted. The data fusion methods involved are: CombSum, CombMNZ, LCP, LCP2, and LCR are tested. A total of 150 topics (851-950, 1001-1050) were used in the 2008 Blog track. We divide all 150 topics into three groups of equal size. Topics are put into groups in turn: topic 851 goes to group 1, topic 852 goes to group 2, topic 853 goes to group 3, and so on. One group (1, or 2, or 3) is used as training data to decide the weights for the linear combination method, two other groups (2 and 3, or 1 and 3, or 1 and 2) are used as test data. For all the data fusion methods involved, we randomly selected 5, 10, 15,..., 60 component systems from all available ones to test fusion effectiveness. For any given number, 200 combinations were carried out.

Four metrics are used for retrieval evaluation. They are: average precision over all relevant documents (AP), recall-level precision (RP), precision at 10 document level (P@10), and reciprocal rank (RR).

They are defined as:

$$AP = \frac{1}{R}\sum_{i=1}^{n}(rel(d_i) * p@i)$$
$$RP = \frac{p@R}{R}$$
$$P@10 = \frac{1}{10} * \sum_{i=1}^{10} rel(d_i)$$
$$RR = max_{i\geq 1}\{\frac{rel(d_i)}{i}\}$$

Here $rel(d_i) = 1$ if document d_i is relevant; 0 otherwise. R is the total number of relevant documents in the whole collection.

First let us use average precision to evaluate the experimental results. Tables 2 to 5 present the experimental results with two different score normalization methods. From Tables 2 to 5, we can see that, generally speaking, all data fusion methods are effective [5]. First let us look at the result using MAP (Tables 1-2). On average, all of them outperform the best component system. The smallest improvement rate over the best is 2.62% for CombMNZ with the fitting linear normalization method, and the largest improvement rate over the best is 10.26% for LCR with the fitting method. However, the number of component systems is an important factor that affects the performance of most data fusion methods significantly. When a small number of component systems are fused, all data fusion methods outperform the best component system by a clear margin. When the number of component systems is above a threshold, some data fusion methods become less effective than the best component system, though such a threshold varies considerably across different data fusion methods and score normalization methods. With all two normalization methods, CombMNZ is the worst, which followed by CombSum, LCP, and LCP2, while LCR is the best. Two tailed T test is also carried out to test the significance of the difference between any data fusion method and the best component system. If the difference is significant at the level of 0.95, then a "+" or "-" sign will be put as a superscript of the corresponding value. Figures on bold are the best in that line (setting). In 22 out of a total of 24 settings, LCR is the best; LCP2 is the best in the rest two settings.

Tables 4 and 5 present the results using MRR as the metric for retrieval evaluation. From Tables 4 and 5, we can see that for MRR, the impact of different normalization methods on the data fusion methods is stronger than that for MAP. When Borda normalization is used, LCR manages an improvement rate of 2.49% over the best component system. It is slightly worse than LCP (2.69%) and LCP2 (2.84%). But LCR is still the best when the fitting normalization method is used. When the fitting linear normalization method is used, CombMNZ becomes the second best data fusion method with an improvement rate of 2.35%.

Figure 1 shows the result of data fusion methods using RP, while Figure 2 shows the result of data fusion methods using P@10. In both Figures 1 and 2, for each data fusion method, only the best result is presented with one of the two score normalization methods.For both RP and P@10, all data fusion methods are better than the best component systems when a small number of results are fused. However, when a large number of results are fused, only LCR consistently outperforms the best component system.

In summary, one major observation from this study is: in most cases, the combination of the fitting method for score normalization and multiple linear regression for weights assignment is the most effective approach, especially when a relatively large number of component systems are fused.

[5]Each data value in Tables 2-5 and Figures 1-2 is the average of 200 randomly selected combinations $\times$ 100 queries per test set $\times$ 3 different test sets.

Table II
PERFORMANCE (MAP) OF ALL DATA FUSION METHODS (BORDA NORMALIZATION; LCP, LCP2, AND LCR DENOTE THE LINEAR COMBINATION METHOD WITH PERFORMANCE LEVEL WEIGHTING, PERFORMANCE SQUARE WEIGHTING, AND WEIGHTS DECIDED BY MULTIPLE LINEAR REGRESSION, RESPECTIVELY; FIGURES WITH "+" OR "-" INDICATE THEY ARE DIFFERENT (BETTER OR WORSE) FROM THE BEST COMPONENT SYSTEM SIGNIFICANTLY AT A CONFIDENCE LEVEL OF 95%; FIGURES ON BOLD ARE THE BEST IN THAT LINE)

