Effective Rank Aggregation for Metasearching

Leonidas Akritidis, Dimitrios Katsaros and Panayiotis Bozanis

Department of Computer and Communication Engineering, Univarsity of Thessaly, Glavani 37, Volos, 38221, Greece

Abstract

Nowadays, mashup services and especially metasearch engines play an increasingly important role on the Web. Most of users use them directly or indirectly to access and aggregate information from more than one data sources. Similarly to the rest of the search systems, the effectiveness of a metasearch engine is mainly determined by the quality of the results it returns in response to user queries. Since these services do not maintain their own document index, they exploit multiple search engines by using a rank aggregation method in order to classify the collected results. However, the rank aggregation methods which have been proposed until now, utilize a very limited set of parameters regarding these results, such as the total number of the exploited resources and the rankings they receive from each individual resource. In this paper we present *QuadRank*, a new rank aggregation method, which takes into consideration additional information regarding the query terms, the collected results and the data correlated to each of these results (title, textual snippet, URL, individual ranking and others). We have implemented and tested *QuadRank* in a real-world metasearch engine, *QuadSearch*, a system developed as a testbed for algorithms related to the wide problem of metasearching. The name *QuadSearch* is related to the current number of the exploited engines (four). We have exhaustively tested *QuadRank* for both effectiveness and efficiency in the real-world search environment of *QuadSearch* and also, by using a task from the recent TREC-2009 conference. The results we present in our experiments reveal that in most cases *QuadRank* outperformed all component engines, another metasearch engine (Dogpile) and two successful rank aggregation methods, Borda Count and the Outranking Approach.

Keywords: Ranking, rank aggregation, rank fusion, metasearch, Borda Count, search engines, information search, information retrieval, Web.

1. Introduction

The lack of any specific structure and the vast amount of information published on the Web, makes it extremely difficult for the user to find the information s/he desires without any external help. As of February 2010, there are at least 19 general-purpose search engines [1], as well as numerous special-purpose search engines. Their population is mainly justified by two reasons: a) no ranking algorithm is broadly acceptable, although many users tend to consider *Google*'s ranking method as the most successful; b) no engine can achieve large coverage and high scalability. It is a common belief Sugiura and Etzioni (2000); Manning et al. (2008) that a single general

[1] See http://www.searchenginewatch.com

purpose search engine for all Web data is unrealistic, since its processing power cannot scale up to the rapidly increasing and unlimited amount of Web data.

The tool which rapidly gains acceptance among the users is *metasearch engines* Meng et al. (2002). These systems operate like a filter of the various crawler-based or directory-based search engines which they combine. Metasearch engines run simultaneously a user query across multiple *component search engines*, retrieve the generated results and then aggregate them. Finally, they present the best among them to the user.

The advantages of metasearch engines against search engines are significant Meng et al. (2002):

- They increase the search coverage of the Web, providing a higher *recall*. The overlap among the major search engines is usually very small Spink et al. (2006) and it can be as small as 3% of the total results retrieved. On the other hand, the unique results can be as high as 85% of the total results retrieved by all component engines.
- They solve the scalability problem of searching the Web and they facilitate the exploitation of multiple search engines enabling consistency checking Aslam and Montague (2001a).
- They improve the retrieval effectiveness providing higher precision, due to the "chorus effect" Vogt (1999).

Consequently, metasearch engines and their Web 2.0 successors, mash-up services are important tools and they are becoming increasingly popular. The core of any such system is the ranking function it employs, because this function defines the final ranked result list from the results provided by the component search engines. Hence, finding effective ranking algorithms is a problem of critical significance for metasearch engines and mash-up services.

The problem of rank aggregation is quite old and has been studied for a century, starting from a need to design fair elections. It can be thought of as the unsupervised analog to regression, with the goal of discovering a combined ranking which minimizes the distance to each individual ranking. Despite its seeming simplicity it is surprisingly complicated; finding the optimal combined ranking is NP-hard Dwork et al. (2001) under certain conditions. Thus, several recent efforts describe approximation algorithms for the rank aggregation problem Ailon et al. (2005); Ailon (2007); Coppersmith et al. (2006), after showing its relation to the *feedback arc set problem on tournaments* Ailon et al. (2005). Some of these are extensively applied to many different research domains, such as bioinformatics DeConde et al. (2006), Web spam detection Dwork et al. (2001), pattern ordering Tan and Jin (2004), metasearching Liu et al. (2007); Renda and Straccia (2003); Sculley (2007); Shokouhi (2007); Oztekin et al. (2002) and many more.

Web metasearching in contrast to rank aggregation, is a problem posing its own unique challenges. The results that a metasearch system collects from its component engines, are not similar to votes or any other single dimensional entities: Apart from the individual ranking it is assigned by a component engine, a Web result also includes a title, a small fragment of text which indicates its relevance to the submitted query (textual snippet) and a uniform resource locator (URL). Apparently, the traditional rank aggregation methods are insufficient for providing a robust ranking mechanism suitable for metasearch engines, because they ignore the semantics accompanying each Web result.

Based on these remarks, we conclude that ranking in Web metasearching is a more complex problem than rank aggregation. Individual rankings might be noisy, incomplete or even disjoint, hence they should not be the only parameter affecting ranking. Further processing is required

in order to filter the results and allow the final result list of the metasearch engine to be free of unwanted, devious and unfairly highly ranked Web pages. Since commercial interests might frequently and unpredictably affect the results of searching, the user is not clearly protected against the interests of individual search engines. Therefore, the ranking algorithm employed by a real metasearch engine, should be able to provide results that are as free as they can be from paid listings and links.

In this paper we propose *QuadRank*, a new rank aggregation method suitable for metasearch engines. QuadRank is a positional ranking method designed to deal with top-*k* lists returned by web search engines. Its main features are:

- It assigns scores to the candidate results by considering multiple parameters such as the number of the search engines where a particular item appeared, the total number of exploited search engines, the size of the top-*k* list returned by each search engine, the number of the occurrences of the query terms in each document, term proximity, zone scoring and others
- It refrains from using any training data in order to perform the rank aggregation, because, there is usually no evidence about the underlying data properties and their distributions
- It does not count upon the *scores* of the individual search engine rankings in order to perform the rank aggregation, because, most of the search engines do not provide such scores.

The new algorithm is evaluated on real world data drawn from four major search engines against individual search engines listings as well as results returned by metasearch engine Dogpile[2], using QuadSearch[3], a metasearch engine developed, among others, as a testbed for rank fusion. There is also an independent performance study of metasearch engines Allen (2009), comparing QuadSearch, Dogpile and Mamma, which showed that QuadSearch was the best of the three for that (limited) query load.

We also compare our proposed method to two other existing rank aggregation methods. The first is the well-established *Borda Count* method which assigns scores to the collected documents, by accumulating the individual rankings they received by the component engines. The second method is the *Outranking Approach*, an order-based method presented in Farah and Vanderpooten (2007), which orders the items by specifying a set of thresholds and by comparing each document with all the other collected documents. You can see section 2 for a brief description of these two methods and a discussion on their differences from our proposed algorithm.

Initially, we test these methods by utilizing the results from the Web Adhoc task of the Web Track of the TREC-2009 Conference Soboroff et al. (2009). In the sequel, we report the performance of the examined methods in the real-world environment of QuadSearch.

The rest of this article is organized as follows: in Section 2 we provide some necessary background material and survey the relevant rank aggregation methods. In Section 3, which presents the main article ideas, we describe the new rank aggregation method and the implementation issues behind the developed metasearch engine. In Section 4 we present an evaluation of the proposed method, and finally, Section 5 concludes the work.

[2] http://www.dogpile.com

[3] A publicly accessible prototype of QuadSearch is available under http://quadsearch.csd.auth.gr.

2. Preliminaries and relevant rank aggregation methods

We start with a universe U of items (documents in the context of metasearching); each item has a unique identifier c. A ranked list r of items $c_1, c_2, \ldots, c_n$ drawn from the universe U, is an ordered subset $S \subseteq U$, such that $r = [c_1 \geq c_2 \geq \cdots \geq c_n]$, where $\geq$ is an ordering relation on S. Each item $c \in S$, has the attribute $r(c)$ which represents the ranking of c in list r. Rankings are always positive, the best ranking an item could get is 1, and higher ranks show lower preference (reduced relevance to a query, in the context of metasearching).

If r contains all the items of U, then it is said to be a *full or complete list*; if $|r| < |U|$, then it is said to be a *partial list*, and if $|r| = k$, where k is a fixed constant, it is said to be a *top-k list*. Apparently, a top-k list is a special case of a partial list. The ideal scenario for rank aggregation is when each search engine gives a complete list of all the items of the universe related to the keyword terms of a given query. Unfortunately this is not possible since either each component engine has a partial coverage of the Web, or for reasons of speed or protection of the proprietary ranking algorithms, the engine returns only a top-k list. The worst but unusual scenario is when the result lists of component search engines have no overlapping elements. In this case there is nothing that a standard rank aggregation algorithm can do. However, as we will see later, QuadRank takes into account the metadata accompanying each item, in addition to the individual rankings of the search engines and this is an advantage of our method over the other methods.

