Web Metasearch: Rank vs. Score Based Rank Aggregation Methods*

M. Elena Renda
I.S.T.I. – CNR
Area della Ricerca del CNR
Via G.Moruzzi 1, 56124 Pisa - ITALY
renda@iei.pi.cnr.it

Umberto Straccia
I.S.T.I. – CNR
Area della Ricerca del CNR
Via G.Moruzzi 1, 56124 Pisa - ITALY
straccia@iei.pi.cnr.it

ABSTRACT

Given a set of rankings, the task of ranking fusion is the problem of combining these lists in such a way to optimize the performance of the combination. The ranking fusion problem is encountered in many situations and, *e.g.,* metasearch is a prominent one. It deals with the problem of combining the result lists returned by multiple search engines in response to a given query, where each item in a result list is ordered with respect to a search engine and a relevance score. Several ranking fusion methods have been proposed in the literature. They can be classified based on whether: (i) they rely on the rank; (ii) they rely on the score; and (iii) they require training data or not. Our paper will make the following contributions: (i) we will report experimental results for the Markov chain rank based methods, for which no large experimental tests have yet been made; (ii) while it is believed that the rank based method, named Borda Count, is competitive with score based methods, we will show that this is not true for metasearch; and (iii) we will show that Markov chain based methods compete with score based methods. This is especially important in the context of metasearch as scores are usually not available from the search engines.

Categories and Subject Descriptors

H.3.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Information Search and Retrieval—*Retrieval models*; H.3.4 [**INFORMATION STORAGE AND RETRIEVAL**]: Systems and Software—*Performance evaluation (efficiency and effectiveness)*

Keywords

Meta-search, rank list aggregation

*This work is funded by the European Community in the context of the CYCLADES project IST-2000-25456, under the Information Societies Technology programme.

SAC2003, Melbourne, Florida, USA

1. INTRODUCTION

The problem of *rank fusion* is the problem of computing a "consensus" ranking, given individual ranking preferences (a ranking is a linear ordering of a set of items) of several *judges*. The ranking fusion problem is encountered in many situations and a prominent one is metasearch: it deals with the problem of combining the result lists returned by multiple search engines in response to a given query, where each item in a result list is ordered with respect to (w.r.t.) a search engine and a relevance score.

While search engines certainly help users in locating information relevant to the user's information need, they still have a number of deficiencies: (*i*) indexing Web data is a time and space consuming task. As the content of the Web changes rapidly, each search engine has to set up a tradeoff between the *coverage*, *i.e.,* the number of Web documents indexed w.r.t. the whole Web and the *update frequency*, *i.e.,* the time that occurs between the subsequent re-indexing of the complete database. However large the indexes are, still a search engine (re) indexes only a small subset of all available documents on the WWW; (*ii*) many information sources, *e.g.,* proprietary information sources like the Digital Libraries of Editors[1], are not indexible as they do not admit the gathering of their documents. These are essentially databases that cannot be indexed by search engines. The only way to search for information within these Digital Libraries is to rely on the search services provided by them; (*iii*) for some search engines[2], the more advertisers pay, the higher they will rank in the search results, called *pay-for-placement*, with a consequently average loss in precision; (*iv*) search engines are subject to spamming [7], *i.e.,* a search engine has been spammed by a page in its index, on a given query, if it ranks the page "too highly".

Limitations, as such listed above, has led to the introduction of metasearch engines[3] with the aim of both alleviating user's work and to improve the retrieval effectiveness.

The ideal scenario for ranking fusion is when each judge (search engine) gives a complete ordering of all the alternative items in the universe of alternatives. Unfortunately, in metasearch this is far too unrealistic for two reasons: (*i*) the coverage of search engines is different; and (*ii*) search engines limit access to the top 100 or 1000 ranked items of their ordering. Therefore, any method for rank aggregation

[1] `www.acm.org/dl`,`www.ieee.org`,`www.elsevier.com`
[2] `www.goto.com`, `www.findwhat.com`
[3] For example, see MetaCrawler [16], SavvySearch [6], Inquirus [11], ProFusion [10].

for Web application must be capable of dealing with the fact that only a limited number of entries of each ranking is available. Of course, if there is no overlap among the rankings, there isn't much any fusion method can do. A challenge is to design ranking fusion methods that work when there is limited, but non-trivial, overlap among the top few hundreds or thousands of items in each ranking.

Several ranking fusion methods have been proposed in the literature [1, 4, 7, 12]. A major distinction between the methods is that they can be classified based on whether: (i) they rely on the rank; (ii) they rely on the score; and (iii) they require training data or not. Preliminary experimental results seem to indicate that score based methods outperform rank based methods, while methods based on training data perform better than those without training data.

In this work we will compare rank and score based methods, without training data, in the context of metasearch. In particular, our paper will make the following contributions: (i) we will report experimental results for the rank based method based on Markov chains, for which no large experimental tests have yet been made; (ii) while it is believed that the rank based method, named Borda Count, is competitive with score based methods, we will show that this is not true for metasearch; and (iii) we will show that Markov chain based methods compete with score based methods. This last result is especially important in the context of metasearch as scores are usually not available from the search engines.

