Cluster-Based Fusion of Retrieved Lists

Anna Khudyak Kozorovitsky
annak@tx.technion.ac.il

Oren Kurland
kurland@ie.technion.ac.il

Faculty of Industrial Engineering and Management
Technion — Israel Institute of Technology
Haifa 32000, Israel

ABSTRACT

Methods for fusing document lists that were retrieved in response to a query often use retrieval scores (or ranks) of documents in the lists. We present a novel probabilistic fusion approach that utilizes an additional source of rich information, namely, inter-document similarities. Specifically, our model integrates information induced from clusters of similar documents created across the lists with that produced by *some* fusion method that relies on retrieval scores (ranks). Empirical evaluation shows that our approach is highly effective for fusion. For example, the performance of our model is consistently better than that of the standard (effective) fusion method that it integrates. The performance also transcends that of standard fusion of re-ranked lists, where list re-ranking is based on clusters created from documents in the list.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval]: Retrieval models

General Terms: Algorithms, Experimentation

Keywords: ad hoc retrieval, fusion, cluster-based fusion

1. INTRODUCTION

Fusing document lists that were retrieved from a corpus in response to a query so as to compile a single result list is a long studied task [9]. The lists are often produced by using multiple query representations, document representations or ranking functions [9]. Many effective fusion methods are based on the premise that documents that are highly ranked in many of the lists are likely to be relevant [13, 23, 9, 12, 2, 29].

However, different retrieved lists can contain different relevant documents [10, 14, 5, 3]. Hence, fusion methods based on the premise just mentioned can fall short in such scenarios. A case in point, a relevant document that appears only in one list, and which is ranked low in this list, will be ranked low in the final result list. Nevertheless, if the document content is *similar* to that of other relevant documents — as implied by the *cluster hypothesis* [42] — and those are highly ranked in (many of) the lists, then the document can be "rewarded".

Accordingly, we present a novel probabilistic approach to fusion that lets similar documents across the lists provide relevance-status support to each other. Our model integrates information produced by *some* standard fusion method, which relies on retrieval scores (ranks) of documents in the lists, with that induced from *clusters* that are created from similar documents across the lists.

Empirical evaluation performed using TREC data demonstrates the effectiveness of our model. For example, the model posts performance that is consistently better than that of the standard fusion method that it integrates for three such effective fusion methods.

Our approach is also more effective than standard fusion of cluster-based re-ranked lists. In other words, re-ranking each list using clusters created from documents in the list so as to improve relevance estimates in the list [45, 25, 21]; and then, using standard fusion to aggregate the re-ranked lists, is shown to be less effective than our approach that uses clusters created from documents across the lists. We further demonstrate the merits of using across-list created clusters by showing that they can contain a (much) higher percentage of relevant documents than that in clusters created from each list separately.

Conceptually, our contributions are twofold. From a fusion perspective, we show that using information induced from clusters created from documents across the lists can help to substantially improve over methods that use only retrieval scores/rank information. From a cluster-based retrieval perspective, we address the multiple retrieved lists setting in which documents may appear in several lists with different retrieval scores. In contrast, previous work on cluster-based retrieval addressed the single retrieved list setting [45, 25, 21, 26, 46, 27, 18, 31].

2. FUSION FRAMEWORK

Let q, d, and $\mathcal{C}$ denote a query, a document, and a corpus of documents respectively. We assume that the document lists $L_1, \ldots, L_m$, each of which contains k documents, were retrieved in response to q by m retrievals performed over $\mathcal{C}$. The retrievals could be based, for example, on different representations of q, different document representations and/or different ranking functions [9]. We use $\mathcal{C}_L \stackrel{def}{=} \bigcup_i L_i$ to denote the *set* of documents that appear in the lists.

Our goal is to utilize information from the lists $L_1, \ldots, L_m$ so as to assign a positive *fusion* score, $F(d; q)$, to d ($\in \mathcal{C}_L$);

SIGIR'11, July 24–28, 2011, Beijing, China.

we set $F(d;q) \stackrel{def}{=} 0$ for $d \notin \mathcal{C}_L$. The score should reflect d's presumed relevance to the information need underlying q.

Standard fusion methods assign $F(d;q)$ based on the retrieval scores (or ranks) of d in the lists in which it appears [9]. Specifically, documents that are highly ranked in many of the lists are often "rewarded" [13]. For example, if $S_{L_i}(d)$ is d's positive retrieval score in L_i, and assuming that $S_{L_i}(d) \stackrel{def}{=} 0$ if $d \notin L_i$, then the **CombSUM** fusion method [13] scores d by the sum of its retrieval scores:

$$F_{CombSUM}(d;q) \stackrel{def}{=} \sum_{L_i} S_{L_i}(d).$$

The **CombMNZ** method [13, 22] further rewards documents that appear in many lists:

$$F_{CombMNZ}(d;q) \stackrel{def}{=} \#\{L_i : d \in L_i\} F_{CombSUM}(d;q).$$

The **Borda** method [47], on the other hand, considers only rank information. Specifically, d is scored by the number of documents not ranked higher than it in the lists:

$$F_{Borda}(d;q) \stackrel{def}{=} \sum_{L_i} \#\{d' \in L_i : S_{L_i}(d') <= S_{L_i}(d)\}.$$

2.1 Cluster-based fusion

A rich source of additional information that can be used to estimate d's relevance, and which is not utilized by standard fusion methods, is *inter-document similarities*. For example, suppose that d's content is highly similar to that of many documents that appear in the highest ranks of the lists. Presumably, then, d and these documents discuss the same topics/aspects. Hence, d should potentially be rewarded even if, in the extreme case, it appears in only one list in which it is ranked low. On the other hand, fusion methods that rely only on retrieval scores (ranks) will "penalize" d and rank it low in the final result list.

Thus, inspired by the *cluster hypothesis* [42], which states that "closely associated documents tend to be relevant to the same requests", we opt to devise a fusion method that lets similar documents across the lists provide relevance-status support to each other. To that end, we consider a set $Cl(\mathcal{C}_L)$ of document *clusters* created from $\mathcal{C}_L$ (the set of all documents in the lists) using *some* clustering algorithm; c will be used to denote a cluster. These clusters could potentially be viewed as representing query related "aspects", by the virtue of the way they are created, that is, from documents retrieved in response to the query. Hence, our goal becomes integrating cluster-based information with information used by standard fusion methods — retrieval scores and/or ranks of documents in the lists.

2.1.1 Using clusters as proxies for documents

Our goal is to estimate $p(d|q)$, the probability that d is relevant to the information need underlying q. Inspired by work on cluster-based retrieval in the language-modeling framework [20, 19], we use clusters as proxies for documents to estimate this probability. However, in contrast to these approaches [20, 19], which focus on a *single* retrieved list, we do not use language-model estimates for probabilities of the form $p(x|y)$ in the derivation to follow. Rather, we exploit the rich information provided by the occurrences and retrieval scores of documents in the lists to devise estimates. Thus, the resultant cluster-based fusion model that we derive is different — in the use of probabilities and in the formation of their estimates — from the single-list models by which its development was inspired.

Model derivation. We can write the probability for d's relevance as:

$$p(d|q) = \sum_{c \in Cl(\mathcal{C}_L)} p(d|c,q)p(c|q). \quad (1)$$

To estimate $p(d|c,q)$, we use a linear mixture governed by a free parameter λ: $\hat{p}(d|c,q) \stackrel{def}{=} (1-\lambda)p(d|q) + \lambda p(d|c)$ [20]; we use the notation $\hat{p}(\cdot)$, here and after, to denote estimates of (model) probabilities. Using $\hat{p}(d|c,q)$ in Equation 1, along with assuming that $p(c|q)$ is a probability distribution over $Cl(\mathcal{C}_L)$ — the universe of clusters that we consider, and then applying some probabilistic algebra, results in our **ClustFuse** algorithm:

$$F_{ClustFuse}(d;q) \stackrel{def}{=} (1-\lambda)p(d|q) + \lambda \sum_{c \in Cl(\mathcal{C}_L)} p(c|q)p(d|c). \quad (2)$$

ClustFuse uses a two component mixture model to score d. The first is the original probability of relevance, $p(d|q)$, from which we back off to using cluster-based information. The second component, which uses clusters as proxies for documents, "rewards" d if it is strongly associated with clusters c (as measured by $p(d|c)$) that presumably contain a high percentage of information pertaining to q (as measured by $p(c|q)$).