Two families of rank aggregation techniques exist Renda and Straccia (2003): a) the *score-based* policies Vogt and Cottrell (1999), which assign a score to each entity of the individual ranking lists and then use these scores to perform the ranking, and b) the *order-based* (or *rank-based*) policies Dwork et al. (2001); Sculley (2007); Beg and Ahmad (2003), which work upon the order (rank) information that each entity received in the individual ranking lists. Although there have been proposed a lot of algorithms for rank aggregation, when it comes to applying these techniques to real-world metasearch engines the problem becomes even more complicated.

The methods belonging to the first category, were utilized by the first metasearch engines and they assign a weight to each item of a ranked list, which usually originates from the respective component search engine. The aggregation is performed using these scores; examples of this practice include the works in Vogt and Cottrell (1999); Aslam and Montague (2001b). Although almost no search engine provides the ranking scores, it is possible to convert local ranks into ranking scores. Although score-based methods appear to be more effective for rank fusion Renda and Straccia (2003), the absence of scores (or denial to reveal) from many search engines' rankings turns these methods problematic.

This shortcoming of the score-based algorithms lead to the generation of the second family of aggregation methods, i.e., the rank-based methods, the mainstream for modern metasearch engines. The methods of this family exploit only the relative position of the items in each ranked list to perform the fusion, thus they are also called *positional* methods. A primary advantage of positional methods is their efficiency in calculation, since they can be implemented in time linear w.r.t. the number and size of ranked lists. Unfortunately though, this efficiency in calculation is not accompanied by guarantees to satisfy the Condorcet criterion Young and Levenglick (1978).

A popular positional aggregation method is the Borda Count method Dwork et al. (2001); Renda and Straccia (2003), which assigns scores based on the positions of the item in the ranked lists. Using terminology from the voting literature, we can see each item of a ranked list as a candidate and each search engine as a voter. Each candidate receives points from each voter according to its rank in the voter's list. For example, the top ranked candidate will receive n

points, where n is the number of candidates in the respective ranked list. The total Borda score of the candidate will be the sum of its scores due to each ranked list where it appears. In case that the candidate is not in the top-k list of some voter, then it will receive a portion of the remaining points of the voter (each voter has a fixed number of points available for distribution) or a constant number (0 or 1), depending on the variation of the method Saari (2000). The Borda Count method can be found in different versions, like the weighted Borda Count method Souldatos et al. (2005), where each voter also takes a score and therefore his opinion for a candidate is not treated equally against other voters.

A relatively recent method is the Outranking Approach introduced by Farah and Vanderpooten (2007), which is based on decision rules identifying positive and negative reasons for judging whether a document should get a better rank than another. The method operates by performing pairwise comparisons of each item to all other items in the set S. If c_1 and c_2 are two documents of the set and $r(c_1)$, $r(c_2)$ are their rankings in the list r, then the item c_1 should be ranked higher than c_2 (symbolized as $c_1 \sigma c_2$) if the two following conditions are satisfied:

- The concordance condition which ensures that the majority of the input rankings are concordant with the ordering $c_1 \sigma c_2$. Formally, the concordance coalition is $C_{s_p}(c_1 \sigma c_2) = \{r(c_1) \leq r(c_2) - s_p\}$, where s_p is a preference threshold which determines the boundaries between an indifference and a preference situation between documents.

- The discordance condition which ensures that none of the discordant input rankings strongly refutes the ordering the ordering $c_1 \sigma c_2$. Formally, the discordance coalition is $D_{s_u}(c_1 \sigma c_2) = \{r(c_1) \geq r(c_2) + s_u\}$, where s_u is a veto threshold which determines the boundaries between a weak and a strong opposition to $c_1 \sigma c_2$.

Based on these conditions, a generic outranking relation is defined by the following formula

$$O(c_1 \sigma c_2) \Leftrightarrow |C_{s_p}(c_1 \sigma c_2)| \geq c_{min} \ AND \ |D_{s_u}(c_1 \sigma c_2)| \leq d_{max} \quad (1)$$

where c_{min} and d_{max} are the concordance and discordance thresholds respectively.

Other positional aggregation methods include the Markov chain based on Dwork et al. (2001), soft computing techniques Beg and Ahmad (2003), and median rank aggregation Fagin et al. (2003). The first methods "blend" the ranked lists into a Markov chain (MC), where each distinct item of the lists corresponds to a state of the MC and the transition probabilities correspond in some way to the (partial) ranked lists. The goal of this modeling is to find the stationary distribution vector of this MC, which provides a total ordering upon the states of the MC, and thus an ordering upon the items of the ranked lists. Unfortunately creating such a MC takes time $\Theta(n^2k + n^3)$, where n is the number of distinct items and k the size of the (top-k) lists; this computational cost can be reduced to $O(n^2k)$ to obtain a very rough approximation. Soft computing methods make use of genetic or fuzzy logic algorithms to perform the rank fusion, whereas median rank aggregation uses the media rank for each item to perform the final ranking.

Improved positional methods for rank aggregation of partial lists by exploiting the similarity among the items are described in Sculley (2007), whereas positional methods which are firstly trained and then used for rank aggregation (i.e., supervised rank aggregation) are described in Liu et al. (2007). Nevertheless, when the rank fusion method is going to be implemented as the heart of a metasearch engine, then fast computation of the aggregated rank and effective fusion are the major challenges to be met. Therefore, Markov chain based methods although claimed to be superior among the positional ones, are not the preferred choice over methods with linear time complexity computation cost.

2.1. QuadRank vs. Borda count

In this point, we must stress some differences between Borda Count and the scores of QuadRank.

- QuadRank is a rank aggregation method designed to operate on metasearch engines. Therefore, its scoring formula takes much more parameters into consideration such as the zone weighting, the domain characteristics and the number of the occurrences of each query term within each result.
- Some Borda Count variations assign scores to each and every candidate; a candidate which is not included in the top-k list of a particular search engine takes a part of the remaining points. This does not hold for the QuadRank method: a candidate will be assigned a score only when it is contained in the top-k list of a particular search engine, otherwise its score is zero.
- QuadRank also takes into consideration the total number of exploited search engines, the number of search engines where a candidate has been appeared and the number of items of each top-k list.
- The QuadRank method has better "resolution", in the sense that the possibility of two scores being the same is less than that of the Borda Count.
- Since our proposed method does not assign scores based completely on the individual rankings, it is more difficult to be "deceived" by a spam entry, that is a result which received an unfairly high ranking.

2.2. QuadRank vs. Outranking approach

In this subsection we describe the main differences between our proposed method and the Outranking approach.

- The Outranking approach takes into account only the individual rankings that each document received by the component engines. On the other hand, QuadRank takes much more parameters into consideration.
- The Outranking approach is an order-based method, therefore it is only based on comparison among the ranks only. It neither considers scores, nor hits whereas QuadRank assigns scores according to the ranking each item received by the component engines.
- The Outranking approach introduces four user-defined parameters (preference, veto, concordance and discordance thresholds). Selecting different values for these parameters can lead to significant modifications in the produced output ranking. In Farah and Vanderpooten (2007) it is shown that small changes in the values of these parameters lead to rankings with significant quality fluctuations. On the other hand, QuadRank involves settings of weights for each document zone (See Subsection 3.2 for more details).
- Since the Outranking approach requires a pairwise comparison of each item to every other collected item, it can turn significantly slow particularly when multiple long input lists are involved (because there are more items to be compared).

3. QuadRank

The default ranking fusion algorithm of *QuadSearch* is a positional method able to deal with partial ranked lists; actually it deals with top-k lists originating from single crawler-based search engines. The algorithm treats all component search engines equally. The reason we do this is due to the following plain, but significant observations: (i) all of them are considered by experts as the "major" search engines, (ii) they have been proved reliable during their lifetimes, and (iii) most users and metasearch engines prefer them to perform their searches on.

Let $r_1, r_2, \ldots r_m$ be m ranked lists corresponding to each component search engine. We assume that the each of these lists consists of a fixed number of k items, consequently, the entire process involves in total km elements that have to be merged and ranked. Merging is a procedure which we employ in order to combine the m different result lists into a single list, by removing all the overlapping elements. Notice that this list remains unranked until the scoring function is applied to each of the list's elements.

The overlapping between two component engines varies across different queries and cannot be predicted, hence we assume that our final result list consists of N items. In Table 1 we describe the symbols we use in our presentation and the parameters employed by QuadRank.

In the sequel we present the main ideas implemented by our proposed method. In particular we describe the methodology employed by QuadRank in order to deal with individual rankings and zone weighting and we examine the significance of the results' URL analysis. Finally, we discuss how all these different components can be combined together into a single scoring formula.