Num.	Best	C'Sum	C'MNZ	LCP	LCP2	LCR
5	0.378	0.416^{+}	0.410^{+}	0.425^{+}	0.428^{+}	**0.441**$^{+}$
10	0.403	0.447^{+}	0.438^{+}	0.454^{+}	**0.458**$^{+}$	0.454^{+}
15	0.417	0.458^{+}	0.447^{+}	0.464^{+}	**0.467**$^{+}$	0.464^{+}
20	0.431	0.467^{+}	0.456^{+}	0.472^{+}	0.476^{+}	**0.478**$^{+}$
25	0.442	0.472^{+}	0.460^{+}	0.477^{+}	0.481^{+}	**0.484**$^{+}$
30	0.449	0.473^{+}	0.461^{+}	0.478^{+}	0.483^{+}	**0.490**$^{+}$
35	0.451	0.476^{+}	0.464^{+}	0.480^{+}	0.483^{+}	**0.489**$^{+}$
40	0.460	0.475^{+}	0.463	0.480^{+}	0.484^{+}	**0.490**$^{+}$
45	0.468	0.478^{+}	0.466	0.483^{+}	0.487^{+}	**0.495**$^{+}$
50	0.471	0.480^{+}	0.468	0.485^{+}	0.489^{+}	**0.497**$^{+}$
55	0.473	0.480	0.468	0.484^{+}	0.489^{+}	**0.494**$^{+}$
60	0.481	0.481	0.469^{-}	0.486^{+}	0.491^{+}	**0.501**$^{+}$
Ave.	0.444	0.467	0.456	0.472	0.476	**0.481**
		5.28%	2.77%	6.70%	7.42%	8.52%

Table III
PERFORMANCE (MAP) OF ALL DATA FUSION METHODS (THE FITTING LINEAR NORMALIZATION)

Num.	Best	C'Sum	C'MNZ	LCP	LCP2	LCR
5	0.378	0.417^{+}	0.412^{+}	0.423^{+}	0.425^{+}	**0.425**$^{+}$
10	0.403	0.447^{+}	0.440^{+}	0.451^{+}	0.453^{+}	**0.459**$^{+}$
15	0.417	0.455^{+}	0.448^{+}	0.458^{+}	0.461^{+}	**0.473**$^{+}$
20	0.431	0.462^{+}	0.456^{+}	0.465^{+}	0.468^{+}	**0.488**$^{+}$
25	0.442	0.467^{+}	0.460^{+}	0.470^{+}	0.473^{+}	**0.493**$^{+}$
30	0.449	0.467^{+}	0.460^{+}	0.470^{+}	0.473^{+}	**0.498**$^{+}$
35	0.451	0.468^{+}	0.462^{+}	0.470^{+}	0.473^{+}	**0.497**$^{+}$
40	0.460	0.468^{+}	0.462	0.471^{+}	0.474^{+}	**0.501**$^{+}$
45	0.468	0.471^{+}	0.464^{-}	0.473^{+}	0.477^{+}	**0.506**$^{+}$
50	0.471	0.472	0.466^{-}	0.474	0.477^{+}	**0.507**$^{+}$
55	0.473	0.472	0.466^{-}	0.474	0.478	**0.508**$^{+}$
60	0.481	0.473^{-}	0.466^{-}	0.475^{-}	0.479	**0.511**$^{+}$
Ave.	0.444	0.462	0.455	0.464	0.468	**0.489**
		4.08%	2.62%	4.69%	5.41%	10.26%

Table IV
PERFORMANCE (MRR) OF ALL DATA FUSION METHODS (BORDA NORMALIZATION)

Num.	Best	C'Sum	C'MNZ	LCP	LCP2	LCR
5	0.804	0.833^{+}	0.832^{+}	0.845^{+}	**0.849**$^{+}$	0.842^{+}
10	0.824	0.864^{+}	0.861^{+}	0.870^{+}	**0.872**$^{+}$	0.869^{+}
15	0.835	0.872^{+}	0.868^{+}	0.876^{+}	**0.877**$^{+}$	0.875^{+}
20	0.851	0.882^{+}	0.878^{+}	0.884^{+}	**0.885**$^{+}$	0.884^{+}
25	0.859	0.884^{+}	0.880^{+}	0.886^{+}	**0.888**$^{+}$	0.885^{+}
30	0.863	0.884^{+}	0.880^{+}	0.887^{+}	**0.888**$^{+}$	0.884^{+}
35	0.865	**0.888**$^{+}$	0.885^{+}	0.8871^{+}	0.888^{+}	0.885^{+}
40	0.871	0.886^{+}	0.881^{+}	0.887^{+}	0.**888**$^{+}$	0.884^{+}
45	0.877	0.888^{+}	0.883^{+}	0.888^{+}	**0.889**$^{+}$	0.885^{+}
50	0.880	0.888^{+}	0.884	0.889^{+}	**0.890**$^{+}$	0.887^{+}
55	0.881	**0.889**$^{+}$	0.882	0.888^{+}	0.888^{+}	0.888^{+}
60	0.888	0.890	0.884^{-}	0.884^{-}	**0.890**	0.887
Ave.	0.858	0.879	0.875	0.881	**0.883**	0.880
		2.44%	2.02%	2.69%	2.84%	2.49%