The remaining task for instantiating a specific algorithm from Equation 2 is deriving estimates for $p(d|q)$, $p(c|q)$, and $p(d|c)$. To that end, we assume *some* standard fusion method (e.g., one of those mentioned above, CombSUM, CombMNZ, or Borda) that assigns d with a score $F(d;q)$ that is based on d's retrieval scores (ranks) in the lists. The estimates we present are not committed to a specific standard fusion method.

Estimates. Using probability rules, we can write: $p(d|q) = \frac{p(q|d)p(d)}{p(q)}$; and, $p(q) = \sum_{d' \in \mathcal{C}} p(q|d')p(d')$. Assuming a uniform prior distribution for documents ($p(d)$), we get that $p(d|q) = \frac{p(q|d)}{\sum_{d' \in \mathcal{C}} p(q|d')}$. We use an estimate $\hat{p}(q|d) \propto F(d;q)$, where $F(d;q)$ is d's standard fusion score. Recall that $F(d;q)$ is by definition 0 for documents d not in $\mathcal{C}_L$. Thus, we get that

$$\hat{p}(d|q) \stackrel{def}{=} \frac{F(d;q)}{\sum_{d' \in \mathcal{C}_L} F(d';q)} \quad (3)$$

is d's normalized standard-fusion score. Note that $\hat{p}(d|q)$ is a probability distribution over the *entire* corpus.

Similarly, assuming a uniform prior for clusters, the probability that cluster c contains information pertaining to q can be written as: $p(c|q) = \frac{p(q|c)}{\sum_{c' \in Cl(\mathcal{C}_L)} p(q|c')}$. Now, we would like to exploit the fact that c contains documents that might have multiple appearances in the lists to be fused. Hence, we estimate $p(q|c)$ based on the standard-fusion scores of c's constituent documents. Recent work on representing sets of documents, specifically, clusters of similar documents, has demonstrated the merits of a product-based (geometric) representation [27, 31]. Such representation was advocated

using arguments based on information geometry [31]. Accordingly, we set

$$\hat{p}(c|q) \stackrel{def}{=} \frac{\prod_{d \in c} F(d;q)}{\sum_{c' \in Cl(\mathcal{C}_L)} \prod_{d' \in c'} F(d';q)}; \quad (4)$$

$\hat{p}(c|q)$ is a probability distribution over $Cl(\mathcal{C}_L)$. As all clusters we use in the evaluation in Section 4 contain the same number of documents, there is no bias incurred by using the product of fusion scores of documents in a cluster.[1]

To estimate $p(d|c)$, the document-cluster association strength, we, again, assume a uniform document prior. Thus, we get that $p(d|c) = \frac{p(c|d)}{\sum_{d_i \in \mathcal{C}} p(c|d_i)}$. Now, to estimate $p(c|d)$, we set $\hat{p}(c|d) \stackrel{def}{=} 0$ for $d \notin \mathcal{C}_L$. That is, we assume no association between documents not in $\mathcal{C}_L$ and clusters of documents from $\mathcal{C}_L$. This assumption echoes those used in work on re-ranking a single retrieved list based on clusters of documents in the list [45, 25, 21]. (Refer to Section 3 for further details.) Our next task is then to estimate $p(c|d)$ for $d \in \mathcal{C}_L$. To that end, we use the mean inter-document-similarity between d and c's constituent documents: $\hat{p}(c|d) \propto \frac{1}{|c|} \sum_{d' \in c} sim(d', d)$; $|c|$ is the number of documents in c; $sim(\cdot, \cdot)$ is the inter-document-similarity estimate used to create the clusters. (See Section 4.1 for details). Thus, we arrive to:

$$\hat{p}(d|c) \stackrel{def}{=} \frac{\sum_{d' \in c} sim(d', d)}{\sum_{d_i \in \mathcal{C}_L} \sum_{d' \in c} sim(d', d_i)}, \quad (5)$$

where $d \in \mathcal{C}_L$. As a result, $\hat{p}(d|c)$ is a probability distribution over the corpus.

It is important to note that d ($\in \mathcal{C}_L$) does not have to be a member of c to have a non-zero association strength, $\hat{p}(d|c)$, with c. Thus, even if we use a hard clustering technique, documents could be deemed associated (to some degree) with clusters to which they do not belong, yielding a soft clustering effect. The merits of such association approach were demonstrated in work on cluster-based retrieval [19].

ClustFuse in a nutshell. Using the estimates just described in Equation 2 yields the following fusion principle, which addresses the motivation for utilizing inter-document similarities. Document d is rewarded based on its: (i) standard-fusion score ($F(d;q)$), which reflects the extent to which d is highly ranked in (many of) the lists, and (ii) similarity ($\hat{p}(d|c)$) to clusters c that contain documents highly ranked in many of the lists (as measured by $\hat{p}(c|q)$).

We use **ClustFuseCombSUM**, **ClustFuseCombMNZ**, and **ClustFuseBorda** to denote the implementations of ClustFuse with the standard fusion methods described above. For $\lambda = 0$, ClustFuseCombSUM, ClustFuseCombMNZ, and ClustFuseBorda amount to CombSUM, CombMNZ, and Borda, respectively (see Equations 2 and 3); i.e., ClustFuse reduces to the standard fusion method that it utilizes. Higher values of λ result in more weight put on cluster-based information.

[1]We found that using the product of retrieval scores of documents in a cluster in Equation 4 yields fusion performance that is somewhat better than that of using the geometric mean of the scores. The performance is also superior to that of using the arithmetic mean (or sum) of the scores. The latter finding is in line with reports on using clusters from a single retrieved list to re-rank it [27, 31].

2.1.2 Ranking clusters

The ClustFuse method utilizes the estimate $\hat{p}(c|q)$ from Equation 4 for the probability that cluster c contains information pertaining to q. Naturally, the estimate should be high for clusters containing a high percentage of relevant documents. To study the extent to which $\hat{p}(c|q)$ reflects this percentage in c, we take an approach that was used in work on ranking clusters created from a single list [26, 21, 18].

We use $\hat{p}(c|q)$ to rank the clusters in $Cl(\mathcal{C}_L)$. Then, *all* δ documents of the *highest* ranked cluster are positioned at the top of the result list. (All clusters contain the same number, δ, of documents.) To neutralize within-cluster ordering effects, we only use the precision at δ measure (p@δ) to evaluate the resultant retrieval performance; this precision is the percentage of relevant documents in the highest ranked cluster. We use **ClustRank** to denote this retrieval approach, which, formally, scores d with 1 if d is a member of the highest ranked cluster, and with 0 otherwise; **ClustRankCombSUM**, **ClustRankCombMNZ**, and **ClustRankBorda** denote the implementations of ClustRank with the standard fusion methods.

3. RELATED WORK

Most fusion approaches utilize retrieval scores or ranks of documents but not the document content [13, 44, 23, 1, 9, 12, 2, 29, 24, 34]. Our ClustFuse method, which can incorporate these approaches, utilizes also information induced from clusters of similar documents. In Section 4.2 we show that ClustFuse outperforms the standard fusion method it incorporates; specifically, CombSUM, CombMNZ and Borda.

Clusters of documents can potentially be created based on document *snippets*, rather than the entire document content, if the content is not (quickly) available [48]. Document snippets, and other content-based features, were used for fusion [6, 41, 4, 30, 33], but inter-snippet (document) similarities were not utilized in these approaches.

Similarities between document titles were used for merging lists retrieved from non-overlapping corpora in response to a query [36]. In contrast to our approach, which operates on a single corpus, retrieval scores were not integrated with these similarities, and clusters were not utilized. Document clusters were also used for selecting terms for query expansion in federated search [35]. In contrast, ClustFuse uses clusters to rank documents based on the original query. Inter-document similarities were also used to re-rank a retrieved list using a second retrieved list [28]. However, clusters were not utilized. Furthermore, this asymmetric fusion (re-ranking) approach cannot be naturally extended to fusing several retrieved lists [28].

