

Necessary Constraints for Fusion Algorithms in Meta Search Engine Systems¹

Xiaohua Yang
Department of Computer Science
South-China University
Hengyang, Hunan 421001
China
Email: xhyang@ctit.zhnut.edu.cn

Minjie Zhang
Department of Computer Science and Software
Engineering
The University of Newcastle
NSW 2308, Australia
Email: minjie@cs.newcastle.edu.au

Abstract: Fusion is one of the main technical problems of running a meta search engine (MSE). In this paper, we have classified the potential cases of fusion in MSE systems (MSEs) and the types of MSEs. Based on these classifications, the necessary constraints of fusion methods in different fusion cases are described. This provides a reliable foundation for the construction of good fusion algorithms in MSEs.

Key words: World Wide Web, Meta Search Engine, Information Retrieval, Rank, and Fusion

1. Introduction

A search engine (SE) is an automatic robot, which indexes Internet data mainly by itself, allowing users to search the Internet via keyword-based queries. Altavista, Lycos, Excite, Hitbot and Infoseek etc. are examples of currently popular Internet search engines. Since there are a huge number of documents on the Internet, and each search engine indexes only a subset of the Internet documents, any single search engine cannot solve the problem of Internet information retrieval completely. In order to increase the recall and precision of queries, users may have to query several search engines.

A meta search engine (MSE) is an information retrieval agent which is built on the top of other search engines. Queries are submitted to the meta search engine, which in turn sends the query to multiple single search engines in parallel and then merges multiple results offered by different search engines. Metasearch, SavvySearch, Metacrawler, Profusion, Inquirus and MetaGer are examples of meta search engines.

The principle of a meta search engine can be described by the following steps, which are:

- (1) to accept a user query;
- (2) to convert the query into the correct syntax for every underlying search engine;
- (3) to launch the multiple queries;
- (4) to collect the results;

- (5) to merge the results; and
- (6) to deliver the postprocessed results to the client.

As we can see from the above steps, one of the main technical problems of running a meta search engine is the fusion of results from multiple search engines. Given a query, each underlying search engine will return a result, that is usually a subset of the final postprocessed result of a meta search engine. Since the number of hits of a search engine is often very large and users are subject to checking only the several top-ranked documents of the result, the order of documents in the final result is very important if users are to really get a significant benefit from the use of meta search engines.

A simple method of result merge, which is used by some meta search engines, is the one-after-the-other listing [6]. That is to strictly interleave the results, ordering the top-ranked by each of search engines first, followed by all of the second-ranked results, and so on. The difficulty is that there is no ordering among documents ranked equivalently by different sources.

The merge algorithm of MetaCrawler [6] is more sophisticated than the one-after-the-other listing. MetaCrawler uses a confidence score to determine how close a reference matches a query. A higher confidence score indicates a more relevant document. To calculate each

¹ Xiaohua Yang is currently a visiting fellow at the Department of Computer Science and Software Engineering, The University of Newcastle, NSW 2308, Australia. E-mail: xhyang@cs.newcastle.edu.au. This work is partly supported by RMC research grant (142/1078), University of Newcastle.

reference's confidence score, MetaCrawler first distributes the confidence scores returned by each engine into the range 0 to 1000. Thus, the top pick from each engine will have a confidence score of 1000. Then, MetaCrawler eliminates duplicates, and adds the removed reference's score to the sum of the duplicated references confidence scores. In essence, this allows engines to vote for the best reference, as a reference returned by many engines will most likely have a higher total score than a reference returned by only one.

Like MetaCrawler, Profusion [4] uses a merge algorithm that is also based on the original order of pages provided by search engines. Profusion uses a weighted score merge algorithm that is based on two factors: the value of the query-document match reported by the search engine and the estimated accuracy of that search engine. This kind of fusion has been found to be computationally simple yet as effective as a more expensive fusion [3].

Inquirus [5] uses a very different fusion method. In Inquirus, the actual pages of hits are downloaded and analyzed. Then a uniform ranking measurement is applied to documents returned by different engines. The meta search engine displays pages in descending order of relevance. It considers the number of query terms presented in the document, the proximity between query terms, and term frequency. The order of pages in the final result has no relationship with the original orders from underlying search engines.