Symbol	Meaning
m	The number of exploited component engines
c	A single result (document) in a component list
$r_i(c)$	The ranking of c in the i^{th} component list
n_c	The number of component lists containing c
k	The number of items included in each component list
q	An arbitrary user query
t_q	A term of q
Q	The number of terms of q
N	The number of items included in the final merged list
N_{t_q}	The number of items containing t_q
$z_c(i)$	The i^{th} zone of c (see Table 3)
$W_{z_c(i)}$	The weight of $z_c(i)$
$f(c, t_q, z_c(i))$	The number of occurrences (frequency) of t_q in $z_c(i)$

Table 1: Summary

3.1. Dealing with individual rankings

The ranking that each element receives by the component engines is of primary importance for a rank aggregation method and the majority of the proposed ranking algorithms are mainly based on these rankings. Consequently, we must design our function in order to reward a result which achieves high rankings, since such entries are considered to be more relevant to a given query than others placed in lower positions.

Therefore, for each item $c \in S$ we introduce and evaluate the quantity

$$K(c) = \sum_{i=1}^{m} \Big(k + 1 - r_i(c)\Big) \tag{2}$$

where $r_i(c)$ is the ranking that the item is assigned by the i^{th} component engine. In the special case where the item c is not ranked by list r_i, we assume that $r_i(c) = k + 1$. Obviously, the best score an item can receive is km (if it is ranked first on every component list), whereas the lower score is 1 and it is assigned on a result which was ranked last only in one component list.

The introduced score rewards the items which received high rankings by multiple component engines, however it is relatively frequent that two or more results are assigned equal $K(c)$ values. For instance, consider the occasion where two different items c_1, c_2 receive the following rankings by four top-10 lists r_1, r_2, r_3 and r_4.

Item	r_1	r_2	r_3	r_4
c_1	1	-	-	-
c_2	7	7	10	10

Table 2: Example

In the example of Table 2 it holds $K(c_1) = K(c_2) = 10$. Nevertheless, we firmly believe that c_2 should be ranked higher than c_1 since:

- it appears in more input rankings, consequently it outperforms c_1 according to the democratic symmetry Saari (2000).
- it appears in more than half of the input rankings and hence, there is a smaller probability of being a spam entry according to the Condorcet criterion.

To handle such cases, we must reward the results which are considered as relevant to a given query by as many component engines as possible. If $n_c \leq m$ is the number of the component engines in which a result c occurs, then we introduce the $rank-based$ score which is determined by the following equation:

$$R(c) = mlog\Big(n_c K(c)\Big) \tag{3}$$

Where m is the total number of the exploited engines. Note that the logarithm in 3 is employed in order to reduce the deviation among the different values that $R_i(c)$ can receive. Additionally, although multiplying the logarithm with m makes no difference (since m is constant for all items), it is justified by our intention to assign the $R(c)$ score a larger value.

3.2. Zone weighting

Zone weighting is a well established methodology for ranking documents in traditional search systems. The main idea it is based on suggests partitioning each Web document into locations of special interest, namely *zones* or *fields*. Such partitioning is usually performed in structured or semi-structured documents (i.e. XML files), where the available information is distributed across multiple zones and represents different semantics (i.e. authorship, publication date, title, etc).

The introduction of zones allows us to compute scores by taking into consideration the physical location of a term within a document (i.e. in which field of an XML document a term

appears). An example of such ranking scheme is the BM25F function presented in Lu et al. (2006).

The main difficulty hidden behind the idea of using zone scoring in a rank aggregation method is that a Web result retrieved by a component search engine is not a complete document and only some limited representative information is provided to a metasearch engine. This information comprises of three individual semantics: the title, a small fragment of the document's text, called snippet, which indicates the relevance of the document to the user query and a uniform resource locator (URL).

Zone	ID	Weight
Title	1	10
Snippet	2	3
URL	3	5

Table 3: Zones

In this paper we attempt to integrate zone scoring characteristics to our ranking function. The idea we introduce here is to treat the three aforementioned semantics as separate document zones. Although zone weighting is not a new scheme in information retrieval, to the best of our knowledge we are the first to apply its principles in the field of rank aggregation. Our motives are multiple: A Web result which contains the query terms on its title is possibly more relevant than another which does not. This is also valid for results containing the query terms in their snippets more times than others. Hence, the number of the occurrences of the query terms in the individual zones is another parameter which should be taken into consideration by an efficient ranking method.

As we have mentioned, there already exists a zone scoring scheme, BM25F. Nevertheless, we found that this method is not suitable for our case since:

- The BM25F function requires the length of each zone (in number of terms) to be computed, which undoubtedly is a time consuming operation.
- The second reason that turns the usage of zone lengths problematic is that the Web results returned by the component engines are not complete documents. The small fragments of text used to represent the similarity of each document to the submitted query, have similar or identical lengths for each entry. Consequently, zone lengths have a small contribution to the score of an item.
- It depends on several (typically three) user-defined parameters.

In this Subsection we propose our own policy for zone scoring and we believe that our suggestions are more suitable for metasearch applications. At first, in Table 3 we determine the weights assigned to each zone. Based on these weights, we use the following formula to compute a weight factor:

$$Z(c) = \sum_{t=1}^{Q} log\frac{N}{N_t} \sum_{z=1}^{3} W_z f(c,t,z) \tag{4}$$

where N represents the total number of items included in the final merged list and N_t is the number of items containing the query term t. Furthermore, W_z denotes the constant weight of a

zone, which we show in Table 3, whereas $f(c,t,z)$ represents the frequency of the query term t within zone z.

The logarithm $logN/N_t$ is drawn from the traditional tf/idf scoring functions used by search engines to rank Web documents (i.e. BM25, BM25F). It is used to reward the documents containing the query terms appearing a few times only, because such terms are expected to reveal the information need hidden behind each query.

Note that the $Z(c)$ score rewards the documents which include the query terms on their title, snippet or URL as many times as possible since it is sensitive to the corresponding frequency values. Consequently, the documents which include all or some of the query terms in their title multiple times are considered more relevant to the given query and they are assigned higher scores.

To compute the desired frequency values, the text accompanying each result must be appropriately processed. Therefore, each document is tokenized (that is, we obtain all of its distinct terms) and each of the extracted distinct terms passes multiple filters which sequentially perform punctuation removal, case folding and stemming. These are CPU-intensive procedures and one would expect query processing to decelerate significantly. However, the small document population as well as their limited size (titles and snippets consist of a few terms only) render this delay rather inconsequential (see experimental subsection 4.3).

3.3. URL analysis

To the best of our knowledge, the current publicly known rank aggregation methods do not take into consideration the URLs of the results that the component engines return. In this subsection we attempt to mine the information revealed by the URLs, in order to improve the quality of the produced results.

Since a metasearch engine is a system which exploits multiple resources, it is possible that several results under the same domain name would appear in the final merged list. Although these entries have different URLs and probably include different content, their origin is identical. This observation forces us to conclude that this domain possibly contains adequate information relevant to the given query. Consequently, we should enhance our ranking function in order to reward such results.

On the other hand, a result list comprised of different items from the same resource is hardly informative. An effective search system must provide qualitative results from many sources, because this wide variety partially ensures that the user can locate the desired information. For this reason, the major crawler-based search engines include at most two documents with the same domain name in their result lists, even if there are additional relevant documents.

The problem is now becoming straightforward: from a pool of X documents having the same domain name we must identify and reward the best two among them, whereas we should discard the rest $X-2$ of them. The challenge becomes even greater if maximum efficiency is required and fast system response is a key issue. QuadRank solves the problem by employing an auxiliary table of *domain accumulators* which it populates during the component engines' result list merging. When an item enters the list, we also search for its domain name into this accumulator table. If the record is not present we insert it and set the corresponding accumulator value equal to 1. In the opposite case we increase the accumulator by one.

During the scoring process we consult the accumulator table and we assign additional scores according to the following formula:

$$u(c) = log\left(10\frac{2m - 1 + acc_c}{2m}\right) \tag{5}$$

where acc_c is the domain accumulator of c. Later, when the results are going to be presented to the user, only the two highest scored results with the same domain name are used; the rest of them are simply discarded. Note that for the results having domain names appearing only once, it holds that $acc_c = 1$ and the quantity $U(c)$ is assigned a value equal to 1.

The analysis of a result's URL can lead to additional information regarding the page it represents. More specifically, the two or three trailing characters of the domain name (also known as the domain extension), usually reveal the originating country of that page.

Additionally, a significant number of queries submitted to a search engine is directly connected to a specific geographic region. For instance the travel, vacation and news oriented searches are cases falling into this category. When such information is required, there is a significant possibility that the pages hosted under affiliate domain extensions are the most satisfactory.

To address such issues, we introduce an expansion to our ranking function, the *GeoFactor*, a parameter which is determined by the relationship between the geographic locality of the user and the proximity of each result c. The Geo Factor receives values according to the following formula:

$$G(c) = \begin{cases} 1 & \text{if domain extension and user locality are not identical,} \\ \lambda, \lambda > 1 & \text{otherwise.} \end{cases}$$

where λ is a predefined constant receiving values $\lambda > 1$; a typical option which we have used in our experiments is to set $\lambda = 1.2$.

