Cost-Effective Combination of Multiple Rankers: Learning When Not To Query

ABSTRACT

Combining multiple rankers has potential for improving the performance over using any of the single rankers. However, querying multiple rankers for every search request can often be too costly due to efficiency or commercial reasons. In this work, we propose a more cost-effective approach that predicts the utility of any additional rankers, prior to querying them. We develop a combined measure that allows us to maximize the gains in retrieval effectiveness of ranker combination subject to a given efficiency constraint. Given the results of a baseline ranker, we predict two quantities that indicate the utility of querying an additional ranker: (1) Overlap — the fraction of its results already retrieved by the baseline ranker and (2) Gain — the increase in effectiveness over the baseline ranker. Our experimental results on both a large web and TREC collections demonstrate the viability of our approach to cost-effective ranker combination. We also show that we can significantly improve the effectiveness and efficiency of ranker combination by augmenting the explicit relevance judgments with automatically generated overlap data. Overall, using easy-to-compute features based on the top retrieved results of a baseline ranker, we attain nearly 10% improvements over a class-prior based method.

1. INTRODUCTION

Applications that combine multiple rankers are ubiquitous on the web today. These applications include meta-search engines (*Dogpile.com, Clusty.com, Metacrawler.com*), search engine comparisons (*Bing-vs-Google.com*) and specialized applications that use general-purpose search engines to augment their own results (*Powerset.com, Facebook.com*). However, querying all potentially available rankers for every incoming user request can be both computationally expensive (if rankers have high latency) or expensive due to commercial licensing restrictions [18]. In this paper we propose a cost-effective model of ranker combination, which allows the application to predict, given an incoming user query and a base ranking, whether querying an additional ranker is likely to bring in new relevant information.

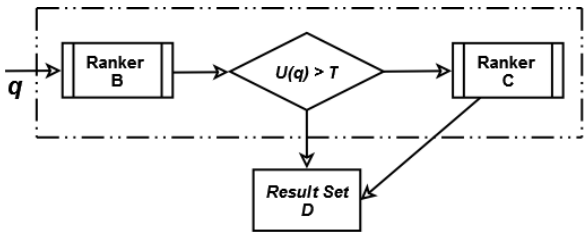

Figure 1: Schematic diagram of a cost-effective ranking combination.

Figure 1 schematically describes the model of cost-effective combination of rankers. For simplicity, we assume that a system (e.g., a meta-search engine) has access to two separate *black-box* rankers: a *base ranker* B and a *candidate ranker* C. First a user query q is received by the system. The query is directly issued to ranker B and its results are processed. Based on these results a decision is made conditioned on the utility $\mathcal{U}(q)$ of querying ranker C being greater than some threshold T. If this condition is satisfied, ranker C is queried and its results are merged with the base ranker results to produce a final ranking $\mathbf{D}$. Otherwise, only the results of the base ranker are returned as $\mathbf{D}$.

We can measure the utility $\mathcal{U}(q)$ of querying the candidate ranker by calculating the inter-ranker overlap. In web search, for instance, the inter-ranker overlap varies significantly across different search queries, depending on query length, intent and the type of results the query retrieves[1]. In experiments we have conducted with 30,000 web search queries, we have found that while the median inter-ranker overlap[2] is below 30%, for more than 20% of queries the overlap is higher than 60%. Such high inter-ranker overlap indicates that there will be little utility in querying an additional candidate ranker for these queries, and we could save bandwidth and time by querying the base ranker alone.

Motivated by these findings, in this paper we develop a statistical model for *cost-effective* ranker combination in a meta-search setting. While most previous work on meta-search and rank-fusion [21, 2, 13] assumes that all the candidate rankers are queried all the time, using our model, we can account for a tradeoff between the cost and the utility

[1]For instance, a query that returns a result from Wikipedia on one web search engine is likely to return the same result on another web search engine.

[2]Using two web search engines, Bing and Yahoo!, as rankers.

of querying a candidate ranker, and only choose to query it if its expected utility exceeds a defined threshold.

Our model is directly optimized for obtaining the best (weighted) balance between the efficiency (number of times a candidate ranker was *not* queried) and the effectiveness (number of additional new/relevant documents retrieved by the candidate ranker) of the system. To perform this optimization we develop a set of easy-to-compute features that rely solely on the query and the output of the base ranker, with no assumptions about the inner workings of their retrieval algorithms or access to their indexes. This *black-box* approach is motivated by common scenario in web search, where rankers are stand-alone search engines that expose their results through a limited-access API.

Thus far, we have focused on using the inter-ranker overlap as a measure for utility of querying a candidate ranker. However, given relevance judgments, it is more accurate to measure this utility by gauging the gain in *relevant* new (rather than *all* new) documents that occurs as a result of querying the candidate. Unfortunately, such relevance judgments have to be done manually, and are expensive to obtain, while overlap data is abundant, and can be generated automatically.

To overcome the problem of scarcity of relevance judgments, we propose a *transfer learning* approach for ranker combination. The transfer learning approach allows us to learn a mapping between the inter-ranker overlap and the relevance gain values. This mapping is then used to create large amounts of surrogate training data, using which we can construct a more effective (in terms of true relevance gains) model of ranker combination.

Evaluation of our methods using both overlap data and relevance judgments affirms the viability of our cost-effective ranker combination model. Experiments on both Web and TREC data show that our method always achieves a better effectiveness-efficiency tradeoff than a prior-based combination baseline. We also empirically demonstrate the merits of our transfer learning approach by augmenting TREC relevance judgments with overlap data.

