The Opposite of Smoothing: A Language Model Approach to Ranking Query-Specific Document Clusters

Oren Kurland
Faculty of Industrial Engineering and Management
Technion — Israel Institute of Technology
Haifa 32000, Israel
kurland@ie.technion.ac.il

ABSTRACT

Exploiting information induced from (*query-specific*) clustering of top-retrieved documents has long been proposed as means for improving precision at the very top ranks of the returned results. We present a novel language model approach to ranking query-specific clusters by the presumed percentage of relevant documents that they contain. While most previous cluster ranking approaches focus on the cluster as a whole, our model also exploits information induced from documents associated with the cluster. Our model substantially outperforms previous approaches for identifying clusters containing a high relevant-document percentage. Furthermore, using the model to produce document ranking yields precision-at-top-ranks performance that is consistently better than that of the initial ranking upon which clustering is performed; the performance also favorably compares with that of a state-of-the-art pseudo-feedback retrieval method.

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

General Terms: Algorithms, Experimentation

Keywords: ad hoc retrieval, cluster-ranking, query-specific clusters, language models, cluster-based language models

1. INTRODUCTION

Clustering the results returned by a search engine in response to a query has long been proposed as means for improving retrieval effectiveness [35, 43, 14, 27, 29, 21]. Much of the motivation for employing clustering of search results (a.k.a. *query-specific clustering*) comes from van Rijsbergen's *cluster hypothesis* [40], which states that "closely associated documents tend to be relevant to the same requests". Indeed, researchers showed that applying various clustering techniques to the documents most highly ranked by some initial search produces some clusters that contain a very high percentage of relevant documents [14, 38, 17, 30]. Moreover, positioning these clusters' constituent documents at the very top ranks of the returned results yields precision-at-top-ranks performance that is substantially better than that of (state-of-the-art) document-based retrieval approaches [14, 38, 17]. However, automatically identifying these clusters is a very hard task [43, 29, 21, 31, 30].

We present a novel language-model-based [34, 8] approach to ranking query-specific clusters by the presumed percentage of relevant documents that they contain. The key insight that guides the derivation of our cluster-ranking model is that documents that are strongly associated with a cluster can serve as proxies for ranking it. Case in point, since documents can be considered as more "focused" units than clusters, they can serve, for example, as mediators for estimating the cluster-query similarity. Thus, while most previous approaches to ranking various types of clusters [15, 7, 41, 43, 19, 29, 31] focus on the cluster as a "whole" unit, our model integrates cluster-induced information with that induced from its proxy (associated) documents. Hence, we conceptually take the opposite approach to that of cluster-based *smoothing* of document language models that has recently been used for document ranking [2, 19, 29, 42].

Our model integrates two types of information induced from clusters and their proxy documents. The first is the estimated similarity to the query. The second is the *centrality* of an element (document or cluster) with respect to its reference set (documents in the initially-retrieved list or clusters of these documents); centrality is defined in terms of textual similarity to (many) other (central) elements in the reference set [20]. Using either (or both) type(s) of information just described — induced from a cluster as a whole and/or from its proxy documents — yields several novel cluster-ranking criteria that are integrated in our model. We study the relative contribution of each of the criteria to the overall effectiveness of our approach.

Empirical evaluation shows that our cluster-ranking model consistently outperforms previously proposed methods in identifying clusters that contain a high percentage of relevant documents. Furthermore, positioning the constituent documents of the cluster most highly ranked by our model at the top of the returned document list yields precision-at-top-ranks performance that is substantially better than that of the initial document ranking upon which clustering was performed. The resultant performance also favorably competes with that of a state-of-the-art pseudo-feedback-based document retrieval method.

2. RANKING FRAMEWORK

Notation and conventions. Throughout this section we assume that the following have been fixed: a query q, a corpus of documents $\mathcal{D}$, and an initial list of N documents $\mathcal{D}^N_{\text{init}} \subset \mathcal{D}$ (henceforth $\mathcal{D}_{\text{init}}$) that are the highest ranked by some search performed in response to q. We assume that $\mathcal{D}_{\text{init}}$ is clustered into a set of (*query-specific*) document clusters $Cl(\mathcal{D}_{\text{init}}) = \{c_1, \cdots, c_M\}$ by some clustering algorithm[1]. Our goal is to rank the clusters in $Cl(\mathcal{D}_{\text{init}})$ by the presumed percentage of relevant documents that they contain. In what follows we use the term "cluster" to refer either to the set of documents it is composed of, or to a (language) model induced from it. We use $p_y(x)$ to denote the language-model-based similarity between y (a document or a cluster) and x (a query or a cluster); we describe our language-model induction method in Section 4.1.

2.1 Cluster Ranking

Similarly to the language model approach to ranking documents [34, 8], and in deference to the recent growing interest in automatically labeling document clusters and topic models [11, 39, 32], we state the problem of ranking clusters as follows: estimate the probability $p(c|q)$ that cluster c can be labeled (i.e., its content can be described) by the terms in q. Our hypothesis is that the higher this probability is, the higher is the percentage of documents pertaining to q that c contains.