By integrating the Geo Factor into the scores of equation 5 we introduce the *URLAware* scores determined by the following formula:

$$U(c) = G(c)log\left(10\frac{2m - 1 + acc_c}{2m}\right) \tag{6}$$

The lowest value that $U(c)$ can receive is 1, for results appearing under a domain name encountered only once in the m component lists and when this domain does not match the geographic locality of the user. On the other hand, if we assume that each component engine returns at most 2 results having the same domain name, then the largest value that $U(c)$ can be assigned is $\lambda log\big((20m - 5)/m\big)$.

3.4. QuadRank scores

In this subsection we merge all the aforementioned components into a single scoring function, namely QuadRank. According to our proposed method, each collected result is assigned a score which is determined by the following formula:

$$Q(c) = U(c)\left(R(c) + \frac{1}{Q}Z(c)\right) \tag{7}$$

where Q is the total number of the query terms. As one may notice from equation 4, the $Z(c)$ score representing our zone weighting policy, is strongly depended on the number of query terms and increases proportionally to Q. Consequently, to regulate the contribution of zone weighting to the final scores, with respect to the other terms $R(c)$ and $U(c)$, we introduce the quantity $1/Q$.

4. Evaluation of the proposed method

The most important measure of a search system's performance is the quality of its search results. Quality is a decision made after the results evaluation by one or more human users. Moreover, it should not be assumed that quality is absolute; one user may well judge that a result is qualitative, while another says it is not.

In addition, quality is usually identified with the relevancy of the returned results to a given query. Every query represents an information need; the user who submits it, seeks for information that is somehow related to the terms of the query. Consequently, a document is judged to be relevant only if it addresses the stated information need, not because it happens to contain all the words of the query.

To evaluate the performance of a metasearch engine, a basic difference from search engines must be considered. A metasearch engine does not maintain its own document collection. It neither employs crawlers, nor indexers to construct a repository and an underlying index structure. Its output is based on the results that its component engines return. It does not even retrieve the entire set of these results; it works only with the first parts of this set, as these parts are the most promising to contain the best answers to a specified query. Therefore, in the extreme situation where all component engines send irrelative results, there is nothing a metasearch engine can do. But as we show later, a metasearch engine having an effective ranking fusion method, can mine the individual result sets and fuse them to a list that ranks the best items at the top.

A number of methods have been suggested for evaluating the effectiveness of a search system and its ranking algorithm. None of these methods are entirely satisfactory, but this is a natural consequence of attempting to represent multidimensional behavior with a single representative value.

To ensure a fair evaluation of QuadRank against its competitors in the context of retrieval effectiveness, we divide our experimentation into two phases: At first, we use data from the TREC-2009 conference which apart from the result sets, it also provides a list containing a set of queries and the corresponding relevant documents.

In the sequel, we measure the performance of the involved algorithms in the real-world environment of QuadSearch. For this series of experiments there is no ground truth regarding the relevancy of each document to our test queries. For this reason, we asked from six of our colleagues to judge the relevancy of each document and in our analysis we consider a document to be relevant if and only if more than half (four or more) of our colleagues judged its relevance positively.

4.1. Retrieval effectiveness evaluation with TREC data

The first phase of our experiments includes the application of our proposed method to the Web Adhoc (WA) Task of TREC-2009 Web Track Soboroff et al. (2009). This task contains 50 topics (test queries) and 72 participating teams. For each query, each team provides a ranking of about 1000 documents which is later evaluated by using a separate file containing all the documents that are relevant to the given topic. The performances of the 10 best runs of the WA task are reported in Table 4.

For evaluation, we used the 'trec_eval' standard program utilized by the TREC community in order to calculate several measures indicating the retrieval effectiveness of a system. These measures are Mean Average Precision (*MAP*), *R*-Precision and Precision@n ($P@n$) for $n = 5, 10$ and 20.

Run	*MAP*	*P*@5	*P*@10	*P*@20	*R*-Precision
uvamrftop	10.2%	41.6%	37.4%	28.8%	13.6%
UMHOOsd	10.2%	34.8%	37.2%	35.4%	17.5%
UMHOOsdp	10.2%	34.8%	37.2%	35.4%	17.5%
NeuLMWeb300	10.0%	44.8%	44.2%	38.0%	16.7%
NeuLMWeb600	9.8%	39.6%	39.8%	36.1%	16.9%
uogTrdphCEwP	9.8%	42.4%	37.4%	32.1%	15.4%
UMHOObm25B	9.8%	34.8%	36.4%	34.3%	16.8%
WatSdmrm3we	9.5%	16.0%	16.4%	16.6%	14.4%
udelIndDMRM	9.4%	26.0%	31.8%	33.1%	17.0%
udelIndDRSP	9.4%	28.0%	32.8%	31.9%	15.8%

Table 4: Performances of the 10 best runs of the WA task of TREC-2009.

QuadRank is compared against the other two rank aggregation algorithms as well as against some high performing official results from TREC-2009. For each topic, we retrieve one list from each participating team, that is 72 rankings. From these rankings we retain only the first k documents and in the sequel, the 72 top-k lists are merged into one large list which is finally ranked by employing QuadRank, Borda Count and the Outranking approach. In the following tables we present results for different values of k.

To measure the performance QuadRank, it was necessary retrieve the full text of each candidate document in order to compute the $Z(c)$ scores. Since all the candidate documents are from the Clueweb09 dataset, we have developed the appropriate software which operates on the collection and extracts the desired information. $Z(c)$ scores are computed by considering the document's title and URL only, because of the absence of textual snippets. Furthermore, the locality of the user is unknown and the geographical relationship between the user and a document can not be determined. For this reason, we have disabled the Geographic extensions of the $U(c)$ scores in this experimental phase.

Regarding the setting of the parameters introduced by the Outranking approach, we used the same values as those mentioned by Farah and Vanderpooten (2007). Therefore, we considered that each input ranking is a complete order ($s_p = 0$) and that an input ranking strongly refutes an ordering between two documents when the difference of both document positions is large enough ($s_u = 75\%$). We also supposed that the majority of the rankings must be concordant ($c_{min} = 50\%$) and that every input ranking can impose its veto ($d_{max} = 0$).

Table 5 shows the performance of the three examined rank aggregation methods in the WA task for variable number of retained documents. If we consider Mean Average Precision (MAP) as the comparison measure between the three methods, it is apparent that QuadRank outperforms both Borda Count and the Outranking Approach for all values of k. It is also remarkable that Borda Count performs better than the Outranking Approach for small values of k, whereas the situation is reversed as k increases.

Regarding the average Precision values at cut off points 5, 10 and 20, the performance of the three methods varies significantly. For instance, QuadRank achieved the highest $P@10$ values for $k = 500$ and $k = 1000$, but for smaller values of k the Outranking Approach becomes the winning method. Furthermore, for $k = 100$ QuadRank achieves the highest $P@20$ value whereas for $k = 200$ the Outranking Approach fetched the most qualitative top-20 list.

Run	*MAP*	*P*@5	*P*@10	*P*@20	*R*-Precision
Best TREC Run	10.2%	41.6%	37.4%	28.8%	13.6%
Borda Count (30)	18.5%	39.2%	37.8%	33.8%	26.0%
Outranking Approach (30)	15.5%	42.0%	38.4%	36.8%	22.2%
QuadRank (30)	19.5%	43.2%	39.2%	36.4%	25.4%
Borda Count (100)	19.0%	44.0%	42.8%	41.1%	23.4%
Outranking Approach (100)	18.4%	50.0%	48.0%	40.5%	25.1%
QuadRank (100)	20.9%	46.4%	43.8%	42.3%	25.0%
Borda Count (200)	18.8%	47.6%	44.2%	42.4%	23.1%
Outranking Approach (200)	18.8%	51.6%	47.6%	44.9%	24.3%
QuadRank (200)	20.6%	49.8%	45.9%	44.7%	24.0%
Borda Count (500)	18.2%	50.4%	47.8%	42.1%	22.9%
Outranking Approach (500)	18.7%	50.4%	47.4%	41.3%	23.7%
QuadRank (500)	19.8%	50.4%	48.2%	42.6%	23.4%
Borda Count (1000)	17.3%	49.6%	48.0%	42.7%	22.0%
Outranking Approach (1000)	18.2%	50.8%	45.8%	39.4%	22.0%
QuadRank (1000)	19.9%	51.2%	49.0%	43.1%	21.4%

Table 5: Performance of different rank aggregation methods for $m = 72$ and varying number of retained documents.

4.2. Retrieval effectiveness evaluation with test queries

To examine the retrieval effectiveness of the new ranking algorithm, we have integrated its implementation within *QuadSearch*, our experimental metasearch engine. Various queries were submitted to the system and the results' lists that the proposed ranking fusion methods returned are recorded and analyzed in this subsection. For all tests demonstrated here we exploited four major component engines (Google[4], Yahoo![5], Live[6] and Ask[7]). The system dispatched our submitted queries to each of these component engines and requested top-30 lists to be returned. Consequently, for the setup that we examine here it holds that $m = 4$ and $k = 30$.