In the following, we first review various approaches to the ranking fusion problem. In Section 3 we present our experimental results. Section 4 concludes with directions for future research.

2. RANKING FUSION METHODS

There exist various methods for merging rank-ordered lists. Basically, they use information that is readily available from ranked lists of items. In most cases, the strategies relies on the following information: (*i*) the ordinal rank assigned to an item in the rank list; and (*ii*) the score assigned to an item in the rank list. In *score based* methods, items are ranked in order of the assigned *scores* in the rank lists, or some transformation of those scores [1, 4, 8, 9, 12, 13, 17, 18], while in *rank based* merging methods, items are ranked in order of the assigned *ranks* in the rank lists, or some transformation of those ranks [1, 7, 12, 19]. Another orthogonal distinction of rank fusion methods is whether the methods rely on training data (*e.g.,* the *Bayes-fuse* method [1], the *linear combination* method [17] and the *preference rank combination* method [8]) or not. Another class of methods is based on the *content* of ranked items. In these methods, the ranked documents are downloaded and analysed in order to produce the final ranking. We will not address them here.

2.1 Preliminaries

At first, some basic definitions (see [7]) towards an uniform way to present all the methods. Let U be a set of items, called *universe*. A *rank list* (or simply *ranking*) τ w.r.t. U is an ordering of a subset $S \subseteq U$, *i.e.,* $\tau = [x_1 \geq x_2 \geq \cdots \geq x_k]$, with each $x_i \in S$, and $\geq$ some ordering relation on S. Also, if $i \in U$ is present in τ, written $i \in \tau$, let $\tau(i)$ denote the position or rank of i (a highly ranked or preferred element has a low-numbered position in the list). By assigning an unique identifier to each $i \in U$, without loss of generality we assume that $U = \{1, 2, \ldots, |U|\}$.

We may distinguish at least the following cases: (*i*) if τ contains all the elements in U, then τ is said to be a *full list*. In fact a full list is a total ordering on U, *i.e.,* a permutation. For instance, if U is the set of all indexed pages of a search engine, we get a full list when we rank the pages, say, w.r.t. the similarity to a query issued by a user; (*ii*) there are cases in which full lists are not convenient or even possible. For instance, let U denote the set of all Web pages in the world. Let τ denote the results of a search engine in response to a fixed query. Even though the query might induce a total ordering of the pages indexed by the search engine, since the index set of the search engine is most surely only a subset of U, we have a strict inequality $|\tau| < |U|$. Such lists that rank only some of the elements in U are called *partial lists*; (*iii*) a special case of partial lists is the following. If S is the set of all the pages indexed by a search engine and if τ corresponds to the top k (say, $k = 10$) results of the search engine w.r.t. a query, clearly the pages that are not present in the list τ can be assumed to be ranked below k by the search engine. Such lists are called *top k rank lists*, where k is the size of the list.

With $s^\tau(i)$ we will denote the real-valued *score* assigned to item $i \in U$ in rank list τ. Without loss of generality, we can assume that the higher the score the better the rank $\tau(i)$ of item i. If the score is not provided, $s^\tau(\cdot)$ is undefined. We will assume that either a score is provided for *all* items in the rank list or the score is provided for none of them. A rank list with scores will be called *scored rank list*. With $w^\tau(i)$ we will indicate the *normalised weight* of item $i \in \tau$ in ranked list τ. $w^\tau(i)$ is computed according to one of the following methods. Consider a set R of rank lists, let $\tau \in R$ and let U denote the union of the items occurring in R.

Definition 1 (Normalisation methods). *The* normalised weight, $w^\tau(i)$, *of an item $i \in \tau$ is defined as follows:*

Score normalisation: *for an item $i \in \tau$*

$$w^\tau(i) = \frac{s^\tau(i) - \min_{j \in \tau} s^\tau(j)}{\max_{j \in \tau} s^\tau(j) - \min_{j \in \tau} s^\tau(j)} \tag{1}$$

Z-score normalisation: *for an item $i \in \tau$*

$$w^\tau(i) = \frac{s^\tau(i) - \mu_{s^\tau}}{\sigma_{s^\tau}} \tag{2}$$

where μ_{s^τ} is the mean of the scores in τ and σ_{s^τ} is their standard deviation

Rank normalisation: *for an item $i \in \tau$*

$$w^\tau(i) = 1 - \frac{\tau(i) - 1}{|\tau|} \tag{3}$$

Borda rank normalisation: *for an item $i \in U$*

$$w^\tau(i) = \begin{cases} 1 - \frac{\tau(i)-1}{|U|} & \text{if } i \in \tau \\ \frac{1}{2} + \frac{|\tau|-1}{2 \cdot |U|} & \text{otherwise} \end{cases} \tag{4}$$