The remainder of the paper is organized as follows. Section 2 covers the background and the related work. In Section 3 we develop a model for cost-effective ranker combination. In Sections 4 and 5 we report the experimental setup and the experimental results. We conclude the paper in Section 6.

2. BACKGROUND AND RELATED WORK

Combining results from multiple search engines has been studied in diverse application settings such as meta-search [21], rank-fusion [2, 13] and distributed search [8]. A complete survey of this work is beyond the scope of our paper. Instead, we focus on approaches that are pertinent to cost-effective combination of search results in a black box meta-search scenario that is common on the web today. In particular, we present prior work that discusses the relationship between overlap of search results and retrieval effectiveness, and work that focuses on the effectiveness versus efficiency trade-offs in combining results from multiple search engines.

2.1 Overlap versus Effectiveness

In the meta-search setting, multiple rankers can be used to either fuse search results to produce a new ranking [21], or to route the user to the most effective search engine [25].

Rank fusion approaches combine evidence from multiple search engines in order to create a fused rank list [2, 17, 22, 12, 13]. There are two possible reasons for improvements obtained by rank fusion approaches. First, if there is high overlap of relevant documents in the search results compared to non-relevant overlap, the combination of evidence will favor relevant documents more and improve the precision of the fused results ranking [15]. Second, the search results can contain different relevant documents that are combined to improve the recall of the fused results. Bietzel et al [5, 6], show that when the retrieval systems used are highly effective but have systemic differences[3], the improvements due to fusion are largely due to improvements in recall. In the web meta-search scenario, where the rankers are both effective and diverse, rankers with low overlap in search results are more likely to produce better fused results compared to rankers with higher overlap [11].

Different search engines can provide higher quality results for different queries. White et al [25] develop query and result-set dependent techniques to solve the problem of automatically routing users to the search engine that provides the best result for a given query. We consider this problem in the practical meta-search setting which imposes further efficiency restrictions. In particular, we consider the scenario where a meta-search engine has access to the search results page alone and needs to minimize the number of queries issued to rankers. In this setting, suggesting the search results of an additional search engine is useful only if the additional search results contain new relevant documents.

Thus, predicting inter-ranker overlap in search results is useful for effective and efficient combination of these rankers in both rank fusion and routing scenarios. In this work, we target the prediction of both overlap and gain based measures that are proportional to the number of new (relevant) documents that can be obtained by using an additional search engine.

2.2 Effectiveness versus Efficiency

Prior work on federated search has focused on the effectiveness versus efficiency trade-offs involved in selecting a small number of resources that maximize the number of relevant documents returned. Selecting too few resources might yield low recall, whereas selecting too many resources can be inefficient. Si et al [23] utilize a centralized sample of documents created from past queries to estimate the relevance of documents obtained from individual search engines. The estimated relevance of the top-k documents from each search engine is then used in an optimization framework to determine the smallest set of search engines that maximize the expected utility. Cetintas et al [9] propose an extension which considers the cost of downloading search results. Using an optimization framework that addresses the trade-off between effectiveness and efficiency their approach determines the number of documents to download in order to assess the utility of results merging. More recently, Arguello et al [1] use several corpus dependent, query-category based, and click-based features to select few sources (collections) whose results can be combined with effectiveness comparable to a full retrieval on all collections.

In a web search setting, Baeza-Yates et al [4] investigate the effectiveness versus efficiency issues in a two-tiered

[3]Systemic differences are not just differences in retrieval models but also include different tokenization, stemming, stop words etc.

search model with a local server, and a remote server. Usually, every query is routed to a local server first, and is routed to the secondary server only if the results from the local server are deemed inadequate. To avoid the poor response times due to sequential querying, they propose an approach that is able to predict whether the local server's results will be sufficient prior to retrieval.

Our work differs from these federated search approaches both in terms of the additional constraints imposed by the *black box* metasearch scenario and the techniques employed. First, in contrast to the standard federated search setting [1, 4, 3, 9], each query to a ranker incurs a cost regardless of the number of top-k search results obtained, and the meta-search engine has no direct access to full document texts and rankers indexes. In addition, instead of just relying solely on inter-ranker overlap, as is done in some previous work [4], we utilize a relevance gain metric when relevance judgments are available. Finally, we propose to leverage the overlap in order to automatically create surrogate training data in cases when relevance judgments are scarce.

3. LEARNING TO COMBINE RANKERS

In this section, we formally define the ranker combination problem, develop a measure for evaluating ranker combination solutions, and describe an approach for learning when to combine rankers.

3.1 Problem Definition

In this paper we focus on the standard meta-search scenario, where our (meta) search engine has access to *rankers* that retrieve and rank documents from different (but potentially overlapping) collections. In the traditional setting, a meta-search engine *always* accesses *all* the rankers and fuses the returned results [2, 21]. However, accessing all the rankers all the time can be *time-consuming*. For instance, one or more rankers may have high latencies thus, hurting the overall response times. Accessing rankers can also be *expensive* due to licensing restrictions. For example, some search API's already impose time-based or call-based quotas [18]. Instead, we propose a more cost-effective approach that always uses a single base ranker, and only accesses additional ranker(s) when its expected utility is above a certain threshold.

For simplicity, throughout this paper, we will assume that our meta-search engine has access to two rankers: 1) a base ranker, B and 2) a candidate ranker, C. The base ranker, B, is always queried. The candidate ranker, C, is queried only if some criteria is met. Note that while simple, this setting can be easily extended. For instance, we can extend it to a case of multiple rankers by treating C as a set of candidate rankers.