The merge algorithm of MetaGer[2], which is the representative of another popular class of fusion method, is very different from that of MetaCrawler. In MetaGer, pages are ranked based not only on their original order relationships but also on word counts within the title, the URL and the description of the hits.

As we can see from the discussion above, the problem of fusion has been studied in MSEs broadly. It has also been analysed in other systems that search distributed information collections [1, 3, 7, 8]. Up to our best knowledge, however, an important issue has never been investigated in MSEs or other distributed information retrieval systems. That is, the necessary constraints that a rational fusion algorithm has to satisfy [9]. A lot of algorithms have been presented, and evaluated only based on experimental data. No reliable criterion is available for us to judge whether a

merge algorithm is good or bad, and no reliable methodology is available to construct a reasonable fusion method.

In this paper, we concentrate on the discovery of necessary constraints for fusion algorithms in meta search engines. In Section 2, the fusion problem is formally described. In Section 3, the potential fusion cases in MSEs are identified. In Section 4, the types of MSEs are classified and the relationships between fusion cases and MSE types is presented. In Section 5, necessary constraints of fusion algorithms in different cases are proposed. Finally, in Section 6, this paper is concluded with a brief discussion of further works.

2. Problem Description

A search engine is a ranked Internet information retrieval system. It can be formally defined as follows.

Definition 1. A search engine (SE) is a 4-tuple, $SE = \langle D, Q, r, t \rangle$, where D is the index database of Internet documents, Q is the set of valid queries, r is the rank algorithm and t ($0 \leq t$) is the threshold of result selection.

The indexes of a document can be different in different search engines. In this paper, however, we will pay no attention to formats of document indexes and will treat an index as a whole document object.

The sets of valid queries, which are parts of the capabilities of search engines, can be also different in different search engines. Keywords are the basic form of query that is valid in every search engine, but the sets of operators are varied with various engines. For example, a special operator NEAR, which is not available in many other search engines, can be used in the advanced search queries of Altavista. The expression $(q_1 \text{ NEAR } q_2)$ means that the distance between q_1 and q_2 must be less than 10 words.

The rank algorithm of a search engine can be formally defined as $r: D \times Q \rightarrow \mathbf{R}^+ \cup \{0\}$, where \mathbf{R}^+ is the set of positive reals. Given $\forall d \in D, \forall q \in Q, r(d, q)$ is the relevance of the document d to the query q . If the rank function satisfies condition: $\forall d \in D, \forall q \in Q, 0 \leq r(d, q) \leq 1$, then it is called a normal rank function. Since all rank functions can be transformed to equivalent normal functions, we will assume all rank functions are normal in the following discussion.

Given $\forall q \in Q$, the principle of a search engine can be described by 2 steps:

- (1) $\forall d \in D$, calculate its relevance with q .
- (2) Select all documents, which satisfy the condition $t \leq r(d, q)$, from D to compose the result.

The result of a query of a search engine is essentially an ordered list. Each item of the list corresponds to a document. The formats of items may be different in different search engines. In this paper, however, an item will be simply considered as a whole document object. Therefore, the result of a query q can be represented as $\langle d_1, d_2, \dots, d_m \rangle$, where d_j ($1 \leq j \leq m$) is a document of D , which satisfies two conditions:

- (1) $t \leq r(d_j, q)$, and
- (2) $\forall j, \forall l$, if $1 \leq j \leq l \leq m$, then $r(d_l, q) \leq r(d_j, q)$.

Definition 2. A meta search engine (MSE) is a 5-tuple, $MSE = \langle E_m, Q_m, F_m, r_m, t_m \rangle$, where E_m is the set of underlying search engines, Q_m is the set of valid queries, F_m is the set of query transformations, r_m is the rank algorithm and t_m ($0 \leq t_m$) is the threshold of result selection.

Suppose that E_m is composed of n ($2 \leq n$) search engines, SE_1, SE_2, \dots, SE_n , and F_m contains n functions, h_1, h_2, \dots, h_n . Then $\forall q \in Q_m$, given by the user, the query received by SE_i ($1 \leq i \leq n$) will be $h_i(q)$ ($1 \leq i \leq n$).