Let us suppose that a user searches for tickets for the forthcoming UEFA Champions League final game and thus, he/she phrases and submits the query *"tickets for uefa champions league final 2010"*. The informational need is overt in this query. As mentioned above, a document not only has to contain the query terms, but also to address the stated information need to be considered as relevant. For the specified example query, a document is relevant only when:

- it is relevant to the *"uefa champions league"* and
- it provides information about *"tickets"* and
- is about the *final* game that will be conducted in *"2010"*.

[4]http://www.google.com
[5]http://search.yahoo.com
[6]http://www.live.com
[7]http://www.ask.com

	URL	R	G	Y	L	A	BC	OA
1	www.championsleagueticketservice.com	R_1	1	1	-	6	3	5
2	www.championsleagueticketshop.com	R_2	15	3	5	11	2	2
3	www.worldticketshop.com	-	6	-	14	-	16	15
4	en.wikipedia.org	R_3	8	-	1	-	7	10
5	www.soldoutentertainments.com	R_4	10	2	7	7	1	1
6	www.1st4footballtickets.com	R_5	-	-	13	-	45	62
7	www.uefa.com	-	3	-	9	-	9	14
8	www.1st4footballtickets.com	R_6	-	-	3	-	28	61
9	www.livefootballtickets.com	R_7	16	-	10	-	18	13
10	www.championsleagueticketservice.com	-	2	-	-	-	10	78
11	www.nwtix.com	R_8	29	-	22	-	23	21
12	www.roadtrips.com	R_9	4	4	12	-	4	3
13	www.globalticketshop.com	R_{10}	-	-	16	-	31	51
14	www.uefa.com	-	-	-	6	4	8	9
15	www.1st4footballtickets.com	R_{11}	-	-	-	3	29	27
16	www.ticketcity.com	-	12	11	8	-	5	4
17	soccerlens.com	-	14	-	11	-	17	11
18	www.webuytickets.net	R_{12}	23	-	-	-	27	81
19	www.livefootballtickets.com	-	-	24	26	-	19	20
20	www.freetickets.org.uk	R_{13}	5	9	-	-	15	17

Table 6: Top-20 list and relevant documents for the query *"tickets for uefa champions league final 2010"* when the *QuadRank* algorithm is applied.

All documents containing information about the *"uefa cup"* or past *"uefa champions league finals"* (e.g. 2008, 2009) are considered as irrelevant, since they do not satisfy the user's information need.

QuadSearch requested and received 120 results (thirty results from each component engine). After the merging of the results' lists, a set of candidates comprised of 88 unique items is constructed.

The top-20 list that *QuadRank* produced for this query, is displayed in ranked order in Table 6. The first column denotes the QuadRank ranking, whereas in the second column we choose to display only the domain name of the returned result, to form a compact and legible table. The next column shows which of the documents are relevant to the given query (with respect to the judgement made by our colleagues) and the symbol R_i is used to indicate the i^{th} relevant document of the QuadRank's top-20 list. The next four columns signify the ranking that the component engines gave to this document[8]. The dash symbol represents the absence of this result from an engine's top-20 list. In the last two columns we compute and display the ranking generated by the Borda Count method and the Outranking approach (column headers *BC* and *OA* respectively), when they are applied on the same result set.

A significant number of results that the search engines returned, were about past *"champions league finals"*. Particularly we found that all component lists contained results regarding the final game of 2009 in their top-10 lists. Some other entries were about tickets for the uefa cup final,

[8]Column headers: *G* for Google, *Y* for Yahoo!, *L* for Live Search and *A* for Ask

Engine	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**R**
QuadRank	R	R	-	R	R	R	-	R	R	-	7
Google	R	R	-	R	-	R	-	-	R	R	6
Yahoo!	R	R	R	R	-	R	-	R	-	-	6
Live Search	-	R	R	-	R	-	R	-	-	R	5
Ask	-	R	R	-	-	R	R	-	-	-	4
Dogpile	S	S	S	R	-	-	-	R	R	-	3
Borda Count	R	R	R	R	-	R	-	-	-	-	5
Outranking Approach	R	R	R	-	R	R	-	-	-	R	6

Table 7: Relevant Documents in the Top-10 Lists for the Query *"tickets for uefa champions league final 2010"*.

whereas others included no information about tickets at all. All these results were considered as irrelevant.

Now let us examine the first ten results returned by each algorithm. Ask performed poorly in this query, since it produced only four relevant documents in its top-10 list, whereas Google and Yahoo were more accurate as they returned six relevant documents. On the other hand, the top-10 ranking that QuadRank produced was the most qualitative, since it included seven relevant results. Note that QuadRank achieved to construct a result list which is improved compared to the rankings of the combined component engines. These notations are all summarized in Table 7. The symbol *R* is used to signify a relevant document, the dash symbol is used to mark the irrelevant ones, whereas *S* is used for sponsored entries. The last column shows the total number of relevant documents returned by each component engine or ranking algorithm in their top-10 list.

In the same Table we present the top-10 list provided by the most widespread metasearch engine, Dogpile. The top three results are sponsored links found on Google Ads [9], an online advertisement service maintained by Google. The fourth result which is organic, is relevant to the query, but it leads to the same location as the second result. It is clear that Dogpile's policy to include paid listings within organic results leads to a confusing top-10 list, which contains different results leading to the same location. In total, Dogpile's top-10 list for this query contains three paid and only seven organic results. From these seven results, only three are relevant to the query, as the other four concern past *"Champions League finals"*.

In Figure 1 we illustrate the effectiveness of the search engines which we examine in our experiments. The two diagrams depict the precision values measured for the first 10 and 20 results respectively. QuadRank outperformed all of its opponents, since it produced the most qualitative result list compared to the component engines, Dogpile, Borda Count and the Outranking approach

The QuadRank algorithm outperformed all of its components including the component engines, Dogpile and the Borda Count and Outranking methods, since it achieved the highest precision values for both the top-10 and top-20 lists. Google, Yahoo and Dogpile constructed top-10 lists of equal quality, whereas the latter's ranking algorithm performed better than the component engines. Regarding Borda Count, the top-10 and top-20 lists included five and eleven relevant results respectively, consequently, our algorithm presented more qualitative results. The Outranking approach was slightly more effective than Borda Count but it was also outperformed by

[9] https://www.google.com/adsense

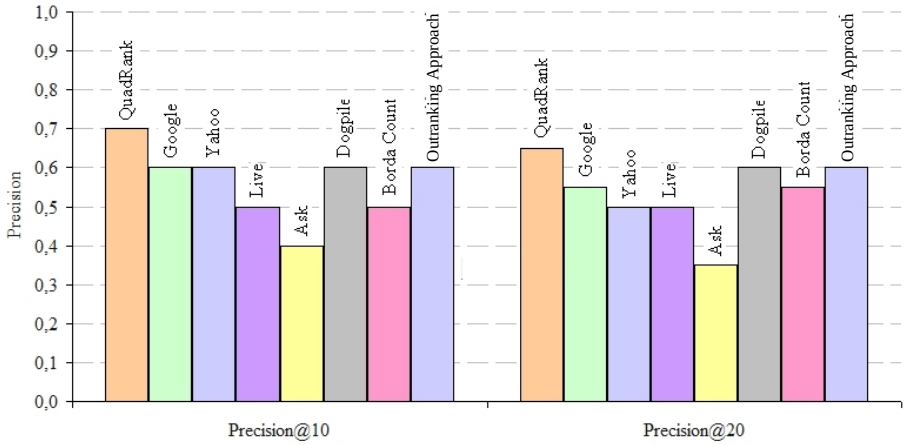

Figure 1: Measurements of Precision@10 and Precision@20 for various search engines for the query *"tickets for uefa champions league final 2010"*

Engine	QuadRank	G	Y	L	A	BC	OA
QuadRank	-	-0.26	0.16	-0.20	-0.07	0.41	0.31
Google	-0.26	-	0.09	0.05	-0.09	0.09	0.25
Yahoo!	0.16	0.09	-	-0.46	-0.42	0.03	0.16
Live	-0.20	0.05	-0.46	-	-0.32	-0.05	-0.03
Ask	-0.07	-0.09	-0.42	-0.32	-	-0.30	-0.63
Borda Count	0.41	0.09	0.03	-0.05	-0.30	-	0.84
Outranking Approach	0.31	0.25	0.16	-0.03	-0.63	0.84	-

Table 8: Rankings Correlation for the Query *"tickets for uefa champions league final 2010"*.

QuadRank.

In addition, the reader should note an interesting case. If we compare the 8^{th} and 10^{th} entries of Table 6, we conclude that the latter received better ranking from the component engines than the former. However, since QuadRank is not relied completely on the individual rankings of the component engines, it positions them in the opposite manner. That decision was vindicated, since the 8^{th} entry is relevant to the given query, whereas the 10^{th} is not. There are many such cases in the rankings generated by QuadRank: For example, compare the 6^{th} entry to the 7^{th} and 15^{th} to 16^{th} where the individual rankings were correctly overlooked. On the other hand, the competitor rank aggregation methods could not distinguish the difference and provided incorrect rankings.