We are interested in a *black-box* scenario, a very common scenario for meta-search engines on the web, in which the search engine has no access to the internal working of its rankers such as their retrieval algorithms or their indexes. Instead, the meta-search engine can only submit a query q to a ranker r, and get in response a ranking $\mathbf{D}_r$, containing K retrieved results. Ranking $\mathbf{D}_r$ typically includes an ordered list of links to retrieved documents (URLs in web search), each accompanied by a brief snippet that provides a short query-biased preview of the document content.

Hence, given a query q, our goal is to query the base ranker, B, and make a binary decision $\mathcal{I}(q,\mathbf{D}_B)$ – based solely on the query and the contents of the base ranking – whether the candidate ranker, C, should be queried as well.

3.2 Measures

The decision of whether or not to query the candidate ranker is based on how much additional *utility* its ranking $\mathbf{D}_C$ will provide, given the current ranking $\mathbf{D}_B$. A simple way to define this utility, when no relevance judgments are available for documents in $\mathbf{D}_B$ and $\mathbf{D}_C$, is by observing the overlap between the rankings. This overlap will be *inversely proportional* to the number of documents in $\mathbf{D}_C$ that are not in $\mathbf{D}_B$. Formally, given a query q and the two rankings $\mathbf{D}_B$, $\mathbf{D}_C$ we define overlap, $\mathcal{O}(q)$, as a fraction of the results in $\mathbf{D}_B$, which appear both in $\mathbf{D}_B$ and in $\mathbf{D}_C$

$$\mathcal{O}(q) = \frac{|\mathbf{D}_B \cap \mathbf{D}_C|}{K}. \tag{1}$$

$\mathcal{O}(q)$ measures how similar the two rankings are. However, it does not convey any information about the overlap in relevant and non-relevant documents retrieved by the two rankers. For instance, rankers B and C can return very different non-relevant results, but the same relevant results. In this case, an actual utility from querying a candidate ranker is low, since no new relevant information has been added. However, the overlap measure in this case may indicate that the difference between the two rankings is high, misleading us to assume a high utility from querying the ranker C.

Accordingly, if we have relevance information about the two rankings, we can utilize it to define the actual relevance gain that can be obtained by querying the candidate ranker. Assuming that $\mathbf{R}_B$ and $\mathbf{R}_C$ are the sets of relevant documents in rankings $\mathbf{D}_B$ and $\mathbf{D}_C$, respectively, we define relevance gain as:

$$\mathcal{G}(q) = \frac{\min\Big(|\mathbf{R}_C \backslash \mathbf{R}_B|, K - |\mathbf{R}_B|\Big)}{K} \tag{2}$$

Note that $\mathcal{G}(q)$ represents the *optimal* bound (in terms of *prec@K*) on the relevance gain from combining the rankings $\mathbf{D}_B$ and $\mathbf{D}_C$. Actual combination of $\mathbf{D}_B$ and $\mathbf{D}_C$ using existing techniques such as CombMNZ [13], Borda-fuse [2] or probFuse [17] may not achieve this bound, but – as some previous work indicates [6] – their performance is likely to correlate with it.

The metrics presented in Equations 1 and 2 suggest two definitions of *utility* of querying an additional ranker.

Def. 1: $\mathcal{U}(q) \triangleq 1 - \mathcal{O}(q)$

Def. 2: $\mathcal{U}(q) \triangleq \mathcal{G}(q)$

We can use *Def. 2* if relevance information is available and *Def. 1* otherwise.

In this paper, we aim to develop a ranker combination model that optimizes the effectiveness of retrieval using the search engine (as embodied by either definition of $\mathcal{U}(q)$), while maintaining a reasonable efficiency. Given a set of queries $\mathbf{Q}$, and an indicator function $\mathcal{I}(q,\mathbf{D}_B)$, which indicates whether, given a query q and a base ranking, we actually query the candidate ranker C, we can formally define the effectiveness of the system as

$$\text{effect}(\mathcal{I}) = \frac{\sum_{q\in\mathbf{Q}} \mathcal{I}(q,\mathbf{D}_B) \times \mathcal{U}(q)}{\sum_{q\in\mathbf{Q}} \mathcal{U}(q)}. \tag{3}$$

We can define the efficiency of the system as the proportion of times we avoid querying the candidate search engine

$$\text{effic}(\mathcal{I}) = 1 - \frac{\sum_{q \in \mathbf{Q}} \mathcal{I}(q, \mathbf{D}_B)}{|\mathbf{Q}|}. \quad (4)$$

Since we are interested in obtaining the best possible trade-off between the competing effectiveness and efficiency measures, we directly optimize our ranker combination model for a measure that combines the two, using a weighted harmonic mean

$$\mathcal{E}_\alpha(\mathcal{I}) = \frac{\text{effect}(\mathcal{I}) \times \text{effic}(\mathcal{I})}{\alpha \times \text{effect}(\mathcal{I}) + (1-\alpha) \times \text{effic}(\mathcal{I})}, \quad (5)$$

where α is a free parameter that determines the relative weight of effectiveness and efficiency.

Note that there is an inherent trade-off in this formulation of ranker combination. Setting $\mathcal{I} = 0$ for all queries in $\mathbf{Q}$, results in effic = 1 and effect = 0, which is equivalent to always using a single base ranker. On the other hand, setting $\mathcal{I} = 1$ for all queries in $\mathbf{Q}$, results in effic = 0 and effect = 1, which is a standard setting in rank fusion and meta-search. We seek to bridge these two edge cases by developing a prediction model that aims to maximize $\mathcal{E}_\alpha(\mathcal{I})$ and sets $\mathcal{I} = 1$ only for queries in $\mathbf{Q}$, for which the expected utility is above a certain threshold T.