Suppose that the original result of query q of SE_i is $\langle d_i^1, d_i^2, \dots, d_i^{m(i)} \rangle$ and the final result of query q of MSE is $\langle d^1, d^2, \dots, d^m \rangle$, then

- (1) $\{ d^1, d^2, \dots, d^m \} = \bigcup_{i=1}^n \{ d_i^1, d_i^2, \dots, d_i^{m(i)} \}$,
- (2) $\forall k$ ($1 \leq k \leq m$), $t_m \leq r_m(d^k, q)$, and
- (3) $\forall k, \forall l$, if $1 \leq k \leq l \leq m$, then $r_m(d^l, q) \leq r_m(d^k, q)$.

The rank algorithm of a MSE is the fusion of all the underlying search engine's rank algorithms. The procedure of fusion can be formally defined as following:

- (1) If $n = 2$, then $r_m = f_m(r_1, r_2)$, where $f_m: R \times R \rightarrow R$ is a map called as the fusion algorithm of the MSE and R is the set of all rank algorithms.
- (2) If $n > 2$, $r = f_m(f_m(r_1, r_2), \dots, r_{n-1}), r_n)$.

The problem is to discover the necessary constraints of fusion strategies so that we can get good final results by fusion of original results.

3. Potential Fusion Cases in MSEs

In this section, we will classify the fusion cases in MSEs based on the relationships between results of each search engine. First, we will focus on the set relationships of the results, which leads to four kinds of basic fusion cases. Then, we will consider the order relationships of the results, which leads to another two further kinds of fusion cases.

Definition 3. Given a valid query q , if the document sets of results returned by underlying search engines are equivalent, then we will say that the fusion in the MSE, corresponding to the query q , is an equivalent case.

Definition 4. Given a valid query q , if there is a search engine SE_i which satisfies the following conditions:

- (1) $\forall SE_j \in E_m$ ($j \neq i$), the document set of the result returned by SE_j is equivalent with or included by that of the result returned by SE_i , and
- (2) $\exists SE_j \in E_m$ ($j \neq i$), the document set of the result returned by SE_j is a subset of that of the result returned by SE_i ,

then we will say that the fusion in the MSE, corresponding to the query q , is an inclusion case.

Definition 5. Given a valid query q , if the intersection of document sets of results returned by any two search engines is empty, then we will say that the fusion in the MSE, corresponding to the query q , is a disjoint case.

Definition 6. Given a valid query q , if the fusion of a MSE, corresponding to the query q , is neither an equivalent case, nor an inclusion case, nor a disjoint case, then we will say that it is an overlap case.

Example 1: Suppose SE_i and SE_j are the only two search engines in a MSE, q is a valid query given by a user, $R_i(q)$ and $R_j(q)$ are the document sets of results from SE_i and SE_j , respectively, then

- (1) If $R_i(q) = R_j(q)$, the fusion in the MSE, corresponding to the query q , is an equivalent case.
- (2) If $R_i(q) \subset R_j(q)$, the fusion in the MSE, corresponding to the query q , is an inclusion case.
- (3) If $R_i(q) \cap R_j(q) = \Phi$, the fusion in the MSE, corresponding to the query q , is a disjoint case.

- (4) If $R_i(q) \neq R_j(q)$, $R_i(q) \not\subseteq R_j(q)$, $R_j(q) \not\subseteq R_i(q)$, and $R_i(q) \cap R_j(q) \neq \Phi$, the fusion in the MSE, corresponding to the query q , is an overlap case.

Definition 7. Given a valid query q , if $\forall SE_i, SE_j \in E_m$, the proposition $\forall x, y \in R_i(q) \cap R_j(q) \rightarrow (r_i(x, q) \leq r_i(y, q) \leftrightarrow r_j(x, q) \leq r_j(y, q))$ is true, then we will say that the fusion in the MSE, corresponding to the query q , is a non-conflict case. Otherwise, the fusion is a conflict case. Where $R_i(q)$ and $R_j(q)$ are document sets of results returned from SE_i and SE_j , respectively, r_i and r_j are rank algorithms of SE_i and SE_j , respectively.

Theorem 1. A disjoint fusion case is a non-conflict case.