Now let us examine how the involved result lists correlate. To evaluate the correlation of the produced rankings, we employed the Spearman's rho measure. The results illustrated in Table 8, reveal that all algorithms produce lists that diverge significantly.

In the sequel, we measure the performance of the proposed algorithm for another situation, where a hypothetical engineer seeks information about how to construct an inverted index in a distributed environment. The submitted query is phrased as *"distributed index construction"* and

	URL	**R**	G	Y	L	A	BC	OA
1	nlp.stanford.edu	R_1	2	1	2	1	1	1
2	www.reedconstructiondata.com	-	4	4	-	3	2	5
3	www.ims.uni-stuttgart.de	R_2	-	8	5	-	8	6
4	www.reedconstructiondata.com	-	5	-	-	4	6	12
5	ilpubs.stanford.edu:8090	R_3	7	-	3	9	4	2
6	nlp.stanford.edu	-	-	-	4	-	10	21
7	www.ics.uci.edu	R_4	-	-	10	-	41	28
8	lecs.cs.ucla.edu	R_5	-	16	14	22	5	3
9	www.ics.uci.edu	R_6	-	-	12	-	49	27
10	nlp.stanford.edu	R_7	3	2	-	2	3	4
11	ilpubs.stanford.edu:8090	R_8	-	-	15	-	74	19
12	en.wikipedia.org	-	-	7	28	-	25	7
13	ilpubs.stanford.edu	R_9	-	9	-	-	28	42
14	www10.org	R_{10}	-	10	-	-	30	53
15	docs.huihoo.com	R_{11}	9	-	-	11	10	9
16	portal.acm.org	-	-	24	-	-	77	43
17	www.reedconstructiondata.com	R_{12}	-	3	-	-	18	52
18	www10.org	R_{13}	-	-	11	-	46	32
19	www.dcs.bbk.ac.uk	R_{14}	-	30	-	-	97	85
20	citeseerx.ist.psu.edu	-	6	-	-	5	6	8

Table 9: The top-20 list for the query *"distributed index construction"* when the *QuadRank* Algorithm is Applied.

dictates out metasearch engine to request 30 results from each of its component engines. *QuadSearch* received 120 results and after the merging process, a set of 100 candidates is generated and ranked.

A document is considered relevant to the given query, only if its content is relevant to *"indexes"* and it contains some instructions regarding *"distributed construction"* in its body. All the results that provided information regarding index compression, organization or single-node construction were marked as irrelevant.

Compared to the previous case (where the set of candidates contained 88 items), we conclude that for this query, the engines' coverage is significantly smaller. At first, the ranking that our algorithm generates, is studied. Table 9 illustrates the top-20 list returned by QuadRank and also displays the individual rankings that each result received by the component engines. The last two columns record the rankings that the Borda Count method and the Outranking approach produced.

The Google and Ask component engines provided answers of medium quality for this query; only five results out of ten were considered relevant, as only five contained the required information. Google concentrated on presenting scientific papers about *"indexing"* but some of these works were originating from other sciences such as biology. In addition, PageRank played its role, as most of the results are well known and institutional sources of scientific information. In addition, Ask returned a top-20 list of low quality, since it was deceived by the term *"index"*, a word that is commonly used to describe the home page of a Web site.

On the other hand, Yahoo! and Live Search performed better in this case, since their top-10 rankings contained six pages containing useful information. These engines also presented

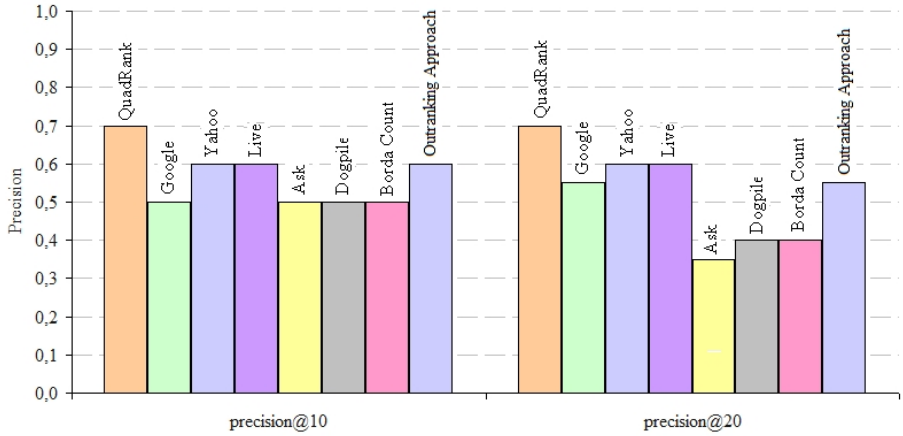

Figure 2: Measurements of Precision@10 and Precision@20 for various search engines for the query *"distributed index construction"*

Engine	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**R**
QuadRank	R	-	R	-	R	-	R	R	R	R	7
Google	-	R	R	-	-	-	R	R	R	-	5
Yahoo!	R	R	-	-	-	R	-	R	R	R	6
Live Search	R	R	R	-	R	R	-	-	-	R	6
Ask	R	R	-	-	R	R	-	-	R	-	5
Dogpile	-	R	-	-	-	-	R	R	R	R	5
Borda Count	R	-	R	R	-	-	-	R	-	R	5
Outranking Approach	R	R	R	R	-	R	-	-	R	-	6

Table 10: Relevant documents in the engines' top-10 lists for *"distributed index construction"*.

satisfactory top-20 lists containing twelve relevant entries each.

QuadRank outperformed all of its opponents, since its top-10 and top-20 lists included 7 and 14 relevant results. Dogpile's top-10 list was of equal quality to Google and Ask, but its top-20 list was more informative only than Ask's. Unlike the previous case where several sponsored links were included in the top-10 list of Dogpile, for this query the metasearch engine provided only organic results. Five of them were relevant to the query, whereas the others provided information regarding distributed audio and video communication systems and distributed process control systems.

The Borda Count method generated results of equal quality to Dogpile; five relevant items in the top-10 list and only eight in the top-20. It becomes obvious that a simple rank aggregation method that is heavily depended on the individual rankings of the component engines, cannot always provide robust ranking. On the other hand, the component of the QuadRank algorithm (equation 4) which examines the titles, the snippets and the URLs of the collected elements, leads to result lists of improved quality.

Once again the Outranking approach was more effective than the Borda Count method since

it fetched six relevant documents in its top-10 list and twelve in its top-20. However, the method did not produce as qualitative rankings as QuadRank.

In Table 10 we depict the relevancy of each item of the examined top-10 lists to the query in question. Moreover, in Figure 2 we illustrate the precision of each ranking algorithm, measured at cutoff points 10 and 20.

Finally, in Table 11 we evaluate the correlation of the various rankings of the experiment, by recording the values of the Spearman's rho measure.

Engine	QuadRank	G	Y	L	A	BC	OA
QuadRank	-	-0.42	0.16	-0.09	-0.44	0.56	0.64
Google	-0.42	-	-0.53	-0.33	-0.50	-0.31	-0.32
Yahoo!	0.16	-0.53	-	-0.18	-0.35	-0.20	-0.41
Live	-0.09	-0.33	-0.18	-	-0.27	-0.12	0.35
Ask	-0.44	-0.50	-0.35	-0.27	-	-0.38	-0.48
Borda Count	0.56	-0.31	-0.20	-0.12	-0.38	-	0.89
Outranking Approach	0.64	-0.32	-0.41	0.35	-0.48	0.89	-

Table 11: Rankings Correlation for the Query *"distributed index construction"*.

The third query we present here originates from the broader fields of health and medication. Here we are interested in locating the symptoms of the cancer of lungs, hence the query we phrase is *lungs cancer symptoms*. The information need is quite targeted in this query, since we desired to attest the accuracy of our algorithm in similar cases. Hence, the documents containing information regarding only the causes or the treatments of the disease are considered as irrelevant. The same holds for results whose content is about other types of cancers or other types of lungs maladies.

A page is considered relevant to the specified query, only if it contains the desired information itself; not because it includes sets of links to other pages which "claim" to provide such information. This rule was strictly followed by our colleagues during the identification of relevant results.

Similarly to the previous experiments, our metasearch engine requested 30 results from each component engine and after the merging process, a set of 91 elements is generated and ranked. In Table 12 we record the elements of the top-20 list that QuadRank produced and also, the individual rankings that each one received by the component engines. As before, that dash symbol represents the absence of the element from an engine's result list.

QuadRank presented the most qualitative top-10 list, since 8 results were relevant to the given query. Furthermore, the first seven results were all of high informational quality, as they included adequate reference regarding the subject of the query. Live search and Yahoo! were also effective, although their top-10 rankings included one result less than QuadRank's list. Regarding the other two component engines, Google and Ask, they were slightly less efficient, by including six relevant results in their top-10 rankings.