Table V
PERFORMANCE (MRR) OF ALL DATA FUSION METHODS (THE FITTING LINEAR NORMALIZATION)

Num.	Best	C'Sum	C'MNZ	LCP	LCP2	LCR
5	0.804	0.838^{+}	0.837^{+}	**0.840**$^{+}$	0.839^{+}	0.830^{+}
10	0.824	0.857^{+}	0.857^{+}	0.858^{+}	**0.858**$^{+}$	0.858^{+}
15	0.835	0.864^{+}	0.865^{+}	0.864^{+}	0.865^{+}	**0.870**$^{+}$
20	0.851	0.871^{+}	0.873^{+}	0.871^{+}	0.873^{+}	**0.882**$^{+}$
25	0.859	0.874^{+}	0.877^{+}	0.874^{+}	0.875^{+}	**0.885**$^{+}$
30	0.863	0.873^{+}	0.877^{+}	0.873^{+}	0.875^{+}	**0.891**$^{+}$
35	0.865	0.873^{+}	0.878^{+}	0.874^{+}	0.876^{+}	**0.886**$^{+}$
40	0.871	0.874	0.879^{+}	0.874	0.877^{+}	**0.890**$^{+}$
45	0.877	0.876	0.881^{+}	0.876	0.878	**0.892**$^{+}$
50	0.880	0.879	0.884^{+}	0.879	0.881	**0.891**$^{+}$
55	0.881	0.879	0.884	0.878	0.880	**0.890**$^{+}$
60	0.888	0.880	0.885	0.880^{-}	0.881^{-}	**0.896**$^{+}$
Ave.	0.858	0.870	0.873	0.870	0.871	**0.880**
		1.38%	2.35%	1.40%	1.54%	2.56%

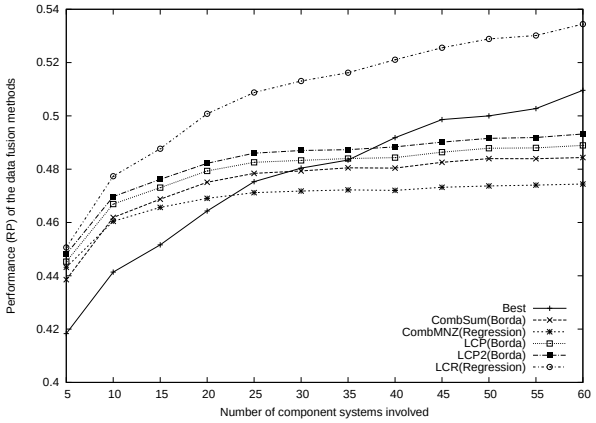

Figure 1. Performance (RP) comparison of different data fusion methods for each given number of component systems (for each data fusion method, only the best performance is presented with the corresponding score normalization method that is indicated in parentheses)

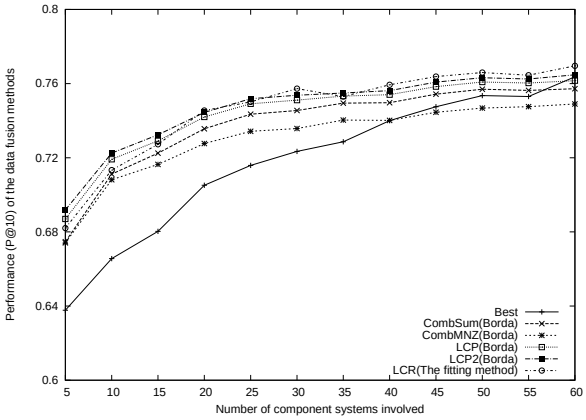

Figure 2. Performance (P@10) comparison of different data fusion methods for each given number of component systems (for each data fusion method, only the best performance is presented with the corresponding score normalization method that is indicated in parentheses)

On average, the improvement rate over the best component system is 10.26% for MAP, 4.51% for RP, 4.03% for P@10,

and 2.56% for MRR.

Apart from the above one, we also have some other observations as follows:

1) CombSum and CombMNZ are always close. Most of the time CombSum is a little better than CombMNZ. Sometimes the difference is significant, sometimes it is not.
2) With very few exceptions, LCP2 is always a little better than LCP. The difference between them is very often significant.
3) On average, LCP2 is the second best method in the experiment. It consistently outperforms the best component system when 35 or less systems are fused.
4) When a relatively small number (say, 5 or 10) of component systems are fused, then all data fusion methods outperform the best component system by a clear margin.
5) Compared with the best system, all the data fusion methods are more effective on average. However, the improvement rate varies when different metrics are used. The most favourable metric is AP, followed by RP and P@10, while RR is the least favourable.
6) For CombSum, CombMNZ, LCP, and LCP2, Borda is better; for LCR, the fitting method is better.