Proof: In a disjoint fusion case, $\forall q \in Q_m$, $R_i(q) \cap R_j(q) = \Phi$, so the proposition $\forall x, y \in R_i(q) \cap R_j(q) \rightarrow (r_i(x, q) \leq r_i(y, q) \leftrightarrow r_j(x, q) \leq r_j(y, q))$ is true.

In total, there are seven kinds of potential fusion cases in MSEs:

- (1) non-conflict equivalent case,
- (2) conflict equivalent case,
- (3) non-conflict inclusion case,
- (4) conflict inclusion case,
- (5) disjoint case,
- (6) non-conflict overlap case, and
- (7) conflict overlap case.

4. The Classification of Types of MSEs

Definition 8. Suppose $MSE = \langle E_m, Q_m, F_m, r_m, t_m \rangle$ is a meta search engine, $\forall SE_i = \langle D_i, Q_i, r_i, t_i \rangle$, $SE_j = \langle D_j, Q_j, r_j, t_j \rangle \in E_m$, and $\forall q \in Q_m$, let $R_i(q)$ and $R_j(q)$ be document sets of results from SE_i and SE_j , respectively.

- (1) If $\forall x, y \in D_i \cap D_j \rightarrow (r_i(x, q) \leq r_i(y, q) \leftrightarrow r_j(x, q) \leq r_j(y, q))$, then we will say that the rank algorithms of search engines are consistent in MSE.
- (2) If $(\forall x (x \in R_i \rightarrow (x \in D_j \rightarrow x \in R_j)))$ and $(\forall y (y \in R_j \rightarrow (y \in D_i \rightarrow y \in R_i)))$, then we will say that the result selection thresholds of search engines are consistent in MSE.

Definition 9. A *homogeneous MSE* is a MSE in which all of the SEs have the same index database, consistent rank algorithms and thresholds.

Definition 10. A *first class partially homogeneous MSE* is a MSE in which all of the SEs have consistent rank algorithms and consistent thresholds, but at least one SE has a different index database from others.

Definition 11. A *second class partially homogeneous MSE* is a MSE in which all of the SEs have the same index database and consistent rank algorithms, but at least one SE has an inconsistent threshold from others.

Definition 12. A *third class partially homogeneous MSE* is a MSE in which all of the SEs have consistent rank algorithms, but at least one SE has a different index database from others and at least one SE has an inconsistent threshold from others.

Definition 13. A *heterogeneous MSE* is a MSE in which at least one SE has an inconsistent rank algorithm from others, at least one SE has a different index database from others and at least one SE has an inconsistent threshold from others.

Definition 14. A *first class heterogeneous MSE* is a MSE in which at least one SE has an inconsistent rank algorithm from others and at least one SE has a different index database from others, but all of the SEs have consistent thresholds.

Definition 15. A *second class heterogeneous MSE* is a MSE in which at least one SE has an inconsistent rank algorithm from others and at least one SE has an inconsistent threshold from others, but all of the SEs have the same index database.

Definition 16. A *third class heterogeneous MSE* is a MSE in which at least one SE has an inconsistent rank algorithm from others, but all of the SEs have consistent thresholds and all of the SEs have the same index database.

Theorem 2. The fusion in a partially homogeneous MSE, corresponding to a valid query, is a non-conflict case.

Proof: In a partially homogeneous MSE, all of the SEs have consistent rank algorithms. According to Definition 8, $\forall SE_i, SE_j \in E_m$, the proposition $\forall x, y \in D_i(q) \cap D_j(q) \rightarrow (r_i(x, q) \leq r_i(y, q) \leftrightarrow r_j(x, q) \leq r_j(y, q))$ is valid. Since $R_i(q) \subseteq D_i(q)$ and $R_j(q) \subseteq D_j(q)$, $\forall x, y \in R_i(q) \cap R_j(q) \rightarrow (r_i(x, q) \leq r_i(y, q) \leftrightarrow r_j(x, q) \leq r_j(y, q))$ is true.

Theorem 3. The fusion in a second class partially homogeneous MSE, corresponding to a valid query, is either a non-conflict equivalent case or a non-conflict inclusion case.