For both normalisation methods based on scores, we assume that there are at least two items with different score in the rank list. Otherwise we assign $w^\tau(i) = 1$. Furthermore, note that w^τ is always monotone not increasing. For score normalisation, the top ranked item has normalised weight 1, and the bottom ranked item has normalised weight 0. In

case of rank normalisation, w^τ is evenly distributed, which is not true for the score case. The difference in weight between two subsequent ranked items in τ is $1/|\tau|$. The top ranked item, has normalised rank weight 1, whereas the bottom ranked item, has weight $1/|\tau|$. Except the Borda rank normalisation method, all other normalisation methods do not depend on R. Additionally, some explanation is necessary for the Borda rank normalisation method. If $\tau \in R$ is a full list w.r.t. U, then the normalised Borda rank weight *coincides* with the normalised rank weight. In case of partial lists, there is a difference between the two. In fact, in case of normalised rank weight, the weight decreases with a factor $1/|\tau|$, while in Borda rank normalisation case the factor is $1/|U|$. As a consequence, the distribution of ranked items is equal within all lists $\tau_i \in R$. Furthermore, the normalised rank weight does not assign any value to unranked items ($i \notin \tau$), while in the Borda rank normalisation, for the items left unranked, the remaining weights are divided evenly among the unranked candidates. Formally, following the ideas of Borda [2] (see also [7, 14]), if $i \in \tau$, then $|U| + 1 - \tau(i)$ is the *Borda weight* assigned to item i ranked as $\tau(i)$. Normalising this weight we get $(|U| - \tau(i) + 1)/|U|$. In case item i is not ranked by τ, *i.e.,* $i \notin \tau$, $(|U| - |\tau| + 1)/2 \cdot |U|$ is the weight assigned to i according to an evenly distribution. Indeed, τ does not rank $d = |U| - |\tau|$ items. The remaining sum of Borda weights is, thus, $p = \sum_{i=1}^{d} i = d \cdot (d+1)/2$, which distributed evenly to the d items gives $d \cdot (d+1)/2 \cdot d = (d+1)/2$ as weight for each item. After normalisation we have $(|U| - |\tau| + 1)/2 \cdot |U|$. Normalisation is usually applied before merging rank lists in order to uniform score distributions. Normalisation allows us to capture within an unique framework different fusion methods.

Finally, another important concept in ranking fusion methods is the notion of rank hits, introduced first in [12]. In [12] a ranking fusion method is presented, in the context of information retrieval, based on the assumption that the more search engines retrieve the same document, the more this document may be considered as relevant to a query and, thus, be ranked higher. This assumption can be rephrased in our context as: the more partial rank lists rank the same item, the higher this item should be ranked in the fused list.

DEFINITION 2 (RANK HITS). *Consider a set of rankings R. Let U denote the union of the items occurring in R. For each item $i \in U$, the* rank hits *of i w.r.t. R, denoted $h(i, R)$, is given by $h(i, R) = |\{\tau \in R : i \in \tau\}|$.*

2.2 Fusion methods

In this section we will present the ranking fusion methods that we will analyse. In the following, consider a set of n rankings $R = \{\tau_1, \ldots, \tau_n\}$. Let U denote the union of the items in $\tau_1, \ldots, \tau_n$, *i.e.,* $U = \bigcup_{\tau \in R, i \in \tau} \{i\}$. With $\hat{\tau}$ we will indicate the ranking (called *fused ranking* or *fused rank list*), which is the result of a rank fusion method applied to the rank lists in R. In order to fully specify $\hat{\tau}$, it will be sufficient to determine the score $s^{\hat{\tau}}(i)$ (called *fused score*), for each item $i \in U$, as $\hat{\tau}$ will be ordered according to decreasing values of $s^{\hat{\tau}}$. We will say that two fused rankings $\hat{\tau}_1, \hat{\tau}_2$, which are the result of two fusion methods, are *equal* iff $\hat{\tau}_1 = \hat{\tau}_2$, while we will say that $\hat{\tau}_1$ and $\hat{\tau}_2$ are *equivalent* iff for each $i \in U$, $\hat{\tau}_1(i) = \hat{\tau}_2(i)$ (they have the same ordering). Of course, equality implies equivalence and not vice-versa.

2.2.1 Linear combination methods

Fox and Shaw [9], and later Lee [12], presented ranking fusion methods, which are based on the unweighted min, max, or sum of each item's normalised score. Lee, additionally, addressed the case where the rank has been considered in place of the score. Two interesting methods are listed below

$$\begin{array}{ll} \text{CombSUM:} & s^{\hat{\tau}}(i) = \sum_{\tau \in R} w^\tau(i) \\ \text{CombMNZ:} & s^{\hat{\tau}}(i) = h(i, R) \cdot \sum_{\tau \in R} w^\tau(i) \end{array}$$

From the test results in [9, 12], CombMNZ is considered as the best ranking fusion method, even if it performed slightly better than CombSUM. Essentially, CombMNZ is based on the fact that, according to Lee's experiments, "different search engines return similar sets of relevant documents but retrieve different sets of non-relevant documents". Indeed, the CombMNZ combination function heavily weights common documents.