IV. Conclusion

In this paper we have presented a piece of work on data fusion to improve effectiveness of blog opinion retrieval. Extensive experimentation, with one large data set including all 191 runs submitted to the blog opinion task in TREC 2008, has been conducted and the results shows that on average, all data fusion methods involved are at least as good as the best component systems. Among them, the linear combination method with weights trained by multiple linear regression (LCR) and the linear combination with performance square weighting (LCP2) perform better than the others. This study demonstrates that data fusion can be a very good approach for us to develop effective blog retrieval systems.

References

[1] H. Du and C. Wagner, "Learning with weblogs: enhancing cognitive and social knowledge consstruction," *IEEE Transactions on Professional Communication*, vol. 50, no. 1, pp. 1–16, October 2007.

[2] Y. Huang, T. Huang, and Y. Huang, "Applying an intelligent notification mechanism to blogging systems utilizing a genetic-based information retrieval approach," *Expert Systems with Applications*, vol. 37, no. 1, pp. 705–715, January 2010.

[3] Y. Lin, H. Sundaram, Y. Chy, J. Tatemura, and B. Tseng, "Detecting splogs via temporal dynamics using self-similarity analysis," *ACM Transactions on Web*, vol. 2, no. 1, pp. 1–35, February 2008.

[4] M. Thewall and L. Hasler, "Blog search engines," *Online information review*, vol. 31, no. 4, pp. 467–479, 2007.

[5] O. Vechtomova, "Facet-based opinion retrieval from blogs," *Information Processing & Management*, vol. 46, no. 1, pp. 71–88, 2010.

[6] J. A. Aslam and M. Montague, "Models for metasearch," in *Proceedings of the 24th Annual International ACM SIGIR Conference*, New Orleans, Louisiana, USA, September 2001, pp. 276–284.

[7] M. Farah and D. Vanderpooten, "An outranking approach for rank aggregation in information retrieval," in *Proceedings of the 30th ACM SIGIR Conference*, Amsterdam, The Netherlands, July 2007, pp. 591–598.

[8] J. H. Lee, "Analysis of multiple evidence combination," in *Proceedings of the 20th Annual International ACM SIGIR Conference*, Philadelphia, Pennsylvania, USA, July 1997, pp. 267–275.

[9] S. Wu, Y. Bi, X. Zeng, and L. Han, "Assigning appropriate weights for the linear combination data fusion method in information retrieval," *Information Processing & Management*, vol. 45, no. 4, pp. 413–426, July 2009.

[10] S. Wu and S. McClean, "Improving high accuracy retrieval by eliminating the uneven correlation effect in data fusion," *Journal of American Society for Information Science and Technology*, vol. 57, no. 14, pp. 1962–1973, December 2006.

[11] I. Ounis, C. Macdonald, and I. Soboroff, "Overview of the trec-2008 blog track," in *Proceeding of the 17th Text Retrieval Conference*, Gaithersburg, MD, USA, 2008.

[12] E. A. Fox, M. P. Koushik, J. Shaw, R. Modlin, and D. Rao, "Combining evidence from multiple searches," in *The First Text REtrieval Conference (TREC-1)*, Gaitherburg, MD, USA, March 1993, pp. 319–328.

[13] C. C. Vogt and G. W. Cottrell, "Predicting the performance of linearly combined IR systems," in *Proceedings of the 21st Annual ACM SIGIR Conference*, Melbourne, Australia, August 1998, pp. 190–196.

[14] P. Thompson, "Description of the PRC CEO algorithms for TREC," in *The First Text REtrieval Conference (TREC-1)*, Gaitherburg, MD, USA, March 1993, pp. 337–342.

[15] S. Wu, "A geometric probabilistic framework for data fusion in information retrieval," in *Proceedings of the 10th International Conference on Information Fusion*, Quebec, Canada, July 2007, pp. 1–8.

[16] S. Wu, F. Crestani, and Y. Bi, "Evaluating score normalization methods in data fusion," in *Proceedings of the 3rd Asia Information Retrieval Symposium (LNCS 4182)*, Singapore, October 2006, pp. 642–648.

[17] I. Ounis, M. de Rijke, C. Macdonald, G. Mishne, and I. Soboroff, "Overview of the trec-2006 blog track," in *Proceeding of the 15th Text Retrieval Conference*, Gaithersburg, MD, USA, 2006.