There is work studying the *potential* (e.g., for improving result interfaces) of using clusters created from documents retrieved from several collections [7]. Specifically, cluster-based and document-based retrieval are contrasted wherein the former is based on using clusters that are known to contain a high percentage of relevant documents. In Section 4.2 we present a similar study applied for the single corpus case that we address here. In contrast to our work, a specific document (or cluster) ranking method was not proposed [7].

There is recent work on using a graph-based approach for fusion that utilizes inter-document similarities [16]. In contrast to ClustFuse, document clusters were not used. In Section 4.2 we further discuss this work, and demonstrate

the performance merits of ClustFuse with respect to the proposed graph-based method.

The clusters used by ClustFuse are *query specific* [45], as they are created from documents retrieved in response to the query. Previous work utilizing query-specific clusters has focused on re-ranking a single retrieved list using information induced from clusters of documents *within* the list [45, 25, 21, 26, 46, 27]. In contrast, ClustFuse fuses lists using information induced from clusters created from documents *across* the lists. Furthermore, ClustFuse also exploits the fact that documents might have multiple occurrences in the lists as is manifested in their standard-fusion scores.

We note that ClustFuse can, in fact, be used to re-rank a *single* retrieved list. The standard-fusion document score used by ClustFuse in this case is simply the retrieval score of the document in the list. This specific single list implementation echoes a previous cluster-based ranking method [20]. However, while this method [20] uses language-model-based estimates for the document-query and cluster-query "match", ClustFuse utilizes retrieval scores, however induced, for these estimates. Furthermore, we show in Section 4.2 that independently re-ranking each of the retrieved lists using this variant of ClustFuse, and then using standard fusion to merge the re-ranked lists, yields performance that is inferior to that of the original ClustFuse implementation, which exploits information induced from across-list clusters.

There has been some work on identifying clusters of documents from the same retrieved list that contain a high percentage of relevant documents [25, 21, 26, 27, 18, 31]. The estimate used by ClustFuse for the amount of query pertaining information in a cluster, which we explore as a cluster-based fusion method at its own right (ClustRank), is reminiscent of some of these methods [27, 18]; specifically, by the virtue of using the document-query "match" of the cluster's constituent documents. However, in contrast to previous methods, ClustFuse uses the standard-fusion score of documents rather than a retrieval score in a single list. Furthermore, while it was shown that there are clusters containing a high relevant-document percentage when created from a single retrieved list [15, 40, 32], we show in Section 4.2 that clusters created across multiple retrieved lists can contain a much higher relevant-document percentage.

4. EVALUATION

4.1 Experimental setup

We measure inter-document similarities using a previously proposed language-model-based estimate that was shown to be effective [20, 21]. Specifically, let $p_d^{[\mu]}(\cdot)$ denote the unigram, Dirichlet-smoothed, language model induced from document d; μ is the smoothing parameter which is set to 1000 [49]. The similarity between documents d_1 and d_2 is defined using the KL divergence:

$$sim(d_1, d_2) \stackrel{def}{=} \exp\left(-KL\left(p_{d_1}^{[0]}(\cdot) \,\middle\|\, p_{d_2}^{[\mu]}(\cdot)\right)\right).$$

To cluster the *set* $\mathcal{C}_L$ of documents in the retrieved lists, we use a simple nearest-neighbors-based approach [14, 20, 21, 26, 39, 27]. Some previous work [19] demonstrated the merits of using this clustering approach with respect to other clustering methods for cluster-based document retrieval using a single retrieved list. For each d ($\in \mathcal{C}_L$) we define a cluster that contains d and the $\delta - 1$ documents d' in $\mathcal{C}_L$ ($d' \neq d$) that yield the highest $sim(d, d')$; note that the resultant clusters overlap. As relatively small clusters are known to be most effective for cluster-based retrieval [20, 21, 39, 27], we set $\delta = 10$, unless otherwise specified. (The performance of ClustFuse with $\delta = 5$ was inferior to that of using $\delta = 10$.)

We use TREC data for experiments: (i) the ad hoc track of trec3 (50 queries; 741,856 documents), which is based on news articles; (ii) the Web track of trec10 (50 queries; 1,692,096 documents); and, (iii) the robust track of trec12 (50 queries; 528,155 documents), which is a challenging benchmark [43]. We apply tokenization, Porter stemming, and stopword removal (using the INQUERY list) to documents using the Lemur toolkit[2], which is used for experiments.

Fusion methods are quite effective when the lists to be fused are relatively short [38, 41, 3]. Furthermore, utilizing similarities between top-retrieved documents is most effective when the number of documents considered is relatively small [11, 21]. Hence, to maintain the number of documents to be ranked ($|\mathcal{C}_L|$) relatively small, we fuse three lists, each of which is composed of the k ($= 20$) highest ranked documents in a submitted run in a track. In Section 4.2.3 we show that the resultant relative performance patterns are quite consistent for $k \in \{10, 20, 30, 40, 50\}$. The three runs to be fused, unless otherwise stated, are randomly selected from *all* submitted runs in a track (both automatic and manual). We use **run1**, **run2**, and **run3**, to denote the runs in descending order of MAP(@k) performance. We use 20 samples of randomly selected triplets of runs and report average performance over the samples. In Sections 4.2.6 and 4.2.7 we study the case of fusing the three most effective runs in a track.

The CombSUM and CombMNZ methods, utilized by the ClustFuse and ClustRank algorithms, require inter-list compatibility of retrieval scores. To that end, we normalize the retrieval score of a document in a list with respect to the sum of all retrieval scores in the list. If retrieval scores are negative, which is due to using logs, we use the exponent of a score for normalization,

To evaluate retrieval performance, we use MAP(@k) and the precision of the top 5 and 10 documents (p@5 and p@10, respectively). Statistically significant differences of performance are determined using the two-tailed paired t-test at a 95% confidence level [37].[3]

The ClustFuse method incorporates a single free parameter, λ. (ClustRank does not incorporate free parameters.) The value of λ ($\in \{0, 0.1, \ldots, 1\}$) is set using leave-one-out cross validation performed over the entire set of queries in a track; performance is optimized in the learning phase with respect to MAP. In other words, the performance for a query is that attained using a value of λ that maximizes MAP performance over all other queries in the track.

A note on efficiency. The computational overhead posted by our methods with respect to standard fusion approaches is not significant. The clustering of a few dozen documents from the retrieved lists can be performed quickly (e.g., based on document snippets) as was shown in work on clustering

[2] www.lemurproject.org

[3] A statistical significance test for a pair of methods in the case of 20 random samples of triplets of runs is employed upon the average (over the 20 samples) performance of the methods for the queries.

	trec3			trec10			trec12		
	MAP	p@5	p@10	MAP	p@5	p@10	MAP	p@5	p@10
run1	8.4	68.0	64.9	14.1	39.7	34.4	25.2	48.4	42.2
run2	6.5	57.2	54.9	10.4	33.5	29.3	20.8	42.4	36.7
run3	4.3	41.7	40.7	6.3	22.5	19.9	13.2	29.7	25.2
CombSUM	7.4^a	65.3	60.8^a	12.3	38.9	33.9	22.7^a	46.9^a	39.9^a
ClustFuseCombSUM	$\mathbf{9.7}^a_f$	$\mathbf{75.3}^a_f$	$\mathbf{71.7}^a_f$	$\mathbf{14.1}_f$	$\mathbf{42.4}_f$	$\mathbf{37.1}^a_f$	$\mathbf{25.0}_f$	$\mathbf{49.1}_f$	**43.2**
CombMNZ	7.4^a	65.4	60.9^a	12.7	40.1	34.2	22.7^a	46.9^a	39.9^a
ClustFuseCombMNZ	$\mathbf{9.6}^a_f$	$\mathbf{75.2}^a_f$	$\mathbf{71.1}^a_f$	$\mathbf{14.3}_f$	$\mathbf{42.9}_f$	$\mathbf{37.8}^a_f$	$\mathbf{24.5}_f$	**47.9**	$\mathbf{42.3}_f$
Borda	7.0^a	64.0^a	58.7^a	12.4	39.8	32.9	22.3^a	47.0^a	39.6^a
ClustFuseBorda	$\mathbf{9.7}^a_f$	$\mathbf{74.1}^a_f$	$\mathbf{70.8}^a_f$	$\mathbf{14.3}_f$	$\mathbf{43.5}^a_f$	$\mathbf{37.1}^a_f$	$\mathbf{25.1}_f$	$\mathbf{49.8}_f$	$\mathbf{44.0}^a_f$

Table 1: Main result. Boldface marks the better performing between ClustFuseF and the standard fusion method F that it incorporates; a statistically significant difference between the two is marked with 'f'. Underline marks the best result per column; 'a' marks a statistically significant difference with run1. All performance differences between the fusion methods and run2 and run3 are statistically significant.