In the methods above, all rankings have same priority. An obvious extension of the combination method is to introduce a preference weight, α_τ, associated to the rankings $\tau \in R$, where $\sum_{\tau \in R} \alpha_\tau = 1$ and $\alpha_\tau \geq 0$. For instance, Vogt [17] proposed to linearly combine the score normalised weights[4], *i.e.,* the fused score $s^{\hat{\tau}}$ of an item $i \in U$ is $s^{\hat{\tau}}(i) = \sum_{\tau \in R} \alpha_\tau \cdot w^\tau(i)$. Note that the proposed model requires training data to determine the weight α_τ. Although good results are achieved in specific cases, this technique has not yet been shown to produce reliable improvement.

We may generalise these methods as follows.

DEFINITION 3 (LINEAR COMBINATION METHOD). *In* linear combination based ranking fusion methods*, the fused score $s^{\hat{\tau}}(i)$ of an item $i \in U$ is defined according to*

$$s^{\hat{\tau}}(i) = h(i, R)^y \cdot \sum_{\tau \in R} \alpha_\tau \cdot w^\tau(i) \tag{5}$$

where (i) all the rank lists $\tau \in R$ have been normalised according to the same normalisation method; (ii) $y \in \{0, 1\}$ indicates whether hits are counted or not; and (iii) $\sum_{\tau \in R} \alpha_\tau = 1$ and $\alpha_\tau \geq 0$ indicates the priority of the rank list τ.

We assume that α_τ is the constant value $\frac{1}{|R|}$ if all rank list have the same priority, *e.g.,* no training data is considered.

In the following, we will identify the methods based on the above combination function (5) with $\Sigma.x.y$, where y is the value in the formula and $x \in \{r, s, z, b\}$ identifies the normalisation method: $x = r$ stands for rank normalisation, $x = s$ stands for score normalisation, $x = z$ stands for z-score normalisation and $x = b$ stands for Borda normalisation. For instance, the label $\Sigma.s.1$ indicates the fusion method based on Equation (5), where hits are counted and score normalisation has been applied.

2.2.1.1 Borda Count.

Borda's method [2, 14] is a voting method based on ranks, *i.e.,* it assigns a weight corresponding to the ranks in which a candidate appears within each voter's ranked list. Computationally they are very easy, as they can be implemented in linear time. The Borda Count (BC) method has been considered in the context of the rank fusion problem [1, 7], and works as follows. Each voter ranks a fixed set of c candidates

[4]The extension of weighted linear combination to the combination methods of Definition 3 is immediate.

in order of preference. For each voter, the top ranked candidate is given c points, the second ranked candidate is given $c-1$ points, and so on. If there are some candidates left unranked by the voter, the remaining points are divided evenly among the unranked candidates. The candidates are ranked in decreasing order of total points (See, *e.g.,* [7, 14]). Formally, the method is equivalent to the following: for each item $i \in U$ and rank list $\tau \in R$, consider the Borda normalised weight $w^\tau(i)$ (Definition 1). The fused rank list $\hat{\tau}$ is ordered w.r.t. the *Borda score* $s^{\hat{\tau}}$, where the Borda score of a item $i \in U$ in $\hat{\tau}$ is defined as

$$s^{\hat{\tau}}(i) = \sum_{\tau \in R} w^\tau(i) \tag{6}$$

Consequently, the BC is equivalent to the CombSUM method combined with Borda rank normalisation, *i.e.,* $\Sigma.b.0$. Aslam and Montague [1] also consider a *Weighted Borda Count*, where in place of the sum over the normalised Borda weights, a linear combination of these weights is considered, as for the linear combination method above.

2.2.2 Markov chain based methods

An interesting approach to rank fusion is proposed in [7] and is based on Markov chains. A (homogeneous) *Markov chain* for a system is specified by a set of states $S = \{1, 2, \ldots, |S|\}$ and an $|S| \times |S|$ non-negative, stochastic (*i.e.,* the sum of each row is 1) matrix M. The system begins in some start state in S and at each step moves from one state to another state. The transition is guided by M: at each step, if the system is at state i, it moves to state j with probability M_{ij}. If the current state is given as a probability distribution, the probability distribution of the next state is given by the product of the vector representing the current state distribution and M. In general, the start state of the system is chosen according to some distribution $\mathbf{x}$ (usually, the uniform distribution) on S. After m steps, the state of the system is distributed according to $\mathbf{x}M^m$. Under some conditions (which we will not discuss here), irrespective of the start distribution $\mathbf{x}$, the system eventually reaches an unique fixed point where the state distribution does not change (few steps may suffice). This distribution is called *stationary distribution*. It can be shown that the stationary distribution is given by the principal left eigenvector $\mathbf{y}$ of M, *i.e.,* $\mathbf{y}M = \lambda\mathbf{y}$. In practice, a simple power-iteration algorithm can quickly obtain a reasonable approximation of $\mathbf{y}$. The entries in $\mathbf{y}$ define a natural ordering on S. We call such an ordering the *Markov chain ordering* of M.