3.3 Ranker Combination

Our ranker combination model, is represented by an indicator function $\mathcal{I}^*$: $\{q, \mathbf{D}_B\} \rightarrow \{0, 1\}$. $\mathcal{I}^*$ maps a given query q and a result set $\mathbf{D}_B$ to a binary decision of whether to query the candidate ranker C.

Given a certain utility threshold T, above which the ranker C will be queried, we can define an indicator function $\mathcal{I}_T$ as

$$\mathcal{I}_T(q, \mathbf{D}_B) = \begin{cases} 1 & \text{if } P(\mathcal{U}(q) > T) \geq p' \\ 0 & \text{else,} \end{cases} \quad (6)$$

where $P(\mathcal{U}(q) > T)$ is the probability of $\mathcal{U}(q)$ being above the threshold T, and p' is a free variable. In the binary classification setting, p' is usually set to $\frac{1}{2}$. In the case of ranker combination, however, we are interested in maximizing the performance measure $\mathcal{E}_\alpha(\mathcal{I}_T)$, instead of a classification measure such as precision or recall. Accordingly we set p' so that $\mathcal{E}_\alpha(\mathcal{I}_T)$ is optimized on the training set, and the learned indicator function $\mathcal{I}_T$ directly optimizes the performance of the search engine in terms of $\mathcal{E}_\alpha$.

Clearly, system performance will vary for different choices of T. Thus, in order to obtain the best possible performance on the training data, we choose the indicator function $\mathcal{I}^*$ such that

$$\mathcal{I}^*(q, \mathbf{D}_B) = \underset{\mathcal{I}_T(q, \mathbf{D}_B)}{\text{argmax}}\ \mathcal{E}_\alpha(\mathcal{I}_T). \quad (7)$$

Accordingly, to optimize the cost-effective ranker combination model, we estimate the probability $P(\mathcal{U}(q) > T)$ in Equation 6 such that the measure $\mathcal{E}_\alpha$[4] is optimized.

3.4 Features

To learn the probability $P(\mathcal{U}(q) > T)$ (Eq. 6) we use a standard logistic regression model

$$P(\mathcal{U}(q) > T) = \frac{1}{1 + e^{\Lambda F_q}},$$

[4]In the remainder of this paper, we will abbreviate $\mathcal{E}_\alpha(\mathcal{I}^*)$ as $\mathcal{E}_\alpha$.

Table 1: Features used for utility prediction. Features marked by † are computed only for Web queries.

Source	Feature Name	Description	Aggregates
q	`qLenTerms`	# terms in the query	
	`qLenChars`	# characters in the query	
	`qAggTermLen`	Term lengths in the query	`Max,Mean`
	`qIsCap`	Does the query contain capitalized terms?	
	`qNStops`	# stopwords in the query	
	`qIsQuestion`	Does the query start with a wh-word?	
	`qWikiNgram`	Fraction of query n-grams appearing as wiki titles	
$\mathbf{D}_B$	`uDepth`	URL depth in $\mathbf{D}_B$	`Max,Mean,Std`
	`uLenChars`	# chars in URL's	`Max,Mean,Std`
	`uIsWiki` †	Wikipedia URL's	`Max,Mean,Std`
	`uuOvlp`	Inter-snippet overlap in	`Max,Mean,Std`
	`uEntropy`	Entropy of snippets	`Max,Mean,Std`
	`uqOvlp`	Query-snippet overlap	`Max,Mean,Std`
	`uqCover`	Fraction of query terms covered by the snippets	
	`uqNgramCover`	Fraction of query n-grams covered by the snippets	
	`uqFullCover`	Does exact query match appear in the snippets?	

where $F_q = f_{q1}, \ldots, f_{qn}$ is a feature vector representing query q and $\Lambda = \lambda_1, \ldots, \lambda_n$ is an associated weight vector, which is optimized to reduce the classification error on a training set.

Table 1 shows the features in F_q. We divide the features into two groups based on their source - the query itself, and the retrieved set $\mathbf{D}_B$. We can either use only the features based on the query itself, and perform a utility prediction without querying the base ranker, or allow for querying the base ranker and use features from both sources.

There are two main motivations for each feature we are computing: (a) it has to be correlated with the expected utility of querying a candidate ranker C for query q, and (b) it has to be highly efficient to compute. For some features that are computed over the retrieved list $\mathbf{D}_B$, we compute several aggregates. Each of these aggregates is used as a separate feature in F_q.

In contrast to previous work we use neither the traditional *pre-retrieval* query performance predictors such as IDF or PMI [14, 4] nor *post-retrieval* performance predictors such as Query Clarity [10] or Weighted Information Gain [26]. This is due to the fact that we restrict our attention to the black-box setting assuming that we have no access to rankers' retrieval algorithms or indexes.

Instead, we use only the information we can glean from the query itself, such as its length and its grammatical structure (e.g., features `qLenTerms,qLenChars,qIsCap,qIsQuestion`), which were shown to correlate with query performance [7], the structure of URLs (features `uDepth,uLenChars`) and the contents of the snippets in the retrieved list. To estimate *inter-ranker* overlap, we use the *intra-ranker* overlap (overlap between the retrieved snippets - `uuOvlp,uuEntropy`) and *query-ranker* overlap (`uqNgramCover,uqFullCover`) as approximations.