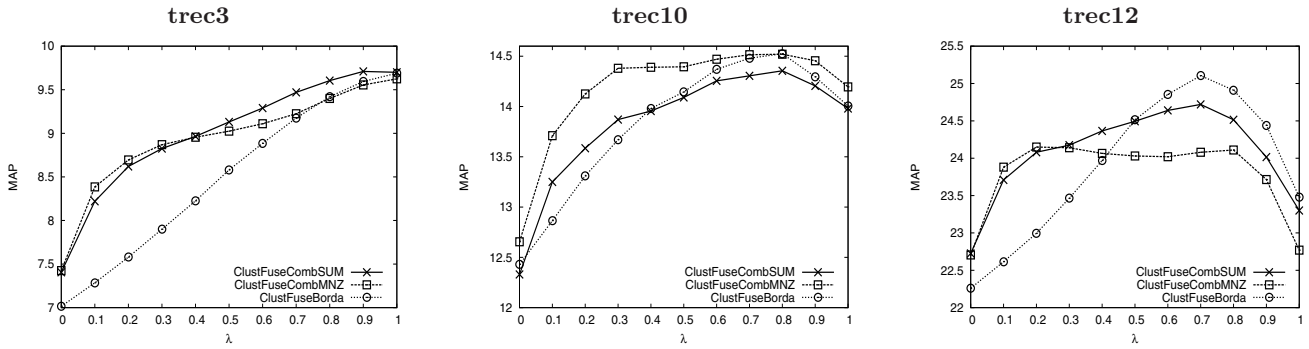

Figure 1: Effect of varying λ on the MAP performance of ClustFuse. For $\lambda = 0$, ClustFuseF amounts to F, the standard fusion method that it incorporates; higher value of λ corresponds to more weight put on cluster-based information. Note: figures are not to the same scale.

the results of Web search [48]; our methods are not committed to a specific clustering approach. Similar efficiency considerations were echoed in work on cluster-based re-ranking of a single document list [45, 25, 21]; and, in work on fusion based on document content [41, 36].

4.2 Experimental results

4.2.1 Main result

In Table 1 we present the performance numbers of ClustFuse. We can see that ClustFuse outperforms the standard fusion method that it incorporates in all relevant comparisons (track × evaluation measure). Many of these improvements are substantial and statistically significant. It is also worth noting that in many cases for trec10 the standard fusion methods underperform run1 (the most effective of the three runs), while ClustFuse almost always improves over run1 for trec10; sometimes, to a statistically significant degree. Moreover, in the very few cases that ClustFuse is outperformed by run1 (mainly, MAP for trec12), these performance differences are not statistically significant. The standard fusion methods, on the other hand, post in all cases for trec12 performance that is statistically significantly worse than that of run1. These findings attest to the merits of ClustFuse that integrates information induced from document clusters (created across the lists) with retrieval scores (or rank) information used by the standard fusion methods.

4.2.2 The effect of using cluster-based information

We next explore the effect of varying the value of λ on the performance of ClustFuse. (Refer back to Equation 2.) For $\lambda = 0$, ClustFuse amounts to the standard fusion method that it incorporates; higher values of λ correspond to more weight put on cluster-based information. For $\lambda > 0$ standard fusion scores of documents are used (also) for estimating the amount of query-pertaining information in a cluster. Figure 1 depicts the MAP-performance curves.

We see in Figure 1 that for $\lambda > 0$, and for all three tracks, the performance of ClustFuse is better than that of the standard fusion method that it incorporates, i.e., ClustFuse with $\lambda = 0$. Even for $\lambda = 0.1$, which amounts to assigning a relatively small weight to cluster-based information, some of the performance gains over not using this information ($\lambda = 0$) are substantial; furthermore, the improvements are quite large for $\lambda \in \{0.6, 0.7, 0.8\}$. Further increasing λ can result in performance decrease (e.g., for trec10 and trec12) that attests to the importance of integrating the standard fusion score assigned to the document with information induced from clusters to which it is similar.

4.2.3 The effect of list size

Insofar, we had our methods, and the reference comparisons, fuse lists of $k = 20$ documents each. We now turn to study the effect of the list size, k, on fusion performance. As noted above, fusion methods, so as methods that utilize inter-document similarities, are quite effective when operating over relatively short lists. In our case, the size of the

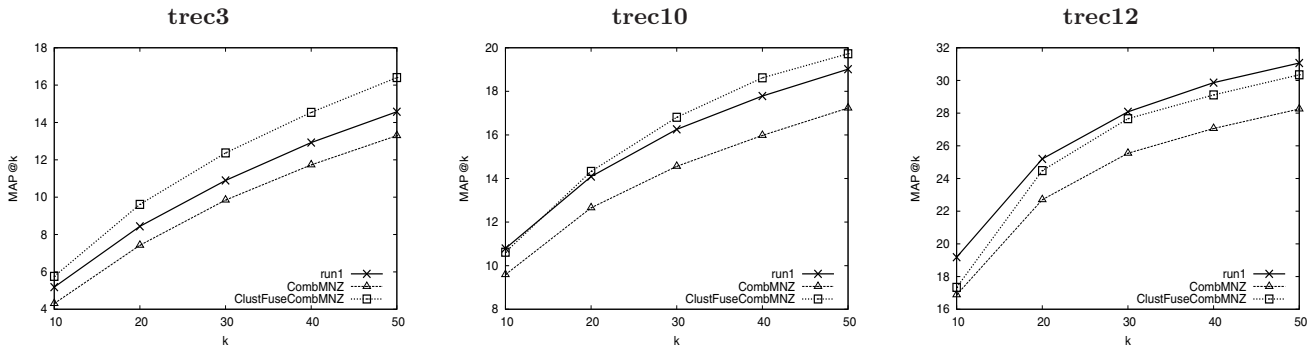

Figure 2: Effect of the size of the lists to be fused, k, on MAP@k. Note: figures are not to the same scale.

	trec3			trec10			trec12		
	MAP	p@5	p@10	MAP	p@5	p@10	MAP	p@5	p@10
run1	8.4	68.0	64.9	14.1	39.7	34.4	25.2	48.4	42.2
Clust(run1)	8.8^{a}	72.2^{a}	68.2^{a}	14.1	40.6	36.1^{a}	24.6^{a}	47.6^{a}	42.0
run2	6.5	57.2	54.9	10.4	33.5	29.3	20.8	42.4	36.7
Clust(run2)	7.3^{b}	69.1^{b}	63.4^{b}	10.4	35.9^{b}	31.3^{b}	21.1	43.1	37.7^{b}
run3	4.3	41.7	40.7	6.3	22.5	19.9	13.2	29.7	25.2
Clust(run3)	5.1^{c}	54.3^{c}	49.7^{c}	6.5	24.5^{c}	21.1^{c}	13.7^{c}	31.8^{c}	27.0^{c}
CombSUM(Clust(run{i}))	8.2^{bc}	70.0^{bc}	64.9^{bc}	12.3^{bc}	41.3^{bc}	35.2^{bc}	22.4^{abc}	46.7^{abc}	40.0^{abc}
ClustFuseCombSUM	$\mathbf{9.7}_{s}^{abc}$	$\mathbf{75.3}_{s}^{abc}$	$\mathbf{71.7}_{s}^{abc}$	$\mathbf{14.1}_{s}^{bc}$	$\mathbf{42.4}^{bc}$	$\mathbf{37.1}_{s}^{abc}$	$\mathbf{25.0}_{s}^{bc}$	$\mathbf{49.1}_{s}^{bc}$	$\mathbf{43.2}^{bc}$
CombMNZ(Clust(run{i}))	8.2^{bc}	69.4^{bc}	64.5^{bc}	12.7^{bc}	41.5^{bc}	35.4^{bc}	22.4^{abc}	46.8^{abc}	40.0^{abc}
ClustFuseCombMNZ	$\mathbf{9.6}_{s}^{abc}$	$\mathbf{75.2}_{s}^{abc}$	$\mathbf{71.1}_{s}^{abc}$	$\mathbf{14.3}_{s}^{bc}$	$\mathbf{42.9}^{bc}$	$\mathbf{37.8}_{s}^{abc}$	$\mathbf{24.5}_{s}^{bc}$	$\mathbf{47.9}_{s}^{bc}$	$\mathbf{42.3}_{s}^{bc}$
Borda(Clust(run{i}))	8.3^{bc}	69.8^{bc}	64.9^{bc}	12.4^{bc}	41.1^{bc}	34.4^{bc}	21.9^{abc}	46.8^{abc}	40.0^{abc}
ClustFuseBorda	$\mathbf{9.7}_{s}^{abc}$	$\mathbf{74.1}_{s}^{abc}$	$\mathbf{70.8}_{s}^{abc}$	$\mathbf{14.3}_{s}^{bc}$	$\mathbf{43.5}_{s}^{abc}$	$\mathbf{37.1}_{s}^{abc}$	$\mathbf{25.1}_{s}^{bc}$	$\mathbf{49.8}_{s}^{bc}$	$\mathbf{44.0}_{s}^{abc}$