The application of Markov chains to the rank fusion problem is as follows. The set of states S corresponds to the list of all candidates to be ranked, *i.e.,* the set of all items in $R = \{\tau_1, \ldots, \tau_n\}$. The transition probabilities in M depend in some way on $\tau_1, \ldots, \tau_n$, as we will see below. $\hat{\tau}$ is then the Markov chain ordering on M. Below, some specific Markov chains (MC) are proposed [7]:

MC_1**:** if the current state is item i, then the next state is chosen uniformly from the multiset of all items j that were ranked higher than or equal to i by some ranking that ranked i, *i.e.,* chose the next state uniformly from the multiset $Q_i^{C_1} = \cup_{k=1}^{n}\{j : \tau_k(j) \leq \tau_k(i)\}$;

MC_2**:** if the current state is item i, then the next state is chosen by first picking a ranking τ uniformly from all the $\tau_1, \ldots, \tau_n$ containing i, then picking an item j uniformly from the set $Q_{\tau,i}^{C_2} = \{j : \tau(j) \leq \tau(i)\}$;

MC_3**:** if the current state is item i, then the next state is chosen as follows: first pick a ranking τ uniformly from all the $\tau_1, \ldots, \tau_n$ containing i, then uniformly pick an item j that was ranked by τ. If $\tau(j) < \tau(i)$ then go to j, else stay in i;

MC_4**:** if the current state is item i, then the next state is chosen as follows: first pick an item j uniformly from S. If $\tau(j) < \tau(i)$ for the *majority* of the lists $\tau \in R$ that ranked both i and j, then go to j, else stay in i.

Note that the Markov chain methods do rely on comparison among the ranks only and neither consider scores nor hits. Below is an example illustrating the Markov chain methods.

EXAMPLE 1. *Consider the following rankings $R = \{\tau_1, \tau_2, \tau_3\}$ of the three items in $S = \{1, 2, 3\}$*

rank lists	τ_1	τ_2	τ_3
1	1	3	3
2	2	1	2
3	3	2	1

It can be shown that the transition matrixes for cases MC_1, MC_2, MC_3 and MC_4 are the matrixes M^1, M^2, M^3 and M^4, respectively.

M^1

item	**1**	**2**	**3**
1	1/2	1/6	2/6
2	2/7	3/7	2/7
3	1/5	1/5	3/5

M^2

item	**1**	**2**	**3**
1	11/18	2/18	5/18
2	5/18	8/18	5/18
3	2/18	2/18	14/18

M^3

item	**1**	**2**	**3**
1	6/9	1/9	2/9
2	2/9	5/9	2/9
3	1/9	1/9	7/9

M^4

item	**1**	**2**	**3**
1	2/3	0	1/3
2	1/3	1/3	1/3
3	0	0	1

In the following, we show some example computations for the matrix entries. Remember that M_{ij}^k is the probability that, given the current state represented by item i, the next state is represented by item j.

- M_{13}^1 *is* 2/6. *Indeed, $Q_1^{C_1}$ is $\{1, 1, 3, 1, 2, 3\}$. So, the probability of picking uniformly one of the elements in $Q_1^{C_1}$ is* 1/6 *and, the probability of picking item* 3 *is* 2/6.
- M_{21}^2 *is* 5/18. *Indeed, the probability of picking uniformly a rank list containing item* 2 *is* 1/3. *If τ_1 has been selected then $Q_{\tau_1,2}^{C_2} = \{2, 1\}$. So, the probability of picking item* 1 *is* 1/2. *Similarly, $Q_{\tau_2,2}^{C_2} = \{2, 1, 3\}$ and $Q_{\tau_3,2}^{C_2} = \{2, 3\}$. Therefore, $M_{21}^2 = \frac{1}{3} \cdot \frac{1}{2} + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot 0 = \frac{5}{18}$.*
- M_{23}^3 *is* 2/9. *Indeed, the probability of picking uniformly a rank list containing item* 2 *is* 1/3. *The probability of uniformly picking an item within a rank list is* 1/3, *as well. Since, $\tau_1(3) \not< \tau_1(2)$, $\tau_2(3) < \tau_2(2)$, $\tau_3(3) < \tau_3(2)$, $M_{23}^3 = \frac{1}{3} \cdot 0 + \frac{1}{3} \cdot \frac{1}{3} + \frac{1}{3} \cdot \frac{1}{3} = \frac{2}{9}$ holds.*
- M_{22}^4 *is* 1/3. *Indeed, the probability of picking uniformly an item in S is* 1/3. *Additionally, let us consider the following table*

item	**1**	**2**	**3**
1	0	1	2
2	2	0	2
3	1	1	0

Each entry a_{ij} in the table above is the count of the lists $\tau \in R$, such that $\tau(j) < \tau(i)$, i.e., *in how many rankings item j is ranked better than item i. As there are three lists,*

the majority threshold is two. M^4_{22} is the probability that, given item 2, after the next step we still stay in document 2. As a_{21}, a_{22} and a_{23} are 2, 0 and 2, respectively, in two cases out of three we move away from item 2, whereas in one case we still stay in item 2. Consequently, M^4_{22} is 1/3.