The first opponent rank aggregation method, Borda Count, constructed a top-10 list comprised of seven relevant documents and it was outperformed by our proposed method. On the other hand, the Outranking approach returned results that were of equal quality to those of QuadRank. The other metasearch engine that we examine in our experiments, Dogpile, exhibited poor performance and it was the worst among our examined search engines, by building a top-10 list which contained only five relevant results. Table 13 provides a detailed image of the top-10

	URL	**R**	G	Y	L	A	BC	OA
1	lungcancer.about.com	R_1	13	7	-	-	10	15
2	www.merck.com	R_2	5	-	6	9	5	6
3	www.cancerssociety.org	R_3	-	-	-	14	42	74
4	www.webmd.com	R_4	10	2	7	-	4	4
5	www.emedicinehealth.com	R_5	2	-	3	5	2	3
6	www.cancerhelp.org.uk	R_6	4	-	-	2	7	9
7	www.medicinenet.com	R_7	1	4	1	3	1	1
8	lungcancer.about.com	-	12	6	-	18	6	5
9	www.webmd.com	R_8	-	1	5	-	8	7
10	www.emedicinehealth.com	-	3	5	-	6	3	2
11	www.macmillan.org.uk	R_9	-	-	-	4	26	75
12	www.mayoclinic.com	R_{10}	11	11	-	-	13	19
13	health.yahoo.com	R_{11}	14	13	-	-	14	14
14	www.cancerhelp.org.uk	R_{12}	-	-	-	11	38	73
15	www.lungscancer.com	-	-	-	2	-	24	34
16	www.cancer.gov	R_{13}	-	24	21	-	21	8
17	www.wrongdiagnosis.com	R_{14}	-	8	-	-	31	63
18	www.cancerbackup.org.uk	-	-	-	-	22	63	71
19	www.webmd.com	-	9	-	-	-	33	88
20	www.cancercenter.com	R_{15}	-	17	-	-	50	50

Table 12: The top-20 list for the query *"lungs cancer symptoms"* when the *QuadRank* algorithm is applied.

Engine	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**R**
QuadRank	R	R	R	R	R	R	R	-	R	-	8
Google	-	R	-	R	R	R	R	-	-	R	6
Yahoo!	R	R	R	-	-	-	R	R	R	R	7
Live Search	-	-	R	R	R	R	R	R	-	R	7
Ask	R	R	-	R	R	-	-	-	R	R	6
Dogpile	-	-	-	-	-	R	R	R	R	R	5
Borda Count	R	R	-	R	R	-	R	R	-	R	7
Outranking Approach	R	-	R	R	-	R	R	R	R	R	8

Table 13: Relevant documents in the engines' top-10 lists for *"lungs cancer symptoms"*.

rankings of the examined search engines.

The top-20 ranking of Borda Count is significantly improved compared to the top-10 list and was equally qualitative compared to the ones constructed by our QuadRank algorithm and the Yahoo! component engine. In total, the top-20 lists of these three ranking methods included 15 relevant documents. The left and right diagrams of Figure 3 illustrate the precision values of the various systems and algorithms for the top-10 and top-20 listings respectively.

Finally, in Table 14 we record the correlation of the rankings by using the Spearman's rho measure.

The example queries we have studied here reveal that a single search engine cannot perform equally well for all types of queries. Although Google was effective for *"tickets for uefa champions league final 2010"*, it did not provide equally informative results for the query *"distributed*

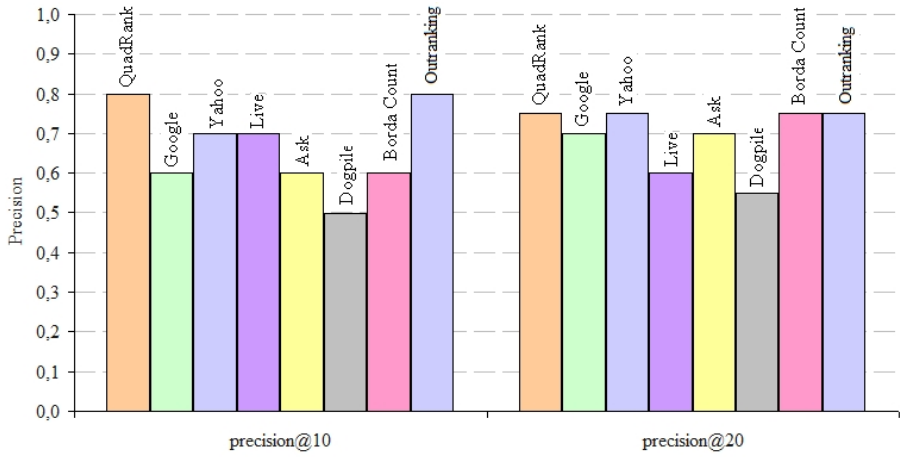

Figure 3: Measurements of Precision@10 and Precision@20 for various search engines for the query *"lungs cancer symptoms"*

Engine	QuadRank	G	Y	L	A	BC	OA
QuadRank	-	-0.41	0.22	-0.26	-0.26	0.65	0.50
Google	-0.41	-	-0.14	-0.48	-0.01	0.82	0.63
Yahoo!	0.22	-0.14	-	-0.58	-0.39	-0.07	0.06
Live	-0.26	-0.48	-0.58	-	-0.50	-0.62	-0.59
Ask	-0.26	-0.01	-0.39	-0.50	-	0.16	-0.07
Borda Count	0.65	0.82	-0.07	-0.62	0.16	-	0.73
Outranking Approach	0.50	0.63	0.06	-0.59	-0.07	0.73	-

Table 14: Rankings Correlation for the query *"lungs cancer symptoms"*.

index construction". The opposite behavior was exhibited by Live Search and Yahoo, which presented informative top-10 lists in the second and third cases.

On top of that, our QuadRank scoring method performed steadily well on all submitted queries. This sense is present on most cases, as our metasearch engine manages to eliminate the component engines' weak spots and combine their advantages efficiently.

Regarding the competitor rank aggregation methods, QuadRank outperformed Borda Count by a significant margin for all of the three queries we have tested. QuadRank was also more effective than the Outranking approach on two cases and equally accurate on the third.

4.3. Evaluation of QuadSearch's response times

In the previous subsection, we have demonstrated how the use of the proposed rank aggregation method can lead to better rankings, when compared to the rankings that the component engines produce. We have indicated that the term "better", concerns both Precision and quality of the returned results.

The second most important measure of a system's performance, is how quickly it responds to a given query. It may well produce qualitative results, but this could be close to useless, unless these results are produced in reasonable times.

Before we proceed to the evaluation of *QuadSearch*'s response, we discuss all the time penalties that any metasearch engine has to suffer, before it presents a list of results to the user. We also define *idle time*, as the total intermediate time between the moment the user submits a query and the moment that the engine presents the results. All idle times presented here, are expressed in seconds. In addition, henceforth, we will use the term *server*, to refer to the machine that hosts the search system.

The idle time is affected by numerous factors. Generally, we cannot exactly calculate it beforehand; we can only estimate a value, which is computed by using the following relationship:

$$t_{tot} = t_{req} + t_{res} + t_{ret} + t_{pr}, \tag{8}$$

where

- t_{req} represents the request time, that is, the time that the system needs to send the request to its component engines. It depends on the server's upload capabilities. This time is usually infinitesimal.

- t_{res} is the total time the metasearch engine has to wait for the component engines to respond. In other words, this is the time a single engine consumes to search its index structure and apply its own ranking algorithm to form the results' list. As it is obvious, there is no technique that can be applied to improve this timing.

- t_{ret} signifies the results' pages downloading duration. This is the most unstable factor that affects a metasearch engine's response. It depends on the server's download capability, the number of concurrent users, the daytime and other parameters that are rigorous to manage. In general, improving the network capacity of the server, leads to smaller download delays.

- t_{pr} indicates the overhead added by the execution of the fusion and ranking algorithms of the metasearch engine. The application of additional filters (such as the anti-spam filter and engine bombing protection), further increases this time.

To evaluate the *QuadSearch*'s speed, we have formed a set of fifteen queries. During the tests we tried to identify some terms whose inverted lists were stored in the component engines' caches and some that were not. These queries have been submitted to *QuadSearch* with various parameters enabled and disabled, for all the proposed rank aggregation methods. We also exploit all of the four component engines available.

In Table 15 we record the time performance of various rank aggregation methods in the environment of QuadSearch for each query of our set, when the system requests $k = 30$ results from its component engines. The second column shows the retrieval times t_{ret}, that is, the time the metasearch engine needs to download the result lists from the component engines. In the third column we record the length of the list which derives from the fusion of the input rankings, whereas the three last columns contain the times each algorithm needs to rank the merged list.