Table 2: Comparison of cluster-based fusion of runs (ClustFuseF) with standard fusion of cluster-based re-ranked runs (F(Clust(run{i}))); the performance of the better performing of the two is boldfaced, and 's' indicates that the difference is statistically significant. Underline marks the best result per column; 'a', 'b', and 'c' mark statistically significant differences with run1, run2, and run3, respectively.

document set to be ranked ($|\mathcal{C}_L|$) is determined by the number of lists to be fused (three), the number of documents in each list (k), and the overlap between the lists. In Figure 2 we present the MAP@k performance of ClustFuseCombMNZ, and that of the standard fusion method that it integrates (CombMNZ), where $k \in \{10, 20, 30, 40, 50\}$. (We use ClustFuseCombMNZ as a representative, as it is the focus of a comparison with some previous work on utilizing inter-document similarities for fusion that we present in Section 4.2.7.) For reference, we present the performance of run1 — the most MAP@k-effective run among the three to be fused.

As we can see in Figure 2, ClustFuseCombMNZ outperforms CombMNZ for all values of k over all three tracks; the relative improvements are often quite substantial. Furthermore, while CombMNZ underperforms (to quite a substantial degree) run1 for all values of k and for all three tracks, ClustFuseCombMNZ improves over run1 for trec3 and trec10 (but slightly underperforms it for trec12). These findings further attest to the benefits in using cluster-based information for fusion. We also note that the overall performance patterns are somewhat similar for $k \geq 20$; yet, the relative improvement of ClustFuseCombMNZ over CombMNZ for $k = 20$ can be somewhat lower than that for higher values of k (e.g., for trec10). Thus, we will continue to focus in the evaluation to follow on $k = 20$ so as to present a conservative picture of the performance our approach.

4.2.4 *Cluster-based fusion vs. fusion of cluster-based re-ranked lists*

Previous work on cluster-based retrieval has focused on the single retrieved list case. Specifically, there is much work on *re-ranking* a retrieved list using clusters of documents in the list [45, 25, 21, 46, 27, 19]. Our approach, on the other hand, fuses several retrieved lists using clusters created across the lists. Thus, we next explore the merits for fusion of using information induced from clusters created across the lists with respect to that induced from clusters created from each list. To that end, we study a fusion paradigm that is based, in spirit, on principles underlying the cluster-based re-ranking approaches just mentioned.

First, we cluster the documents in *each* run. Then, we use ClustFuse to *re-rank* a run using its clusters.[4] The motivation is to improve the relevance estimates in a run; the resultant re-ranking of run$i$ is denoted Clust(run$i$). The re-ranked runs are then fused using one of the standard fusion methods F ($\in$ {CombSUM,CombMNZ,Borda}); the fusion results are denoted F(Clust(run{i})). The resultant performance is compared with that of ClustFuseF — our original fusion methods that utilize clusters created across the runs. The performance numbers, all based on a leave-one-

[4]ClustFuse only requires a retrieval score for each document, $F(d; q)$, and information about clusters and inter-document similarities. When employed over a single list, ClustFuse echoes a previous cluster-based (re-)ranking method [20].

	trec3		trec10		trec12	
	p@5	p@10	p@5	p@10	p@5	p@10
OptCluster	93.6	86.4	64.4	50.7	72.8	58.1
run1	68.0	64.9	39.7	34.4	48.4	42.2
OptCluster(run1)	88.4^{a}	79.1^{a}	57.2^{a}	43.4^{a}	66.1^{a}	51.4^{a}
run2	57.2	54.9	33.5	29.3	42.4	36.7
OptCluster(run2)	83.3^{b}	71.9^{b}	50.3^{b}	37.5^{b}	60.0^{b}	45.5^{b}
run3	41.7	40.7	22.5	19.9	29.7	25.2
OptCluster(run3)	69.2^{c}	57.2^{c}	38.7^{c}	27.8^{c}	45.3^{c}	32.7^{c}
CombSUM	65.3^{bc}	60.8^{abc}	38.9^{bc}	33.9^{bc}	46.9^{abc}	39.9^{abc}
ClustRankCombSUM	68.3_{f}^{bc}	65.7_{f}^{bc}	40.6^{bc}	34.6^{abc}	46.8^{bc}	40.7^{bc}
ClustFuseCombSUM	$\mathbf{75.3}_{f}^{abc}$	$\mathbf{71.7}_{f}^{abc}$	$\mathbf{42.4}_{f}^{bc}$	$\mathbf{37.1}_{f}^{abc}$	$\mathbf{49.1}_{f}^{bc}$	$\mathbf{43.2}_{f}^{bc}$
CombMNZ	65.4^{bc}	60.9^{abc}	40.1^{bc}	34.2^{bc}	46.9^{abc}	39.9^{abc}
ClustRankCombMNZ	68.6_{f}^{bc}	65.9_{f}^{bc}	41.5^{bc}	35.2^{bc}	46.3^{abc}	40.5^{abc}
ClustFuseCombMNZ	$\mathbf{75.2}_{f}^{abc}$	$\mathbf{71.1}_{f}^{abc}$	$\mathbf{42.9}_{f}^{bc}$	$\mathbf{37.8}_{f}^{abc}$	$\mathbf{47.9}^{bc}$	$\mathbf{42.3}_{f}^{bc}$
Borda	64.0^{abc}	58.7^{abc}	39.8^{bc}	32.9^{bc}	47.0^{abc}	39.6^{abc}
ClustRankBorda	67.5_{f}^{bc}	65.7_{f}^{bc}	42.8_{f}^{abc}	35.0_{f}^{bc}	47.7^{bc}	42.1_{f}^{bc}
ClustFuseBorda	$\mathbf{74.1}_{f}^{abc}$	$\mathbf{70.8}_{f}^{abc}$	$\mathbf{43.5}_{f}^{abc}$	$\mathbf{37.1}_{f}^{abc}$	$\mathbf{49.8}_{f}^{bc}$	$\mathbf{44.0}_{f}^{abc}$

Table 3: The percentage of relevant documents in the cluster ranked highest by ClustRank. OptCluster and OptCluster(run$i$) are the clusters containing the highest relevant-document percentage among those created across the lists, and those created from run$i$, respectively. The performance of the fusion methods from Table 1 is presented for reference. Boldface marks the best performance in a block of methods. Statistically significant differences with run1, run2 and run3 and the standard fusion method (CombSUM, CombMNZ and Borda) are marked with 'a', 'b', 'c', and 'f', respectively.

out cross-validation performed for λ as described above, are presented in Table 2.