Finally, the fused rank list $\hat{\tau}_k$ of the rank set R is the Markov chain ordering on M^k, $k = 1 \dots 4$. It can be shown that for all four cases $\hat{\tau}_k = [3 \geq 2 \geq 1]$.

3. EXPERIMENTS

Our experiments aims to compare the ranking fusion methods in the context of metasearch (without training data).

3.1 Data sets

We will use the systems submitted to the annual Text REtrieval Conference[5] (TREC). TREC offers large, standard data sets with many rank lists, ready to be fused. Usually, each year a large document data base S and a list of 50 queries are given. In the ad-hoc and Web information retrieval contest, each system x has to index this document base and then has to provide to the TREC organisation, for each query q, the top 1000 ranked documents, τ_q^x. For all queries q, all the rank lists $\tau_q^{(\cdot)}$ provide a pool of documents. Each document is then evaluated by humans whether being relevant or not w.r.t. q. After this manual step, each system is evaluated in terms of its effectiveness, *i.e.,* its *average precision* [15] is computed. As we would like to evaluate the fusion methods in the context of Web metasearch, we considered the Web results of TREC8, TREC9 and TREC2001, *i.e.,* the rank lists are related to World Wide Web engines. Furthermore, TREC9, is divided into a "short category", which we will identify with TREC9, and a non short one (identified by TREC9L). In summary, we rely on the results of four data sets. For each of them, we considered the top 12 performing rank lists, where at most one rank list per system has been considered[6]. The list is shown below[7].

TREC8 ok8wmx (.3829), Flab8wtdnN (.3405), INQ620 (.3327), mds08w1 (.322), UniNEW2Ct (.315), att99wtde (.3113), uwmt8w0 (.3066), acsys8wm (.3009), CL99WebM (.2889), iit99wt1 (.2265), DCU99L01 (.1939), Scai8Web1 (.1854)

TREC9: jscbt9wcs1 (.2011), hum9te (.197), uwmt9w10g4 (.1812), tnout9t2 (.1801), ric9tpx (.1787), NEtm (.1754), iit00t (.1627), Flab9atN (.136), apl9t (.1272), Sab9web1 (.1265), PuShortBase (.0654), CWI0000 (.0176)

TREC9L: iit00m (.3519), jscbt9wll2 (.2801), ric9dpn (.2616), NEnm (.2499), acsys9mw0 (.2486), hum9tdn (.2335), pir0Watd (.2209), NRKprf20 (.2173), Sab9web3 (.2159), apl9all (.1948), Flab9atdnN (.1923), xvsmman (.1785)

TREC2001: iit01m (.3324), ok10wtnd1 (.2831), csiro0mwa1 (.2817), flabxtd (.2332), UniNEn7d (.2242), fub01be2 (.2226), hum01tdlx (.2201), JuruFull (.2105), kuadhoc2001 (.2088), ricMM (.2084), jscbtawtl4 (.206), apl10wd (.2035)

3.2 Experimental setup

Our method to examine the effectiveness of the fusion methods is as follows. Each experiment is applied to a TREC8, TREC9, TREC9L and TREC2001 individually. For each TREC, each data point is the average of 10 trials. Each trial is performed as follows: randomly select a set R of $n \in \{2, 4, 6, 8, 10, 12\}$ rank lists w.r.t. a given TREC, apply the fusion methods and record the average precision of the fused rank lists. This experiment is designed to compare the fusion methods. The fusion methods that will be tested are $\Sigma.x.y$ with constant α_τ and the MC_is. Concerning MC_i, we have slightly modified them by assuming that each ranked item $i \in \tau$ is ranked *better* than any unranked item $j \notin \tau$.

[5] `http://trec.nist.gov`

[6] Indeed, it may happen that a contender submitted different runs. Then we consider the best run for it.

[7] The rank list ID and its average precision are shown.

3.3 Experimental results

The results of our experiments are reported in Tables 1–4. For each method, the values in the row indicates the *average precision* of the 10 trials for each group. Concerning the linear combination based fusion methods $\Sigma.x.y$, the following can be established:

1. w.r.t. the methods based on ranks, the methods which do not count the hits ($\Sigma.r.0, \Sigma.b.0$) perform better than those counting hits ($\Sigma.r.1, \Sigma.b.1$). The result is surprising, as it is believed that counting hits yields better performance;

	2	4	6	8	10	12	**Avg**
MC_1	.3328	.3711	.3803	.3850	.3872	.3905	**.3745**
MC_2	.3288	.3661	.3793	.3785	.3858	.3905	**.3715**
MC_3	.3301	.3584	.3689	.376	.3780	.381	**.3654**
MC_4	.3642	.3627	.3903	.3986	.3999	.3994	**.3858**
$\Sigma.r.1$	.3150	.3486	.3468	.3467	.3498	.3504	**.3428**
$\Sigma.r.0$	.3341	.3581	.3558	.3538	.3625	.3625	**.3545**
$\Sigma.s.1$	.3418	.3674	.3852	.3886	.3914	.3943	**.3781**
$\Sigma.s.0$	.3317	.3615	.3746	.3948	.3987	.4028	**.3774**
$\Sigma.z.1$	.3450	.3558	.3620	.3643	.3646	.367	**.3598**
$\Sigma.z.0$	.3038	.3569	.3501	.3571	.3608	.3657	**.3491**
$\Sigma.b.1$	.3288	.3440	.3369	.3430	.3416	.3432	**.3396**
$\Sigma.b.0$	.3249	.338	.3426	.3461	.3439	.3454	**.3402**

Table 1: TREC8 Experimental Results.