Proof: In a second class partially homogeneous MSE, all of the SEs have consistent rank algorithms and the same index database. So, all results of SEs are sublists of a same list, which is composed of documents from the same index database. If all results of SEs contain the same document set, the fusion is a non-conflict equivalent case. Otherwise, the fusion is a non-conflict inclusion fusion case.

Theorem 4. The fusion in a homogeneous MSE, corresponding to a valid query, is a non-conflict equivalent case.

Proof: In a homogeneous MSE, all of the SEs have consistent rank algorithms. So, the fusion is a non-conflict case. Furthermore, all of the SEs have consistent thresholds and the same index database, so the fusion is an equivalent case.

5. Necessary Constraints for Fusion Methods in MSEs

5.1. General Necessary Constraints for All Fusion Methods in MSEs

The final result of a query in a MSE is based on the original results returned by underlying SEs, but not on the order of fusion. So, the following properties are the general necessary constraints which an acceptable fusion method in MSEs must satisfy.

- (1) The fusion method must satisfy the associative law. Suppose that R is the set of normal rank algorithms, f_m is a fusion method, then $\forall r_i, r_j, r_k \in R, f_m(f_m(r_i, r_j), r_k) = f_m(r_i, f_m(r_j, r_k))$.
- (2) The fusion method must satisfy the commutative law. That is, $f_m(r_i, r_j) = f_m(r_j, r_i)$.

Besides the general necessary constraints, there are some other specific necessary constraints, SNC for short, of fusion methods in each fusion case. We will discuss them in the following subsections.

5.2. Specific Necessary Constraints of Fusion Methods for Equivalent Case

Since fusion is an inductive process, we will now focus on the fusion of two rank algorithms. Suppose that $SE_i = \langle D_i, Q_i, r_i, t_i \rangle$ and $SE_j = \langle D_j, Q_j, r_j, t_j \rangle$ are the two SEs in a

$MSE = \langle E_m, Q_m, F_m, r_m, t_m \rangle$, q is the query given by a user, $R_i(q)$ and $R_j(q)$ are document sets of results from SE_i and SE_j , respectively, and $R(q)$ is the document set of the final fusion result. These notations will be used through all following discussions.

From a user's point of view, the most important messages of a query result are the documents contained in the results and the order relationships between these documents. Since the document set of the final result is the join of document sets of results from all underlying SEs, the principal property, which makes a fusion method acceptable, is the order relationship between documents.

In an equivalent fusion case, the document sets of results from underlying SEs are all equal, so it is critical for a fusion method to keep the order relationships of documents which are common in all results. Therefore, the specific constraint condition of fusion methods for equivalent case can be described as follows:

$\forall x, y \in R(q)$, if $r_i(x, q) \leq r_i(y, q)$ and $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 2: Given a query q , suppose that the original results from SE_i and SE_j are $\langle a, b, c \rangle$ and $\langle b, c, a \rangle$, respectively. Since the order between b and c is consistent in both original results, it must be kept in the final result. So, there are only 3 possible values for the final result: $\langle a, b, c \rangle$, $\langle b, c, a \rangle$ and $\langle b, a, c \rangle$.

5.3. Specific Necessary Constraints of Fusion Methods for Inclusion Case

Suppose $R_i(q)$ is a subset of $R_j(q)$, $R_i(q) \subset R_j(q)$, then $R(q) = R_j(q) = R_i(q) \cup R_j(q)$ and $R_i(q) = R_i(q) \cap R_j(q)$.

5.3.1. SE_i and SE_j have consistent rank algorithms

Suppose that x and y are any two documents of $R_i(q)$, if $r_j(x, q) \leq r_j(y, q)$ is true, then $r_i(x, q) \leq r_i(y, q)$ must be true. The reason is that SE_i and SE_j have consistent rank algorithms. An acceptable fusion method must keep the order relationships which are common in original results, so $r_m(x, q) \leq r_m(y, q)$ must be valid.

Suppose that x or y is a document that belongs to $R_j(q) - R_i(q)$ and $r_j(x, q) \leq r_j(y, q)$ is valid. If both x and y belong to the index database of SE_i , i.e. D_i , then $r_i(x, q) \leq r_i(y, q)$ is valid, because that SE_i and SE_j have consistent rank algorithms. Otherwise, either x or y is not an element of D_i , so SE_i has no knowledge about the order relationship between x and y . It is

decided by SE_j itself only. So, $r_m(x, q) \leq r_m(y, q)$ must also be true in this situation.