3.5 Transfer Learning

Thus far we have only considered combining rankers using either one of the two definitions of utility presented in Section 3.2. However, in practice, the number of judged relevant documents for learning a ranker combination based on the relevance gain $\mathcal{G}(q)$ is limited, while the data for learning a

ranker combination based on the overlap $\mathcal{O}(q)$ is abundant: it is obtained automatically whenever both rankers B and C are queried. Accordingly, we would like to leverage the overlap data to improve the relevance gains.

However, low overlap between retrieved sets does not always directly correspond to high relevance gains from their combination [16]. To illustrate this point, Figure 2 shows the distribution of relevance gains against different levels of overlap between two different retrieval system for 150 Gov2 TREC topics. As expected, when overlap is high the possible gains are usually low. However, when the overlap is low there is a much higher variance in possible gains.

Therefore, directly using the overlap-based definition of utility (*Def. 1*) as a surrogate for gain-based definition is not practical. Instead, to estimate the possible gain, we attempt to directly predict the possible gains, using the overlap and the same set of features, F, as used for ranker combination. We formally define the transfer learning as follows.

Let $\mathbf{G}$ denote the set of training queries that have both relevance gains and overlap information and let $\mathbf{O}$ denote the set of training queries that only have overlap information but no relevance gains. Then, the transfer learning task is to learn a mapping function $M : \{F_q, \mathcal{O}(q)\} \rightarrow \mathcal{G}(q)$ using $\mathbf{G}$ as the training data. We create an augmented training set $\mathbf{G}'$, by applying the mapping function to $\mathbf{O}$ i.e., $\mathbf{G}' = \mathbf{G} \bigcup_{q \in \mathbf{O}} M(F_q, \mathcal{O}(q))$. Finally, we learn the ranker combination models using this augmented training set[5].

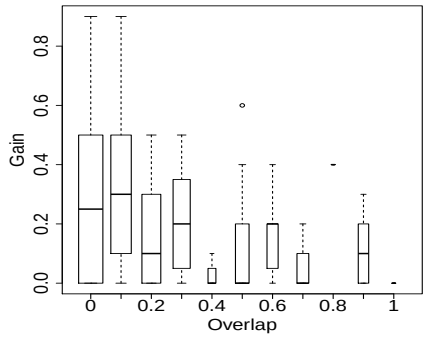

Figure 2: Overlap versus Gains: Distribution of gain values for different levels of overlap in Gov2 queries.

4. EXPERIMENTAL SETUP

4.1 Datasets and Rankers

We evaluate our approach using 3-fold cross validation on three different datasets 1) *Web*, 2) *Million* and 3) *Gov2*. *Web* is a set of 30,000 queries that we sample uniformly at random from a large search log of a commercial web search engine[6]. *Million* is a set of 10,000 title portions of topics created for the TREC 2008 Million Query Track and *Gov2* is set of 150 title portions of topics created for the TREC Terabyte track. Both Million and Gov2 queries were used over the same document collection (a crawl of the .gov domain). In keeping with the *black-box* scenario for meta-search, for each query in all query collections we retrieve the top 10 search results and only extract the corresponding URL, and the snippet information found on the results page.

For the Web queries, we use Bing[7] as the baseline ranker, and Yahoo! search engine[8] as the candidate ranker. For Million and Gov2 queries, we use Indri [24], and use Query Likelihood (QL) [19] as a baseline ranker, and Okapi BM25 [20] as a candidate ranker, and use the default snippet generation available in Indri. For each model, we train parameters to optimize the performance on the Gov2 queries and apply it to both Gov2 and Million queries. In all the experiments, we report the results obtained using a 3-fold cross-validation, and the statistical significance is reported using Fisher's randomization test with 10,000 permutations.

4.2 Tasks

We conduct three sets of experiments to validate our approach for predicting the utility of ranker combination: 1) Overlap ($\mathcal{O}$) based ranker combination, 2) Gain ($\mathcal{G}$) based ranker combination, and 3) Transfer Learning for gain based ranker combination.

1) **Overlap-Based Ranker Combination** – The objective for this task is to select queries for which ranker C is going to be queried, such that the selection maximizes $\mathcal{E}_\alpha$ (Equation 5) when utility is defined as the fraction of new, but not necessarily relevant, results obtained through combination i.e., *Def. 1* of $\mathcal{U}(q)$. To demonstrate the benefits of our prediction approach for this task, we conduct experiments on all three datasets.

2) **Gain-Based Ranker Combination** – We conduct direct gain based ranker combination experiments on the 150 Gov2 queries for which relevance judgments are available. Our goal is to demonstrate the benefits of the prediction approach for the task of selecting queries that maximize $\mathcal{E}_\alpha$ when utility is defined as the fraction or new relevant documents obtained through combination i.e., *Def. 2* of $\mathcal{U}(q)$.

3) **Transfer Learning** – To illustrate the impact of augmenting gain-based ranker combination with easy to generate overlap-based data, we conduct a transfer learning experiment. The goal of transfer is to learn a mapping (M) from overlap to gain on the original Gov2 training folds ($\mathbf{G}$). Using this mapping, described in Section 3.5, we map the overlap values, $\mathcal{O}(q)$, for Million queries ($\mathbf{O}$) to predicted gain values, $M(q)$, and use them to generate additional training instances ($\mathbf{G}'$) for each training fold. The test folds remain unaltered.

4.3 Baseline: PriorRC

Our model of ranker combination is based on a binary decision of whether the utility of querying a candidate ranker is greater than a threshold T. Accordingly, we use a class-prior based method, **PriorRC**, as a competitive baseline. For a given threshold T, **PriorRC** uses the training data to determine the fraction of positive instances

$$fp = \frac{|\{q \in \mathbf{Q} : \mathcal{U}(q) > T\}|}{|\mathbf{Q}|}.$$

[5]We note that the ranker combination model still trains over the same set of features as before and does not use overlap as a feature, since it is not available during testing.