	2	4	6	8	10	12	**Avg**
MC_1	.1738	.1697	.1849	.1907	.1960	.197	**.1853**
MC_2	.159	.1839	.1887	.1946	.2034	.2072	**.1895**
MC_3	.1676	.1835	.1867	.1927	.1979	.2008	**.1882**
MC_4	.1486	.1993	.2110	.2093	.2080	.2065	**.1971**
$\Sigma.r.1$	.1745	.1571	.1543	.1426	.1488	.152	**.1549**
$\Sigma.r.0$	.1505	.1644	.1615	.1651	.1740	.1777	**.1655**
$\Sigma.s.1$	.1555	.173	.1881	.1952	.1966	.2017	**.185**
$\Sigma.s.0$	.1662	.1854	.1975	.2051	.2066	.2049	**.1943**
$\Sigma.z.1$	.1533	.1848	.1947	.2007	.2025	.2021	**.1897**
$\Sigma.z.0$	.1662	.1861	.1950	.2015	.2053	.2053	**.1932**
$\Sigma.b.1$	.1529	.1375	.1318	.1261	.1299	.1168	**.1325**
$\Sigma.b.0$	.1601	.1383	.1412	.1361	.1294	.124	**.1382**

Table 2: TREC9 Experimental Results.

	2	4	6	8	10	12	**Avg**
MC_1	.2979	.3045	.312	.3185	.3283	.3304	**.3153**
MC_2	.2491	.2847	.3168	.3208	.3268	.3304	**.3048**
MC_3	.2463	.2743	.3105	.3026	.3242	.3279	**.2977**
MC_4	.2683	.2990	.3206	.3224	.3303	.3287	**.3115**
$\Sigma.r.1$	.2512	.2804	.2834	.2956	.2979	.2979	**.2844**
$\Sigma.r.0$	.2624	.3029	.3018	.2961	.3085	.3042	**.2960**
$\Sigma.s.1$	.2681	.3056	.3068	.3301	.3332	.3471	**.3152**
$\Sigma.s.0$	.2714	.2912	.3189	.3265	.3355	.3425	**.3143**
$\Sigma.z.1$	.2414	.2732	.2791	.2765	.2832	.2798	**.2722**
$\Sigma.z.0$	.2647	.2533	.2786	.2782	.2754	.2725	**.2705**
$\Sigma.b.1$	.2858	.2691	.2913	.2881	.2940	.2942	**.2871**
$\Sigma.b.0$	.2522	.3064	.2952	.2982	.2954	.2944	**.2903**

Table 3: TREC9L Experimental Results.

2. w.r.t. the methods based on scores, there is no substantial difference between the methods which do count the

	2	4	6	8	10	12	**Avg**
MC_1	.2724	.2960	.3232	.3376	.3443	.3478	**.3202**
MC_2	.2581	.3194	.3146	.3353	.3488	.3478	**.3207**
MC_3	.2721	.3029	.3161	.3315	.3346	.3435	**.3168**
MC_4	.2796	.3096	.3178	.3234	.3294	.3341	**.3155**
$\Sigma.r.1$	.2611	.3035	.3175	.3282	.3543	.348	**.3188**
$\Sigma.r.0$	.2799	.3282	.3196	.3358	.3432	.353	**.3266**
$\Sigma.s.1$	.2740	.3077	.3430	.3563	.3410	.3493	**.3286**
$\Sigma.s.0$	.2699	.3161	.3427	.3425	.3406	.346	**.3263**
$\Sigma.z.1$	.2514	.2711	.2795	.2915	.2769	.2762	**.2745**
$\Sigma.z.0$	.2483	.2767	.2720	.2728	.2721	.2668	**.2681**
$\Sigma.b.1$	.2884	.2892	.3204	.3324	.3350	.3395	**.3175**
$\Sigma.b.0$	.2516	.2912	.3233	.3171	.3272	.3404	**.3085**

Table 4: TREC2001 Experimental Results.

hits ($\Sigma.s.1, \Sigma.z.1$) and those which do not count hits ($\Sigma.s.0, \Sigma.z.0$). Therefore, counting hits seems not to improve performance in metasearch;

3. score normalisation performs clearly better than z-score normalisation;

4. rank normalisation performs better than Borda rank normalisation and, in particular, it performs better than the BC method. While the BC showed to be promising in early experiments [1], it works worse at least for metasearch;

5. the $\Sigma.s.1$ method, *i.e.,* score normalisation and counting hits, wins three out of four tests, but its average precision is substantially equal to $\Sigma.s.0$;

6. the $\Sigma.r.0$ method, *i.e.,* rank normalisation and do not counting hits, may considered the winner (wins three out of four tests), and its average precision is highest among the rank based methods;

7. the $\Sigma.s.y$ methods wins over the $\Sigma.r.0$ method (even if in TREC2001 they are very close), in terms of wins and clearly in terms of average precision.