Therefore, while SE_i and SE_j have consistent rank algorithms, the fusion method must satisfy the following condition:

$\forall x, y \in R(q)$, if $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 3: Given a query q , suppose that the original results from SE_i and SE_j are $\langle b, c \rangle$ and $\langle a, b, c \rangle$, respectively. According to the above necessary constraint conditions, the final result must be $\langle a, b, c \rangle$.

5.3.2. SE_i and SE_j have consistent thresholds

For any document $x \in R_j(q) - R_i(q)$, x must not be an element of D_i , since SE_i and SE_j have consistent thresholds. So, the order relationship between x and any other document is totally determined by SE_j .

For any two documents $x, y \in R_i(q)$, the order relationship between x and y is determined by both SE_i and SE_j together. Therefore, while SE_i and SE_j have consistent thresholds, the fusion method must satisfy the following conditions:

- (1) $\forall x, y \in R(q)$, if x or $y \in R_j(q) - R_i(q)$ and $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$, and
- (2) $\forall x, y \in R_i(q)$, if $r_i(x, q) \leq r_i(y, q)$ and $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 4: Given a query q , suppose that the original results from SE_i and SE_j are $\langle c, b \rangle$ and $\langle b, c, a \rangle$, respectively. According to the above necessary constraint conditions, there are only 2 possible values for the final result: $\langle b, c, a \rangle$ and $\langle c, b, a \rangle$.

5.3.3. SE_i and SE_j have inconsistent rank algorithms and inconsistent thresholds

For any document $x \in R_j(q) - R_i(q)$ and $y \in R_i(q)$, (a) while $x \in D_i$, $r_i(x, q) \leq r_i(y, q)$ is true, because $r_i(x, q) < t_i \leq r_i(y, q)$ is true. (b) while $x \notin D_i$, SE_i has nothing to do with the order relationship between x and y . So, if $r_j(x, q) \leq r_j(y, q)$ is true, $r_m(x, q) \leq r_m(y, q)$ must be true.

Therefore, the constraint conditions that a rational fusion method must satisfy can be described as:

- (1) $\forall x \in R_j(q) - R_i(q)$, $y \in R_i(q)$, if $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$, and
- (2) $\forall x, y \in R_i(q)$, if $r_j(x, q) \leq r_j(y, q)$ and $r_i(x, q) \leq r_i(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 5: Given a query q , suppose that the original results from SE_i and SE_j are $\langle a, b \rangle$ and $\langle a, b, c, d \rangle$, respectively. According to the above necessary constraint conditions, there are only 2 possible values for the final result: $\langle a, b, c, d \rangle$ and $\langle a, b, d, c \rangle$.

5.4. Specific Necessary Constraints of Fusion Methods for Disjoint Case

5.4.1. SE_i and SE_j have consistent rank algorithms

For any x and y belonging to $R_j(q)$, if $r_j(x, q) \leq r_j(y, q)$, then (a) if both x and y belong to the index database of SE_i , i.e. D_i , $r_i(x, q) \leq r_i(y, q)$ is valid, because that SE_i and SE_j have consistent rank algorithms; (b) if either x or y do not belong to D_i , SE_i has no idea of the order relationship between x and y . So, $r_m(x, q) \leq r_m(y, q)$ must be true.

It is easy to get a similar conclusion for any x and y belonging to $R_i(q)$, so the fusion method must satisfy the following necessary constraint condition:

$\forall x, y \in R(q)$, if $(x, y \in R_j(q) \text{ and } r_j(x, q) \leq r_j(y, q))$ or $(x, y \in R_i(q) \text{ and } r_i(x, q) \leq r_i(y, q))$, then $r_m(x, q) \leq r_m(y, q)$.