[6]Available as a part of Microsoft 2006 RFP dataset.

[7]http://www.bing.com

[8]http://search.yahoo.com

To assign labels to test set, **T**, **PriorRC** samples labels from a binomial distribution $Bin(|\mathbf{T}|, fp)$. We report average evaluation measures obtained over 10 different random assignment of labels to the test fold.

5. EVALUATION

5.1 Overlap-Based Ranker Combination

Table 2(a) compares results for $\mathcal{E}_{\alpha=0.5}$, i.e., the best combined measure achieved when effectiveness and efficiency are equally weighted. **LearnRC** achieves higher combined measure values (shown in $\mathcal{E}_{\alpha=0.5}$ column) compared to **PriorRC**, on all three collections. The *Effect* values show the amount of additional gains that can be obtained through ranker combination, and *Effic* values show the savings in terms of avoided candidate searches. Relative to **PriorRC**, **LearnRC** achieves nearly 5%, 6%, and 4% improvements on the Web, Million and Gov2 queries respectively. For all three collections, prediction provides substantial savings in accesses to the candidate ranker, while yielding comparable or more gains than **PriorRC**. For example, on Million, **LearnRC** queries the candidate ranker 8% less times than **PriorRC** while still yielding increase of 3% in effectiveness. Overall, **LearnRC** achieves better performance on the larger datasets Web and Million, compared to the smaller Gov2 dataset, and the improvements for both of them are statistically significant,

Next, we inspect the classification accuracies obtained by the ranker combination. Table 2(b) shows the classification accuracy measures at the thresholds where the best $\mathcal{E}_{\alpha=0.5}$ is obtained. The classification accuracies are modest, especially for the Million, and Gov2 queries. There are two main factors that contribute to the observed classification accuracy levels. Classification accuracies are usually higher when larger amounts of training data are available. In our case, Web has more than 30,000 queries, whereas Gov2 has only 150 queries in all, which in part can explain the different levels of prediction accuracies. Fortunately, overlap based training data does not require any manual judgments and therefore can be automatically generated to further improve accuracy for all the collections. In addition, classification accuracy is affected by the ratio of positive to negative instances, fp. In our case, fp depends on the threshold selected for the task. For the Web queries, at the chosen threshold the positive class ratio is more than 50%, whereas for the Million and Gov2 queries, the ratio is much lower. This shows that classification accuracy is dependent on the collection of queries, and the distribution of overlap amongst the queries. We note, however, that classification precision/recall trade-off does not directly correspond to ranker combination effectiveness/efficiency trade-off measured by $\mathcal{E}_\alpha$. That is why, for instance, F1 attained by **PriorRC** for Gov2 is higher than F1 attained by **LearnRC** while its $\mathcal{E}_\alpha$ is lower.

Finally, to understand the utility of the query-based features (shown in Table 1), we conduct ranker combination using this subset of features alone. The results are shown in Table 3. Using query-based features that do not require initial access to the base ranker still provides small improvements (2%) over **PriorRC**. It is however, not as effective as using the full set of features. This suggests that result set based features are vital for this task. However, in scenarios where base ranker has a slow response time, and parallel access to both rankers is available, predicting the utility of querying a candidate ranker by computing query-based features alone can be still useful.

Table 3: Performance of different feature groups: 1) None - PriorRC with no features. 2) Query - LearnRC with only query-based features, and 3) All - LearnRC with both query-based and results-based features

Method	Features	$\mathcal{E}_{\alpha=0.5}$	Effect.	Effic.
PriorRC	–	0.4973	0.5382	0.4622
LearnRC	Query	0.5082	0.5478	0.4739
	All	0.5206	0.5244	0.5168

5.2 Gain-Based Ranker Combination

Table 4 shows the performance of direct gain prediction. **LearnRC** achieves nearly 50% of the possible gains, despite querying fewer than 42%. Compared to **PriorRC**, **LearnRC** reduces the gain by 2%, while yielding more than 15% relative improvements in efficiency.

Table 4: Results of gain-based ranker combination for 150 Gov2 queries.

Method	$\mathcal{E}_{\alpha=0.5}$	Effect.	Effic.
PriorRC	0.5032	0.5152	0.5020
LearnRC	0.5331	0.4952	0.5772
	F1	Prec.	Rec.
PriorRC	0.5143	0.5134	0.5174
LearnRC	0.5085	0.5455	0.4762

In addition to the overall improved ranker combination, our $\mathcal{E}_\alpha$-based approach for gain prediction has two beneficial properties compared to a standard classification approach.

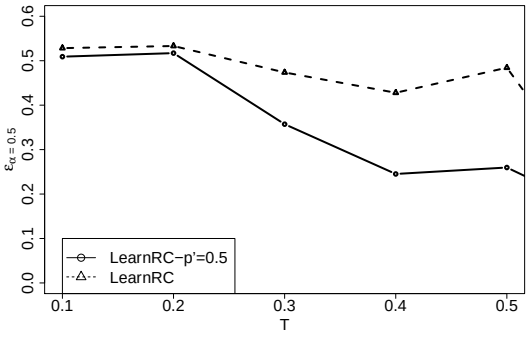

Figure 3: Comparison of LearnRC's performance with fixed decision threshold (LearnRC-p'=0.5), and variable decision thresholds (LearnRC).