Table 5 contains the average precision over the four tests.

Method	MC_1	MC_2	MC_3	MC_4	$\Sigma.r.1$	$\Sigma.r.0$
Avg	.3025	.2966	.292	.2988	.2752	.2857
Method	$\Sigma.s.1$	$\Sigma.s.0$	$\Sigma.z.1$	$\Sigma.z.0$	$\Sigma.b.1$	$\Sigma.b.0$
Avg	.3017	.3031	.2741	.2702	.2692	.2693

Table 5: Average precision over the tests.

About the Markov chain based methods we observe that:

1. there is no clear winner. MC_1 and MC_4 are first twice, and their average precision is not appreciably different;

2. the Markov chain based methods win over $\Sigma.r.0$ three times out of four, and the average precisions of MC_1 and MC_4 are clearly better than those of rank based combination methods;

3. the average precisions of MC_1 and MC_4 are comparable to those of $\Sigma.s.1$ and $\Sigma.s.0$.

4. CONCLUSIONS

We may summarize our results in the context of metasearch as follows. Counting hits does not improve the effectiveness significantly. Additionally, we have shown that, contrary to what is usually assumed, there are rank based methods, *i.e.,* the Markov chain based methods, whose performance is comparable to score based methods. This is especially important in the context of metasearch as scores are usually not available from the search engines. It is worth noting that the Markov chain based methods do not rely on hits, but do rely on rank comparisons only.

There are some directions for future research, which we are currently investigating; (*i*) to verify whether Markov chain based methods perform well also in other contexts, *e.g.,* testing them also w.r.t. other TREC results; (*ii*) to investigate the performance of the methods when training data is available.

5. REFERENCES

[1] A. Aslam, Javed and Mark Montague. Models for metasearch. In ACM SIGIR-01, pages 276–284, 2001.

[2] J.C. Borda. Mémoire sur les élections au scrutin. *Histoire de l'Académie Royal des Sciences*, 1781.

[3] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1-7):107–117, 1998.

[4] Jamie Callan, Zhihong Lu, and Bruce W. Croft. Searching distribute collections with inference networks. In ACM SIGIR-95, pages 21–28, 1995.

[5] Nick Craswell, David Hawking, and Paul Thistlewaite. Merging results from isolated search engines. In *10th Australian Database Conf.*, 1999.

[6] Daniel Dreilinger and Adele E. Howe. Experiences with selecting search engines using metasearch. *ACM Transactions on Information Systems*, 15(3):195–222, 1997.

[7] Cynthia Dwork, Ravi Kumar, Moni Noar, and D. Sivakumar. Rank aggregation methods for the web. In *10th International Conf. on the World Wide Web*, pages 613–622. ACM Press and Addison Wesley, 2001.

[8] Ronald Fagin and Edward L. Wimmers. Incorporating user preferences in multimedia queries. In *Proc. of 6th International Conf. on Database Theory*, LNCS 1186, 1997.

[9] Joseph A. Fox, Edward Shaw. Combination of multiple sources: The TREC-2 interactive track matrix experiment. In ACM SIGIR-94, 1994.

[10] Susan Gauch, Guijun Wang, and Mario Gomez. ProFusion: Intelligent fusion from multiple, distributed search engines. volume 2, pages 637–649, 1996.

[11] Steve Lawrence and Lee C. Giles. Inquirus, the NECI meta search engine. *Computer Networks and ISDN Systems*, 30:95–105, 1998.

[12] Joon Ho Lee. Analysis of multiple evidence combination. In ACM SIGIR-97, pages 267–276, 1997.

[13] R. Manmatha, R. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In ACM SIGIR-01, pages 267–275, 2001.

[14] D. G. Saari. The mathematics of voting: Democratic symmetry. *The Economist*, March 4 2000.

[15] Gerard Salton and J. Michael McGill. *Introduction to Modern Information Retrieval*. Addison Wesley Publ. Co., 1983.

[16] E. Selberg and O. Etzioni. The MetaCrawler architecture for resource aggregation on the Web. *IEEE Expert*, (January–February):11–14, 1997.

[17] Christopher C. Vogt and Garrison W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[18] Ellen M. Voorhees, Narendra K. Gupta, and Ben Johson-Laird. The collection fusion problem. In D.K. Harman, editor, *Proc. 3rd Text REtrieval Cconference (TREC-3)*, number 500-225, 1994. National Institute of Standards and Technology.

[19] Ronald R. Yager and Rybalov. On the fusion of documents from multiple collection information retrieval systems. *Journal of the American Society for Information Science*, 13(49):1177–1184, 1998.