5.4.2. SE_i and SE_j have consistent thresholds

For any $x, y \in R_j(q)$, x and y must not be elements of D_i , since SE_i and SE_j have consistent thresholds and $R_i(q) \cap R_j(q) = \Phi$. So, the order relationship between x and y is totally determined by SE_j . It is easy to get a similar conclusion for $x \in R_i(q)$ and $y \in R_i(q)$. Therefore, the necessary constraint conditions can be described as:

$\forall x, y \in R(q)$, if $(x, y \in R_j(q) \text{ and } r_j(x, q) \leq r_j(y, q))$ or $(x, y \in R_i(q) \text{ and } r_i(x, q) \leq r_i(y, q))$, then $r_m(x, q) \leq r_m(y, q)$.

Example 6: Given a query q , suppose that the original results from SE_i and SE_j are $\langle a, b \rangle$ and $\langle c, d \rangle$, respectively. According to the above necessary constraint conditions, there are 6 possible values for the final result: $\langle a, b, c, d \rangle$, $\langle a, c, b, d \rangle$, $\langle a, c, d, b \rangle$, $\langle c, d, a, b \rangle$, $\langle c, a, b, d \rangle$, and $\langle c, a, d, b \rangle$.

5.5. Specific Necessary Constraints of Fusion Methods for Overlap Case

5.5.1. SE_i and SE_j have consistent rank algorithms

For any x and y belonging to $R_j(q)$, if $r_j(x, q) \leq r_j(y, q)$, then (a) if both x and y belong to the index database of SE_i , i.e. D_i , $r_i(x, q) \leq r_i(y, q)$

is valid, because that SE_i and SE_j have consistent rank algorithms; (b) if either x or y do not belong to D_i , SE_i has no idea of the order relationship between x and y . So, $r_m(x, q) \leq r_m(y, q)$ must be true.

It is easy to get a similar conclusion for any x and y belonging to $R_i(q)$, so the fusion method must satisfy the following necessary constraint conditions:

- (1) $\forall x, y \in R_i(q)$, if $r_i(x, q) \leq r_i(y, q)$, then $r_m(x, q) \leq r_m(y, q)$, and
- (2) $\forall x, y \in R_j(q)$, if $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 7: Given a query q , suppose that the original results from SE_i and SE_j are $\langle a, b, c \rangle$ and $\langle b, c, d \rangle$, respectively. According to the above necessary constraint conditions, the final result must be $\langle a, b, c, d \rangle$.

5.5.2. SE_i and SE_j have consistent thresholds

For any $x \in R_i(q) - R_j(q)$ and $y \in R_j(q)$, x must not be an element of D_i , since SE_i and SE_j have consistent thresholds. So, the order relationship between x and y is totally determined by SE_j . It is easy to get a similar conclusion for $x \in R_i(q) - R_j(q)$ and $y \in R_i(q)$, so the necessary constraint conditions can be described as:

- (1) $\forall x, y \in R_j(q)$, if x or $y \in R_j(q) - R_i(q)$ and $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$,
- (2) $\forall x, y \in R_i(q)$, if x or $y \in R_i(q) - R_j(q)$ and $r_i(x, q) \leq r_i(y, q)$, then $r_m(x, q) \leq r_m(y, q)$, and
- (3) $\forall x, y \in R_i(q) \cap R_j(q)$, if $r_i(x, q) \leq r_i(y, q)$ and $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 8: Given a query q , suppose that the original results from SE_i and SE_j are $\langle b, c, a \rangle$ and $\langle c, e, d \rangle$, respectively. According to the above necessary constraint conditions, there are 3 possible values for the final result: $\langle b, c, a, e, d \rangle$, $\langle b, c, e, a, d \rangle$, and $\langle b, c, e, d, a \rangle$.

5.5.3. SE_i and SE_j have inconsistent rank algorithms and inconsistent thresholds

For any document $x \in R_i(q) - R_j(q)$ and $y \in R_i(q) \cap R_j(q)$, (a) if $x \in D_i$, $r_i(x, q) \leq r_i(y, q)$ must be true, because $r_i(x, q) < t_i \leq r_i(y, q)$ is true. (b) if $x \notin D_i$, then SE_i has nothing to do with the order relationship between x and y . It is easy to get a similar conclusion for any $x \in R_i(q) - R_j(q)$ and $y \in R_i(q) \cap R_j(q)$.