First, as shown in Equation 6, **LearnRC** learns different conditions for querying a candidate ranker for different thresholds T. As a result, **LearnRC** achieves stable overall performance when varying T, as shown in Figure 3.

Table 2: Results of overlap-based ranker combination. Statistically significant differences in $\mathcal{E}_\alpha$ between LearnRC and PriorRC are denoted with †.

(a) **Results for $\mathcal{E}_\alpha$**

Collection	Method	$\mathcal{E}_{\alpha=0.5}$	Effect.	Effic.
Web	**PriorRC**	0.4973	0.5382	0.4622
	LearnRC	0.5206† *(+4.7%)*	0.5244	0.5168
Million	**PriorRC**	0.4999	0.5048	0.4952
	LearnRC	0.5283† *(+5.7%)*	0.5201	0.5368
Gov2	**PriorRC**	0.4933	0.4806	0.5141
	LearnRC	0.5150 *(+4.4%)*	0.5272	0.5034

(b) **Classification Results**

Coll.	Method	F1	Rec.	Prec.	fp
Web	**PriorRC**	0.5392	0.5392	0.5392	54%
	LearnRC	0.6400	0.6400	0.6400	53%
Million	**PriorRC**	0.5048	0.5046	0.5051	50%
	LearnRC	0.5635	0.6806	0.4800	34%
Gov2	**PriorRC**	0.4802	0.4770	0.4852	50%
	LearnRC	0.4504	0.6757	0.3378	25%

LearnRC is more robust to changes in classification performance with respect to T, when compared to a fixed binary classifier. This is due to the fact that **LearnRC** dynamically adapts to reduce p' as T increases, while the classification-based approach is static and always sets $p' = 0.5$.

Second, as illustrated in Figure 4, **LearnRC** adjusts its learning for different settings of α that controls the effectiveness/efficiency tradeoff in $\mathcal{E}_\alpha$ (Equation 5). When $\alpha \to 0$, **LearnRC** increases the number of queries that access the candidate ranker, leading to improved effectiveness, while if $\alpha \to 1$, **LearnRC** decreases the number of queries resulting in improved efficiency. This shows that **LearnRC** can be tuned for different effectiveness and efficiency trade-offs by appropriately setting α.

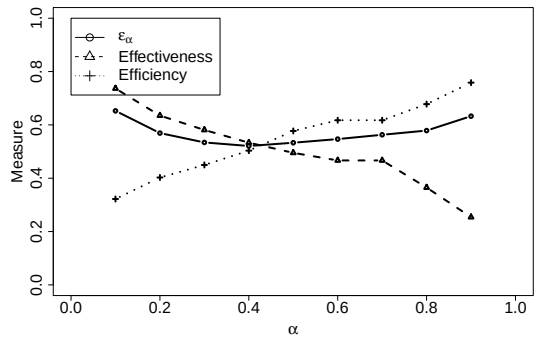

Figure 4: LearnRC performance for varying α.

An inspection of the classification measures shown in Table 4 reveals that **LearnRC**'s classification accuracy is lower than **PriorRC**'s classification accuracy. Despite the lower accuracy **LearnRC** achieves higher $\mathcal{E}_\alpha$ performance, as it is targeted to optimize this measure. This suggests that improving classification accuracy, e.g. by adding more training data over the 150 available Gov2 queries, could further improve the performance of rank combination.

5.3 Transfer Learning

As noted earlier, the small amount of training data available on Gov2 can limits the performance of ranker combination. To improve combination and classification accuracies, we conduct transfer learning experiments with automatically generated training data. Table 5 compares the performance of **PriorRC**, gain-based ranker combination, and the best transfer learning performance obtained when using the augmented training data obtained from Million queries (see Section 4.2 for more details on the transfer learning setup). Using 700 additional surrogate instances, we achieve clear improvements over both **PriorRC** and **LearnRC**. Transfer learning improves $\mathcal{E}_\alpha$ by nearly 10% over **PriorRC** and by more than 4% over **LearnRC**. Note that using transfer learning we are able to attain statistical significant improvements over the **PriorRC** baselines, despite a small amount of gain-based training data. The classification accuracies in the same table show that transfer also improves the classification accuracy of our approach by supplying a large amount of reliable surrogate data for training.

Table 5: Results of transfer learning based ranker combination. Statistically significant differences in $\mathcal{E}_\alpha$ with PriorRC are denoted †.

Method	$\mathcal{E}_{\alpha=0.5}$	% imp	Effect.	Effic.
PriorRC	0.5032		0.5152	0.5020
LearnRC	0.5331	*(+5.9%)*	0.4952	0.5772
Transfer$_{700}$	0.5516†	*(+9.6%)*	0.5365	0.5772
	F1	% imp	Prec.	Rec.
PriorRC	0.5143		0.5134	0.5174
LearnRC	0.5085	*(-1.1%)*	0.5455	0.4762
Transfer$_{700}$	0.5254	*(+2.1%)*	0.5636	0.4921

Furthermore, the improvements obtained through transfer learning are consistent. Figure 5 shows that except for small amounts of transfer instances (less than 300), adding overlap instances improves over the using the Gov2 training data alone. Also, as more transfer instances are added $\mathcal{E}_\alpha$ peaks around 700 instances, beyond which the improvements drop. The surrogate gains in the transferred instances can be viewed as noisy labels. We hypothesize that as the proportion of the noisy transfer instances increase, the relative influence of the original Gov2 instances decreases, thereby causing the observed dip in performance. Nonetheless, our results show that this simple transfer approach can be used to provide improvements for the task of gain-based ranker combination.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we discuss a problem of ranker combination in a black-box setting, where the meta-search engine has no access to the internal information about its rankers. To measure the performance of ranker combination, we propose a measure $\mathcal{E}_\alpha$ that balances the trade-off between the effectiveness and the efficiency aspects. We develop a statistical model for cost-effective ranker combination that directly optimizes the proposed measure.