Borda Count is the fastest algorithm and this is expected since the score of each document is computed by simply adding the individual rankings it received by the component engines. QuadRank is somehow slower (by a margin of about 23%) since the evaluation of the $Z(c)$ and $U(c)$ scores is slightly more expensive. The Outranking approach is by far the slowest method

Query	t_{ret}	N	t_{pr}		
			Borda Count	Outranking	QuadRank
tokyo hotel	1.53	90	0.04	0.09	0.05
artificial fertilizers	1.41	87	0.04	0.08	0.05
free flv player	1.34	80	0.03	0.07	0.04
lewis hamilton	1.56	90	0.04	0.09	0.05
markov chains	1.38	79	0.03	0.07	0.04
greenhouse effect	1.30	77	0.02	0.07	0.04
h-index	1.42	87	0.04	0.08	0.04
marine biology	1.38	85	0.04	0.08	0.05
public key encryption	1.64	90	0.04	0.11	0.06
voice over ip	1.55	89	0.04	0.10	0.06
waterfall	1.29	94	0.04	0.10	0.04
scorched earth	1.34	85	0.04	0.08	0.05
apple	1.21	92	0.04	0.09	0.04
cold war	1.22	84	0.03	0.08	0.04
iran nuclear weapons	1.54	99	0.05	0.11	0.07
AVERAGE	1.41	87.2	0.037	0.086	0.048

Table 15: *QuadSearch* response times for various rank aggregation methods and 30 requested results per engine.

since each document must be compared with the rest $N - 1$ documents in all 4 input rankings. Therefore, it is approximately 2.3 times slower than Borda Count and 1.8 times slower than QuadRank.

Notice that processing times for Borda Count and the Outranking Approach depend only on the number of documents of the merged list. Therefore, the more documents the merged list contains, the slower Borda Count and the Outranking Approach become. On the other hand, apart from the number of documents of the merged list, the timings of QuadRank also depend on the length of the submitted query due to the existence of zone weighting. Consequently, we expect from longer queries to be processed at lower rates.

In Table 16 we repeat the experiment by requesting top-100 lists. Apparently, in this case the average retrieval time increases significantly as more time is required to download larger lists. Furthermore, the fusion and ranking duration increases, since now we have to process more documents. Notice that while QuadRank remains slower than Borda Count by a relatively stable percentage of 24%, the performance of the Outranking Approach degrades considerably faster (2.5 times slower than QuadRank and 3.6 times slower than Borda Count).

5. Concluding remarks and future work

In this article, we considered the issue of developing a new metasearch engine to assist in the process of Web information retrieval. The main motivation to develop this metasearch engine was the common intuition that a rank aggregation algorithm should a) be related to the comparison of the top-k lists of each conventional metasearch engine, and b) offer efficient computation and low response times. Thus, we came up with a new method for rank aggregation, i.e., the *QuadRank* method. We injected some new parameters, like the number of the top-k lists that a page appears, the total number of exploited search engines and the size of the top-k lists. QuadRank is also

Query	t_{ret}	N	t_{pr}		
			Borda Count	Outranking	QuadRank
tokyo hotel	2.01	276	0.23	0.82	0.31
artificial fertilizers	1.95	295	0.25	0.94	0.37
free flv player	1.94	259	0.21	0.74	0.35
lewis hamilton	2.13	277	0.22	0.85	0.30
markov chains	1.98	267	0.21	0.78	0.36
greenhouse effect	1.99	255	0.21	0.71	0.28
h-index	2.03	297	0.26	0.97	0.31
marine biology	2.13	269	0.22	0.80	0.29
public key encryption	2.07	292	0.25	0.93	0.38
voice over ip	2.22	296	0.26	0.96	0.57
waterfall	2.25	306	0.26	1.02	0.33
scorched earth	1.99	278	0.22	0.85	0.32
apple	1.91	274	0.22	0.82	0.26
cold war	2.07	286	0.24	0.88	0.33
iran nuclear weapons	2.40	314	0.27	1.08	0.40
AVERAGE	2.07	281.6	0.24	0.87	0.34

Table 16: *QuadSearch* response times for various rank aggregation methods and 100 requested results per engine.

the only rank aggregation method which takes into consideration additional parameters such as zone weighting, the metadata of each collected document and URL analysis. The best way to experiment and test the introduced method was to develop a new metasearch engine, named *QuadSearch*, a name related to the current number of exploited engines. The new metasearch engine is publicly available at http://quadsearch.csd.auth.gr.

We have tested the new method for both effectiveness and efficiency against other rank aggregation methods (Borda Count, Outranking approach) by using data from the TREC conference and also, in the real-world of metasearching. QuadRank outperformed its competitors in most cases by a significant margin.

References

Ailon, N., 2007. Aggregation of partial rankings, *p*-ratings and top-*m* lists. In: Proceedings of the International ACM-SIAM Symposium on Discrete Algorithm (SODA). pp. 415–424.

Ailon, N., Charikar, M., Newman, A., 2005. Aggregating inconsistent information: Ranking and clustering. In: Proceedings of the ACM International Symposium On Theory of Computing (STOC). pp. 684–693.

Allen, J., 2009. Comparison of metasearch engines. Tech. rep., Southern Methodist University, CSE8337, available at http://jdadesign.net/wp-content/uploads/2010/01/HW2b.pdf.

Aslam, J. A., Montague, M. H., 2001a. Metasearch consistency. In: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR). pp. 386–387.

Aslam, J. A., Montague, M. H., 2001b. Models of metasearch. In: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR). pp. 276–284.

Beg, M. M. S., Ahmad, N., 2003. Soft computing techniques for rank aggregation on the World Wide Web. World Wide Web Journal 6 (1), 5–22.

Coppersmith, D., Fleischer, L., Rudra, A., 2006. Ordering by weighted number of wins gives a good ranking for weighted tournaments. In: Proceedings of the International ACM-SIAM Symposium on Discrete Algorithm (SODA). pp. 776–782.

DeConde, R. P., Hawley, S., Falcon, S., Clegg, N., B., K., Etzioni, R., 2006. Combining results of microarray experiments: A rank aggregation approach. Statistical Applications in Genetics and Molecular Biology 5 (1), 1–23.

Dwork, C., Kumar, R., Naor, M., Sivakumar, D., 2001. Rank aggregation methods for the Web. In: Proceedings of the ACM International Conference on World Wide Web (WWW). pp. 613–622.
Fagin, R., Kumar, R., Sivakumar, D., 2003. Efficient similarity search and classification via rank aggregation. In: Proceedings of the ACM International Conference on Management of Data (SIGMOD). pp. 301–312.
Farah, M., Vanderpooten, D., 2007. An outranking approach for rank aggregation in information retrieval. In: Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR).
Liu, Y.-T., Liu, T.-Y., T., Q., Ma, Z.-M., Li, H., 2007. Supervised rank aggregation. In: Proceedings of the ACM International Conference on World Wide Web (WWW). pp. 481–489.
Lu, W., Robertson, S., MacFarlane, A., 2006. Field-weighted XML retrieval based on BM25. Lecture Notes in Computer Science 3977, 161–171.
Manning, C. D., Raghavan, P., Schutze, H., 2008. Introduction to Information Retrieval. Cambridge University Press, in press.
Meng, W., Yu, C., Liu, K.-L., 2002. Building efficient and effective metasearch engines. ACM Computing Surveys 34 (1), 48–89.
Oztekin, B. U., Karypis, G., Kumar, V., 2002. Expert aggreement and content based reranking in a metasearch environment using Mearf. In: Proceedings of the ACM International Conference on World Wide Web (WWW). pp. 333–344.
Renda, M. E., Straccia, U., 2003. Web metasearch: Rank vs score based rank aggregation methods. In: Proceedings of the ACM International Symposium on Applied Computing (SAC). pp. 841–846.
Saari, D. G., March, 4 2000. The mathematics of voting: Democratic symmetry. Economist, 83.
Sculley, D., 2007. Rank aggregation for similar items. In: Proceedings of the SIAM Conference on Data Mining (SDM).
Shokouhi, M., 2007. Segmentation of search engine results for effective data-fusion. In: Proceedings of the European Conference on Information Retrieval (ECIR). Vol. 4425 of Lecture Notes in Computer Science. pp. 185–197.
Soboroff, I., Craswell, N., Clarke, C., 2009. Overview of the trec 2009 web track.
Souldatos, S., Dalamagas, T., Sellis, T., 2005. Sailing the Web with Captain Nemo: A personalized metasearch engine. In: Proceedings of the ICML workshop: Learning in Web Search (LWS). Bonn, Germany.
Spink, A., Jansen, B. J., Blakely, C., Koshman, S., 2006. Overlap among major Web search engines. In: Proceedings of the IEEE International Conference on Information Technology: New Generations (ITNG). pp. 370–374.
Sugiura, A., Etzioni, O., 2000. Query routing for Web search engines: Architecture and experiments. Computer Networks 33 (1–6), 417–429.
Tan, P.-N., Jin, R., 2004. Ordering patterns by combining opinions from multiple sources. In: Proceedings of the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD). pp. 695–700.
Vogt, C. C., 1999. Adaptive combination of evidence for information retrieval. Ph.D. thesis, University of California at San Diego.
Vogt, C. C., Cottrell, G. W., 1999. Fusion via a linear combination of scores. Information Retrieval 1 (3), 151–173.
Young, H. P., Levenglick, A., 1978. A consistent extension of Condorcet's election pronciple. SIAM Journal on Applied Mathematics 35 (2), 285–300.