Therefore, the constraint conditions that a rational fusion method must satisfy can be described as:

- (1) $\forall x \in R_j(q) - R_i(q)$, $y \in R_i(q) \cap R_j(q)$, if $r_j(x, q) \leq r_j(y, q)$, then $r_m(x, q) \leq r_m(y, q)$,
- (2) $\forall x \in R_i(q) - R_j(q)$, $y \in R_i(q) \cap R_j(q)$, if $r_i(x, q) \leq r_i(y, q)$, then $r_m(x, q) \leq r_m(y, q)$, and
- (3) $\forall x, y \in R_i(q) \cap R_j(q)$, if $r_j(x, q) \leq r_j(y, q)$, if $r_j(x, q) \leq r_j(y, q)$ and $r_i(x, q) \leq r_i(y, q)$, then $r_m(x, q) \leq r_m(y, q)$.

Example 9: Given a query q , suppose that the original results from SE_i and SE_j are $\langle c, a \rangle$ and $\langle c, e, d \rangle$, respectively. According to the above necessary constraint conditions, there are 6 possible values for the final result: $\langle c, a, e, d \rangle$, $\langle c, a, d, e \rangle$, $\langle c, d, e, a \rangle$, $\langle c, d, a, e \rangle$, $\langle c, e, a, d \rangle$, and $\langle c, e, d, a \rangle$.

6. Conclusion

In this paper, we have identified the potential cases of fusion in MSEs and classified the types of MSEs. Based on these results, necessary constraints of fusion methods in different fusion cases are recognized.

Since the constraints of fusion methods in MSEs have never been described before, there is no proof that existing merge algorithms satisfy the necessary constraints completely. The performances of these algorithms are in doubt under some cases. We believe that in order to get a good fusion from multiple SEs in a MSE, people must carry out two steps. The first step is to apply the necessary constraints on the results returned by each underlying search engines and to decide the basic order relationships of documents. The second step is to use merge methods, such as those used in current popular MSEs, to determine any remaining order relationships which are uncertain. It will be part of our future work to construct practical merge algorithms based on these constraints and to assess their performances in real world applications.

Acknowledgement

Authors would like to thank Keith Nesbitt of the University of Newcastle for his comments which have been very helpful in improving the quality of this paper.

References

- [1] Bartell B. T., Cottrell G. W. and Belew R. K.. Automatic Combination of Multiple Ranked Retrieval Systems, *In Proceedings of the 17th International Conference on Research and Development in Information Retrieval (SIGIR-94)*, 173-181, 1994.

- [2] Beuermann W. S. and Schomburg M.. Internet Information Retrieval – The Further Development of Meta-Searchengine Technology, *In Proceedings of Internet Summit*, Internet Society, <http://www.uni-hannover.de/inet98/paper.html>, 1998.
- [3] Callan J. P., Lu Z. and Croft W. B.. Searching Distributed Collections With Inference Networks, *In Proceedings of the 18th International Conference on Research and Development in Information Retrieval (ACM SIGIR-95)*, 21-28, 1995.
- [4] Gauch S., Wang G. and Gomez M.. Profusion: Intelligent Fusion from Multiple, Distributed Search Engines, *Journal of Universal Computer Science*, 9(2): 637-649, 1996.
- [5] Lorence S. and Giles C. L.. Inquirus, the NECI meta search engine, *Computer Networks and ISDN Systems*, 30: 95-105, 1998.
- [6] Selberg E. W.. Towards Comprehensive Web Search, *Thesis of Doctor of Philosophy*, University of Wanshington, 61-64, <http://www.cs.washington.edu/homes/speed/home.html>, 1999.
- [7] Shaw J. A. and Fox E. A.. Combination of Multiple Searches, *In Proceedings of The Third Text Retrieval Conference (TREC-3)*, 105-108, 1994.
- [8] Voorhees E. M., Gupta N. K. and Laird B. J.. The Collection Fusion Problem, *In Proceedings of The Third Text Retrieval Conference (TREC-3)*, 95-104, 1994.
- [9] Zhang M. and Zhang C.. Potential Cases, Methodologies, and Strategies of Synthesis of Solutions in Distributed Expert Systems, *IEEE Transaction on Knowledge and Database Engineering*, 11(3): 498- 503, 1999.