Our first observation based on Table 2 is that ClustFuse is somewhat effective in re-ranking a single list. Indeed, in a majority of the relevant comparisons, the performance of Clust(run$i$) is better than that of run$i$; sometimes, the differences are also statistically significant. Now, fusing the re-ranked runs with the standard fusion methods yields additional (often substantial) performance improvements with respect to run2 and run3, but not with respect to run1. (Refer to CombSUM(Clust(run$\{i\}$)), CombMNZ(Clust(run$\{i\}$)), and Borda(Clust(run$\{i\}$)).)

We can also see in Table 2 that the performance of fusing the cluster-based re-ranked runs is always worse — often, to a statistically significant degree — than that of our ClustFuse approach. (Compare CombSUM(Clust(run$\{i\}$)) with ClustFuseCombSUM, CombMNZ(Clust(run$\{i\}$)) with ClustFuseCombMNZ, and Borda(Clust(run$\{i\}$)) with ClustFuseBorda.) Furthermore, there are many cases wherein the ClustFuse-based methods improve over run1 (the best performing run among the three to be fused) in a statistically-significant manner, while fusion of re-ranked runs does not. These findings attest to the merits of utilizing clusters created across the runs as is done by ClustFuse.

4.2.5 *Ranking clusters with ClustRank*

The ClustFuse method, which was the focus of the evaluation insofar, utilizes an estimate for the presumed amount of query-pertaining information in a cluster. (Refer back to Equations 2 and 4.) The estimate is based on the standard-fusion scores of documents in the cluster. To evaluate the quality of this estimate, we devised ClustRank. This method ranks clusters based on the estimate, and positions the constituent documents of the highest ranked cluster at top of the result list. Hence, for clusters of δ documents the resultant precision at δ (p@δ) of ClustRank is the percentage of relevant documents in the highest ranked cluster. We study the performance of ClustRank with clusters of $\delta = 5$ and $\delta = 10$ documents in Table 3. For reference comparisons, we use **OptCluster** and **OptCluster(run$i$)**: the clusters that contain the highest percentage of relevant documents among those created across the lists, and among those created from run$i$, respectively. In addition, we present the p@δ performance of the standard fusion methods (CombSUM, CombMNZ and Borda) and the ClustFuse implementations.

We can see in Table 3 that the optimal clusters among those created across the lists (OptCluster) contain a very high percentage of relevant documents. If these clusters are automatically identified, then the resultant precision-at-top-ranks performance is by far better than that of all other methods considered. Furthermore, the percentage of relevant documents in these clusters is consistently higher than that in optimal clusters that are created from documents in a run (OptCluster(run$i$)). The latter finding further motivates the use of clusters created across the lists. Yet, identifying the optimal cluster in each run yields much better performance than that of the run, which is in accordance with previous reports on clustering a single retrieved list [18].

We can also see in Table 3 that ClustRank almost always outperforms — and in several cases, statistically significantly so — the standard fusion methods. Furthermore, the highest ranked cluster by ClustRank, which can be composed of documents from the three runs, often contains higher percentage of relevant documents than that in the optimal cluster for run3 (OptCluster(run3)); however, this percentage is still much lower than that in the optimal clusters for run2 and run1. In addition, while ClustRank almost always outperforms run1 for trec3 and trec10 (although statistically significantly so only in a single case), the reverse holds for trec12. As all other fusion methods, ClustRank outperforms run2 and run3 in a statistically significant manner.

Finally, as Table 3 shows, ClustRank is consistently less effective than ClustFuse. This is not surprising as ClustRank uses only the highest ranked cluster, while ClustFuse utilizes information from all clusters, and further integrates this information with standard fusion scores.

All in all, the conclusion rising from the analysis above is the following. While ClustRank is a relatively reasonable

	trec3			trec10			trec12		
	MAP	p@5	p@10	MAP	p@5	p@10	MAP	p@5	p@10
run1	10.4	74.4	72.2	30.7	63.2	58.8	28.8	51.1	44.8
run2	9.6	72.8	67.6	27.7	54.4	50.2	28.4	52.5	48.6
run3	9.5	76.0	71.2	21.6	55.6	46.8	28.1	51.5	45.2
CombSUM	10.9^{bc}	80.8^{ab}	74.6^{b}	37.2^{bc}	$\mathbf{71.2}^{abc}$	61.0^{bc}	30.3	53.7	$\underline{\mathbf{49.2}}^{ac}$
ClustFuseCombSUM	$\underline{\mathbf{11.6}}^{abc}$	$\mathbf{82.4}^{abc}$	$\underline{\mathbf{79.0}}_{f}^{abc}$	$\mathbf{37.3}^{bc}$	70.8^{abc}	$\mathbf{61.8}^{bc}$	$\underline{\mathbf{30.5}}$	**54.3**	49.1^{ac}
CombMNZ	10.9^{bc}	80.8^{ab}	74.6^{b}	37.2^{bc}	$\mathbf{71.2}^{abc}$	61.0^{bc}	30.3	53.9	$\underline{\mathbf{49.2}}^{ac}$
ClustFuseCombMNZ	$\mathbf{11.4}^{abc}$	$\underline{\mathbf{82.8}}^{abc}$	$\mathbf{77.2}^{bc}$	$\underline{\mathbf{38.3}}^{bc}$	70.8^{abc}	$\mathbf{61.8}^{bc}$	$\underline{\mathbf{30.5}}$	**54.1**	49.1^{ac}
Borda	10.7^{bc}	80.0^{b}	75.4^{b}	35.3^{bc}	$\underline{\mathbf{71.6}}^{abc}$	60.6^{bc}	**30.2**	$\underline{\mathbf{54.7}}^{c}$	48.3^{ac}
ClustFuseBorda	$\underline{\mathbf{11.6}}^{abc}$	$\mathbf{80.4}^{b}$	$\mathbf{78.6}^{abc}$	$\mathbf{36.5}^{bc}$	71.2^{abc}	$\underline{\mathbf{62.0}}^{bc}$	29.9	54.3	**48.8**^{ac}

Table 4: Fusing the three best MAP-performing runs in a track. Boldface marks the better performing between ClustFuseF and the standard fusion method F that it incorporates; a statistically significant difference between the two is marked with 'f'. Underline marks the best result in a column. Statistically significant differences between the fusion methods and run1, run2, and run3 are marked with 'a', 'b', and 'c', respectively.

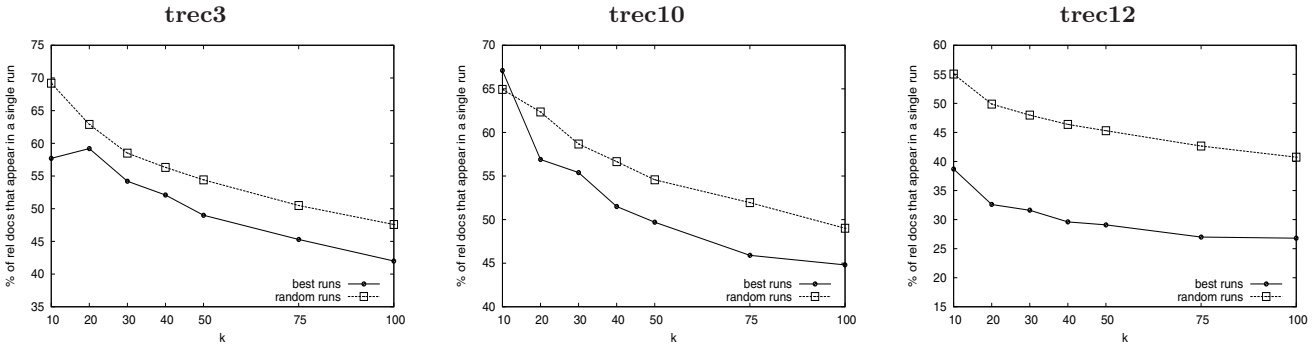

Figure 3: The percentage of relevant documents (of those that appear in the three runs) that appear in only one run as a function of the number of (top) documents considered for a run (k). We use either randomly selected runs, or the best (MAP@k) performing runs; for random runs, numbers represent averages over the 20 random samples of triplets of runs. Note: figures are not to the same scale.

method for ranking clusters based on the presumed percentage of relevant documents that they contain, more effective methods are called for. Evidently, the room for improvement, demonstrated by the numbers for the optimal clusters, is huge. We leave this challenge for future work; specifically, adapting estimates for the amount of within-cluster query-pertaining information that were proposed for the single-list setting to the multiple-lists setting that we address here. Using such estimates in ClustFuse can potentially help to improve its performance.