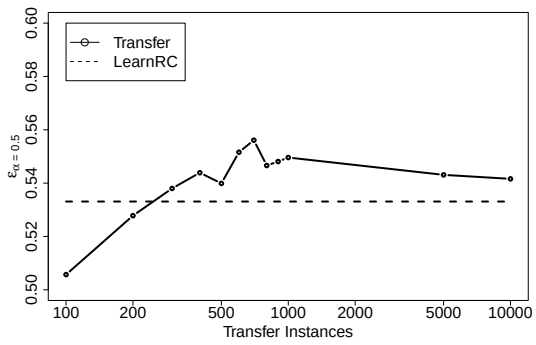

Figure 5: Transfer based improvements: Effect of adding increasing amounts of transfer instances on $\mathcal{E}$. The horizontal line corresponds to the performance of LearnRC with no transfer instances from Million.

Empirical results on three query collections demonstrate the utility of our approach. Compared to the class-prior based method, our approach provides notable improvements using both overlap-based and gain-based utility measures. The improvements are significant for collections with large amounts of training data. We also develop a transfer learning method that automatically generates surrogate relevance gain instances using overlap data. We show that this transfer learning approach attains a 10% improvement over the baseline method even when the amount of available relevance data is extremely limited.

In this work, we operated within the constraints imposed by the black box scenario, with no access to large external collections or search history. In this scenario, we must query the base ranker in order to obtain the reliable features for learning the ranker combination. A natural extension to this work would be to partly relax the black-box constraints and to allow the meta-search engine to use external data when available, in order to avoid always querying the base ranker.

7. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by ARRA NSF IIS-9014442. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

8. REFERENCES

[1] J. Arguello, J. Callan, and F. Diaz. Classification-based resource selection. In *Proc. of CIKM*, pages 1277–1286, 2009.

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

[3] R. Baeza-Yates, A. Gionis, F. Junqueira, V. Plachouras, and L. Telloli. On the feasibility of multi-site web search engines. In *Proc. of CIKM*, pages 425–434, 2009.

[4] R. Baeza-Yates, V. Murdock, and C. Hauff. Efficiency trade-offs in two-tier web search systems. In *Proc. of SIGIR*, pages 163–170, 2009.

[5] S. Beitzel, O. Frieder, E. Jensen, D. Grossman, A. Chowdhury, and N. Goharian. Disproving the fusion hypothesis: An analysis of data fusion via effective information retrieval strategies. In *Proc. of SAC*, pages 827–832. ACM, 2003.

[6] S. Beitzel, E. Jensen, A. Chowdhury, D. Grossman, N. Goharian, and O. Frieder. Recent Results on Fusion of Effective Retrieval Strategies in the Same Information Retrieval System. In *Lecture notes in computer science Vol. 2924*, pages 101–111. Springer, 2004.

[7] M. Bendersky and W. B. Croft. Analysis of long queries in a large scale search log. In *Proc. of WSCD*, pages 8–14, 2009.

[8] J. Callan. Distributed information retrieval. In W. B. Croft, editor, *Advances in information retrieval*, pages 127–150. Springer, 2000.

[9] S. Cetintas and L. Si. Exploration of the tradeoff between effectiveness and efficiency for results merging in federated search. In *Proc. of SIGIR*, pages 707–708, 2007.

[10] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proc. of SIGIR*, pages 299–306, 2002.

[11] Dogpile.com. Different engines, different results: Web searchers not always finding what theyre looking for online. `http://www.infospaceinc.com/files/Overlap-DifferentEnginesDifferentResults.pdf`, April 2007.

[12] M. Efron. Generative model-based metasearch for data fusion in information retrieval. In *Proc. of JCDL*, pages 153–162, 2009.

[13] E. Fox and J. Shaw. Combination of multiple searches. In *Proc. of TREC-3*, pages 243–252, 1994.

[14] B. He and I. Ounis. Inferring query performance using pre-retrieval predictors. In *Proc. of SPIRE*, pages 43–54, 2004.

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

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

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

[18] J. Musser. 12 ways to limit an API. Programmable Web Blog, `http://blog.programmableweb.com/2007/04/02/12-ways-to-limit-an-api/`.

[19] J. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proc. of SIGIR*, pages 275–281, 1998.

[20] S. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proc. of TREC-3*, pages 109–126, 1994.

[21] E. Selberg and O. Etzioni. Multi-service search and comparison using the MetaCrawler. In *Proc. of WWW*, pages 169–173, 1995.

[22] M. Shokouhi. Segmentation of search engine results for effective data-fusion. In *Proc. of ECIR*, page 185195, 2007.

[23] L. Si and J. Callan. Modeling search engine effectiveness for federated search. In *In Proc. of SIGIR*, pages 83–90, 2005.

[24] T. Strohman, D. Metzler, H. Turtle, and W. B. Croft. Indri: A language model-based search engine for complex queries. In *Proc. of International Conference on Intelligence Analysis*, 2004.

[25] R. W. White, M. Richardson, M. Bilenko, and A. P. Heath. Enhancing web search by promoting multiple search engine use. In *Proc. of SIGIR*, pages 43–50, 2008.

[26] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proc. of SIGIR*, pages 543–550, 2007.