4.2.6 *Fusing the best-performing runs in a track*

Heretofore, the runs to be fused were randomly selected from all those available for a track. We now turn to study the effectiveness of ClustFuse in fusing the three most MAP(@k = 20)-effective runs in a track; these are denoted, in descending order of MAP performance, run1, run2 and run3. Naturally, this is not a realistic retrieval setting as in practice the quality of retrieval is not known. Nevertheless, this is a challenge for any fusion approach.

We can see in Table 4 that both the ClustFuse methods and the standard fusion approaches that they incorporate are effective in fusing the most effective runs in a track. Indeed, the performance transcends that of the three runs in almost all relevant comparisons (track × evaluation measure). Furthermore, in most relevant comparisons, ClustFuse outperforms the standard fusion method. While the performance differences are statistically significant for only a single relevant comparison, there are quite a few cases for trec3 in which ClustFuse improves over one of the runs in a statistically significant manner, and the standard fusion method does not; the reverse happens for a single relevant comparison in the table.

We also see that the relative improvements posted by ClustFuse over the standard fusion method when fusing the best performing runs are smaller than those posted for fusing random runs. (Compare Table 4 with Table 1). Furthermore, in a small number of cases ClustFuse is outperformed (although not to a statistically significant degree) by the standard fusion method when fusing the best performing runs, while this never happens for the random runs.

A possible explanation of these findings, following a recent report [17], can be made based on Figure 3. Figure 3 shows that the percentage of relevant documents, of those that appear in the three runs, that appear only in a single run is often higher for random runs than for the best runs; thus, the relevant-documents overlap is higher for the best runs than for the random runs. This finding can help explain the fact that the standard fusion methods that depend on this overlap are much more effective for the best runs than for the random runs — e.g., compare their performance with that of run1 in Tables 4 and 1; especially, for trec12 for which the difference in the percentages for random and best runs presented in Figure 3 (k = 20) is very high. Accordingly, our approach can improve over the standard fusion methods more for random runs than for the best runs as it

		trec3			trec10			trec12		
		MAP	p@5	p@10	MAP	p@5	p@10	MAP	p@5	p@10
Random runs	run1	8.4	68.0	64.9	14.0	39.5	34.2	25.2	48.4	42.2
	run2	6.5	57.2	54.9	10.7	34.0	29.7	20.8	42.4	36.7
	run3	4.3	41.7	40.7	6.6	23.2	20.2	13.2	29.7	25.2
	CombMNZ	7.4^{abc}	65.4^{bc}	60.9^{abc}	12.7^{bc}	40.1^{bc}	34.2^{bc}	22.7^{abc}	46.9^{abc}	39.9^{abc}
	GraphFuse	9.4_{f}^{abc}	72.7_{f}^{abc}	70.1_{f}^{abc}	$\mathbf{14.6}^{bc}$	41.8^{bc}	36.6_{f}^{bc}	$\mathbf{25.0}_{f}^{bc}$	$\mathbf{48.0}^{bc}$	$\mathbf{42.5}_{f}^{bc}$
	ClustFuseCombMNZ	$\mathbf{9.6}_{fg}^{abc}$	$\mathbf{75.2}_{fg}^{abc}$	$\mathbf{71.1}_{fg}^{abc}$	14.3_{fg}^{bc}	$\mathbf{42.9}_{fg}$	$\mathbf{37.8}_{fg}^{abc}$	24.5_{f}^{bc}	47.9^{bc}	42.3_{f}^{bc}
Best-performing runs	run1	10.4	74.4	72.2	30.7	63.2	58.8	28.8	51.1	44.8
	run2	9.6	72.8	67.6	27.7	54.4	50.2	28.4	52.5	48.6
	run3	9.5	76.0	71.2	21.6	55.6	46.8^{o}	28.1	51.5	45.2^{o}
	CombMNZ	10.9^{bc}	80.8^{ab}	74.6^{b}	37.2^{bc}	$\mathbf{71.2}^{abc}$	61.0^{bc}	30.3	53.9	$\mathbf{49.2}^{ac}$
	GraphFuse	11.0^{bc}	77.6	76.6^{b}	37.7^{bc}	70.4^{bc}	$\mathbf{62.0}^{bc}$	29.6_{f}	52.5_{f}	48.6_{f}^{ac}
	ClustFuseCombMNZ	$\mathbf{11.4}^{abc}$	$\mathbf{82.8}^{abc}$	$\mathbf{77.2}^{bc}$	$\mathbf{38.3}^{bc}$	70.8^{abc}	61.8^{bc}	**30.5**	**54.1**	49.1^{ac}

Table 5: Comparison with a graph-based fusion method (GraphFuse) [16]. Boldface marks the best result in a column per setup (random runs and best-performing runs). Statistically significant differences with run1, run2, run3, and CombMNZ are marked with 'a', 'b', 'c', and 'f' respectively; 'g' marks statistically significant differences between ClustFuseCombMNZ and GraphFuse.

can help address a low relevant-documents overlap by using inter-document similarities; and, since the standard fusion methods are already quite effective for the best runs.

Finally, we hasten to point out that in *both* settings (random runs and best runs) ClustFuse outperforms the standard fusion method that it incorporates in most relevant comparisons; and, posts more statistically significant improvements over the runs to be fused.

4.2.7 Comparison with graph-based fusion

As mentioned in Section 3, there is some recent work on utilizing inter-document-similarities for fusion using a graph-based approach [16]. Specifically, a graph is constructed from document instances in the lists, and edge weights are based on inter-document similarities and retrieval scores of documents. The stationary distribution of a Markov chain defined over the graph is used to rank documents. The best performing graph-based method reported (BagDupMNZ) [16], which we refer to here as **GraphFuse**, amounts to CombMNZ if inter-document-similarities are not utilized. As our ClustFuseCombMNZ method also amounts to CombMNZ if inter-document-similarities are not utilized (i.e., cluster-based information is not used), we turn to compare the performance of ClustFuseCombMNZ with that of GraphFuse. As is the case for the λ parameter of ClustFuseCombMNZ, we use leave-one-out cross validation for setting the values of the two free parameters that GraphFuse incorporates (namely, graph out degree and edge weight smoothing factor); the search ranges for the parameter values are those originally reported [16]. The performance numbers for fusing randomly selected runs and the best-performing runs are presented in Table 5.[5]

We see in Table 5 that for the random-runs setting ClustFuseCombMNZ outperforms GraphFuse to a statistically significantly degree for 5 out of 9 relevant comparisons (track $\times$ evaluation measure), while the reverse holds for a single relevant comparison. (Refer to the 'g' marks.) Furthermore, for trec10 ClustFuseCombMNZ outperforms CombMNZ in a statistically significant manner for all three evaluation measures, while GraphFuse does so only for p@10. However, for trec12 GraphFuse is somewhat more effective than ClustFuseCombMNZ, although not statistically significantly so.

For the best-runs setup, ClustFuseCombMNZ outperforms GraphFuse in almost all relevant comparisons. Although the performance differences are not statistically significant, ClustFuseCombMNZ posts more statistically significant improvements over the runs, especially for trec3. Furthermore, for trec12, GraphFuse underperforms CombMNZ in a statistically significant manner for all three evaluation measures, while ClustFuseCombMNZ outperforms CombMNZ for two evaluation measures; moreover, in contrast to GraphFuse, ClustFuseCombMNZ is never outperformed by CombMNZ in a statistically significant manner.

All in all, we see that both ClustFuseCombMNZ and GraphFuse that utilize inter-document-similarities, although using different approaches, are highly effective for fusion. Yet, ClustFuseCombMNZ is somewhat more effective — and robust, with respect to relative improvements over CombMNZ — than GraphFuse, which attests to the potential merits of using document clusters to exploit inter-document similarities. In addition, we note that an interesting venue for future work is using graph-based methods in ClustFuse. A case in point, a method similar in spirit to GraphFuse was used to find clusters containing a high percentage of relevant documents in the single retrieved list setting [18]. Adapting the method to the multiple lists setting, and integrating it in ClustFuse, is a challenge we plan to address.

5. CONCLUSIONS

We presented a novel approach to fusing document lists retrieved in response to a query. The approach is based on integrating information used by standard fusion methods (i.e., retrieval scores or ranks) with that induced from clusters of similar documents created across the lists. Specifically, our model, which can incorporate various standard fusion methods, lets similar documents across the lists provide relevance-status support to each other. Empirical evaluation shows that our model outperforms the standard fusion method that it integrates. In addition, we showed that our cluster-based fusion approach outperforms standard fusion of re-ranked lists, wherein list re-ranking is based on

[5]The performance numbers of GraphFuse that we present (for the best-runs setting) are different than those originally reported [16] for the following reasons. While we use leave-one-out cross validation to set free-parameter values (using MAP for the performance-optimization criterion), those were not learned in the original report. Rather, the best potential average performance with respect to free-parameter values (optimized for average p@5) was presented.

clusters of documents from the list. We also showed that clusters created from documents across the lists can contain a much higher percentage of relevant documents than that in clusters created from documents in a list — a finding which further motivates our approach.

Acknowledgments We thank the reviewers for their comments. This paper is based upon work supported in part by the Israel Science Foundation under grant no. 557/09, and by IBM's SUR award. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

6. REFERENCES

[1] C. C. V. ant Garrison W. Cottrell. Fusion via linear combination of scores. *Information Retrieval*, 1(3):151–173, 1999.

[2] J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of SIGIR*, pages 276–284, 2001.

[3] S. M. Beitzel, E. C. Jensen, A. Chowdhury, O. Frieder, D. A. Grossman, and N. Goharian. Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *Proceedings of SAC*, pages 823–827, 2003.

[4] S. M. Beitzel, E. C. Jensen, O. Frieder, A. Chowdhury, and G. Pass. Surrogate scoring for improved metasearch precision. In *Proceedings of SIGIR*, pages 583–584, 2005.

[5] A. Chowdhury, O. Frieder, D. A. Grossman, and M. C. McCabe. Analyses of multiple-evidence combinations for retrieval strategies. In *Proceedings of SIGIR*, pages 394–395, 2001. poster.

[6] N. Craswell, D. Hawking, and P. B. Thistlewaite. Merging results from isolated search engines. In *Proceedings of ADC*, pages 189–200, 1999.

[7] F. Crestani and S. Wu. Testing the cluster hypothesis in distributed information retrieval. *Information Processing and Management*, 42(5):1137–1150, 2006.

[8] W. B. Croft, editor. *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*. Number 7 in The Kluwer International Series on Information Retrieval. Kluwer, 2000.

[9] W. B. Croft. Combining approaches to information retrieval. In Croft [8], chapter 1, pages 1–36.

[10] P. Das-Gupta and J. Katzer. A study of the overlap among document representations. In *Proceedgins of SIGIR*, pages 106–114, 1983.

[11] F. Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of CIKM*, pages 672–679, 2005.

[12] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the Web. In *Proceedings of WWW*, pages 613–622, Hong Kong, 2001.

[13] E. A. Fox and J. A. Shaw. Combination of multiple searches. In *Proceedings of TREC-2*, 1994.

[14] A. Griffiths, H. C. Luckhurst, and P. Willett. Using interdocument similarity information in document retrieval systems. *Journal of the American Society for Information Science (JASIS)*, 37(1):3–11, 1986.

[15] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/Gather on retrieval results. In *Proceedings of SIGIR*, pages 76–84, 1996.

[16] A. K. Kozorovitzky and O. Kurland. From "identical" to "similar": Fusing retrieved lists based on inter-document similarities. In *Proceedings of ICTIR*, pages 212–223, 2009.

[17] A. K. Kozorovitzky and O. Kurland. From "identical" to "similar": Fusing retrieved lists based on inter-document similarities. *Journal of Artificial Intelligence Research*, 41, 2011. (To appear.).

[18] O. Kurland. The opposite of smoothing: A language model approach to ranking query-specific document clusters. In *Proceedings of SIGIR*, pages 171–178, 2008.

[19] O. Kurland. Re-ranking search results using language models of query-specific clusters. *Journal of Information Retrieval*, 12(4):437–460, August 2009.

[20] O. Kurland and L. Lee. Corpus structure, language models, and ad hoc information retrieval. In *Proceedings of SIGIR*, pages 194–201, 2004.

[21] O. Kurland and L. Lee. Respect my authority! HITS without hyperlinks utilizing cluster-based language models. In *Proceedings of SIGIR*, pages 83–90, 2006.

[22] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of SIGIR*, pages 180–188, 1995.

[23] J. H. Lee. Analyses of multiple evidence combination. In *Proceedings of SIGIR*, pages 267–276, 1997.

[24] D. Lillis, F. Toolan, R. W. Collier, and J. Dunnion. Probfuse: a probabilistic approach to data fusion. In *Proceedings of SIGIR*, pages 139–146, 2006.

[25] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Proceedings of SIGIR*, pages 186–193, 2004.

[26] X. Liu and W. B. Croft. Experiments on retrieval of optimal clusters. Technical Report IR-478, Center for Intelligent Information Retrieval (CIIR), University of Massachusetts, 2006.

[27] X. Liu and W. B. Croft. Evaluating text representations for retrieval of the best group of documents. In *Proceedings of ECIR*, pages 454–462, 2008.

[28] L. Meister, O. Kurland, and I. G. Kalmanovich. Re-ranking search results using an additional retrieved list. *Information Retrieval*, 2010.

[29] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In *Proceedings of CIKM*, pages 538–548, 2002.

[30] S. B. Selvadurai. Implementing a metasearch framework with content-directed result merging. Master's thesis, North Carolina State University, 2007.

[31] J. Seo and W. B. Croft. Geometric representations for multiple documents. In *Proceedings of SIGIR*, pages 251–258, 2010.

[32] J. G. Shanahan, J. Bennett, D. A. Evans, D. A. Hull, and J. Montgomery. Clairvoyance Corporation experiments in the TREC 2003. High accuracy retrieval from documents (HARD) track. In *Proceedings of TREC-12*, pages 152–160, 2003.

[33] D. Sheldon, M. Shokouhi, M. Szummer, and N. Craswell. Lambdamerge: merging the results of query reformulations. In *Proceedings of WSDM*, pages 795–804, 2011.

[34] M. Shokouhi. Segmentation of search engine results for effective data-fusion. In *Proceedings of ECIR*, pages 185–197, 2007.

[35] M. Shokouhi, L. Azzopardi, and P. Thomas. Effective query expansion for federated search. In *Proceedings of SIGIR*, pages 427–434, 2009.

[36] X. M. Shou and M. Sanderson. Experiments on data fusion using headline information. In *Proceedgins of SIGIR*, pages 413–414, 2002.

[37] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of CIKM*, pages 623–632, 2007.

[38] I. Soboroff, C. K. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of SIGIR*, pages 66–73, 2001.

[39] T. Tao, X. Wang, Q. Mei, and C. Zhai. Language model information retrieval with document expansion. In *Proceedings of HLT/NAACL*, pages 407–414, 2006.

[40] A. Tombros, R. Villa, and C. van Rijsbergen. The effectiveness of query-specific hierarchic clustering in information retrieval. *Information Processing and Management*, 38(4):559–582, 2002.

[41] T. Tsikrika and M. Lalmas. Merging techniques for performing data fusion on the web. In *Proceedings of CIKM*, pages 127–134, 2001.

[42] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, second edition, 1979.

[43] E. M. Voorhees. Overview of the TREC 2005 robust retrieval task. In *Proceedings of TREC-14*, 2005.

[44] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. The collection fusion problem. In *In Proceedings of TREC-3*, 1994.

[45] P. Willett. Query specific automatic document classification. *International Forum on Information and Documentation*, 10(2):28–32, 1985.

[46] L. Yang, D. Ji, G. Zhou, Y. Nie, and G. Xiao. Document re-ranking using cluster validation and label propagation. In *Proceedings of CIKM*, pages 690–697, 2006.

[47] H. P. Young. An axiomatization of Borda's rule. *Journal of Economic Theory*, 9:43–52, 1974.

[48] O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. In *Proceedings of SIGIR*, pages 46–54, 1998.

[49] C. Zhai and J. D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of SIGIR*, pages 334–342, 2001.