Since q is fixed, we use the rank equivalence

$$p(c|q) \overset{rank}{=} p(q|c) \cdot p(c)$$

to rank the clusters in $Cl(\mathcal{D}_{\text{init}})$. Thus, c is ranked by combining the probability $p(q|c)$ that q is "generated"[2] as a label for c with c's prior probability ($p(c)$) of "generating" *any* label. Indeed, most prior work on ranking various types of clusters [15, 7, 43, 41, 19, 29] implicitly uses uniform distribution for $p(c)$, and estimates $p(q|c)$ (in spirit) by comparing a representation of c as a whole unit with that of q.

Here, we suggest to incorporate a document-mediated approach to estimating the probability $p(q|c)$ of generating the label q for cluster c. Since documents can be considered as more coherent units than clusters, they might help to generate more "informative/focused" labels than those generated by using representations of clusters as whole units. Such an approach is conceptually the opposite of *smoothing* a document representation (e.g., language model) with that of a cluster [2, 19, 29, 42]. In what follows we use $p(q|d)$ to denote the probability that q is generated as a label describing document d's content (cf. the language modeling approach to ranking documents [34, 8]). Also, we assume that $p(d)$ — the prior probability that document d generates any label — is a probability distribution over the documents in the corpus $\mathcal{D}$.

We let all (and only) documents in the corpus $\mathcal{D}$ to serve as proxies for label generation for any cluster in $Cl(\mathcal{D}_{\text{init}})$. Consequently, we assume that $p(d|c)$, the probability that d is chosen as a proxy for c for label generation, is a probability distribution defined over the documents in the corpus $\mathcal{D}$.

Then, we can write using some probability algebra

$$p(c|q) \overset{rank}{=} p(c) \sum_{d_i \in \mathcal{D}} p(q|c, d_i) p(d_i|c). \tag{1}$$

We use $\lambda p(q|c) + (1-\lambda) p(q|d_i)$ (where λ is a free parameter) as an estimate for $p(q|c, d_i)$ [37, 19] in Equation 1, and by applying probability algebra we get the following scoring principle[3] for clusters

$$\lambda p(c) p(q|c) + (1-\lambda) \sum_{d_i \in \mathcal{D}} p(q|d_i) p(c|d_i) p(d_i). \tag{2}$$

Equation 2 scores c by a mixture of (i) the probability that q is directly generated from c combined with c's prior probability of generating any label, and (ii) the (average) probability that q is generated by documents that are both "strongly associated" with c (as measured by $p(c|d_i)$) and that have a high prior probability $p(d_i)$ of generating labels.

We next derive specific ranking algorithms from Equation 2 by making some assumptions and estimation choices.

2.2 Algorithms

We first make the assumption, which underlies (in spirit) most pseudo-feedback-based retrieval models [4, 44, 26], that the probability of generating q directly from d_i ($p(q|d_i)$) is quite small for documents d_i that are not in the initially retrieved list $\mathcal{D}_{\text{init}}$; hence, these documents have relatively little effect on the summation in Equation 2. Furthermore, if the clusters in $Cl(\mathcal{D}_{\text{init}})$ are produced by a "reasonable" clustering algorithm, then $p(c|d_i)$ — the cluster-document "association strength" — might be assumed to be significantly higher for documents from $\mathcal{D}_{\text{init}}$ that are in c than for documents from $\mathcal{D}_{\text{init}}$ that are not in c. Consequently, we truncate the summation in Equation 2 by allowing only c's constituent documents to serve as it proxies for generating q. Such truncation does not only alleviate the computational cost of estimating Equation 2, but can also yield improved effectiveness as we show in Section 4.3. In addition, we follow common practice in the language model framework [8], specifically, in work on utilizing cluster-based language models for document retrieval [29, 19], and use language-model estimates for conditional probabilities to produce our primary ranking principle:

$$Score(c) \overset{def}{=} \lambda p(c) p_c(q) + (1-\lambda) \sum_{d_i \in c} p_{d_i}(q) p_{d_i}(c) p(d_i). \tag{3}$$

Note that using $p_d(c)$ for $p(c|d)$ means that we use the probability of generating the "label" c (i.e., *some* term-based representation of c) from document d as a surrogate for the document-cluster association strength.

The remaining task is to estimate the document and cluster priors — $p(d)$ and $p(c)$, respectively.

[1]Clustering the documents most highly ranked by a search performed in response to a query is often termed *query-specific clustering* [43]. We do not assume, however, that the clustering algorithm has knowledge of the query in hand.

[2]While the term "generate" is convenient, we do not assume that clusters or documents literally generate labels, nor do we assume an underlying generative theory as in Lavernko and Croft [25] and Lavrenko [23], *inter alia*.

[3]The shift in notation and terminology from "$p(c|q) \overset{rank}{=}$" to "score of c" echoes the transition from using (model) probabilities to estimates of such probabilities.

Algorithm	Scoring function ($Score(c)$)
ClustCent	$Cent(c)$
ClustQueryGen	$p_c(q)$
ClustCent $\wedge$ ClustQueryGen	$Cent(c)p_c(q)$
DocCent	$\sum_{d_i \in c} p_{d_i}(c)Cent(d_i)$
DocQueryGen	$\sum_{d_i \in c} p_{d_i}(q)p_{d_i}(c)$
DocCent $\wedge$ DocQueryGen	$\sum_{d_i \in c} p_{d_i}(q)p_{d_i}(c)Cent(d_i)$
ClustCent $\wedge$ DocCent	$\lambda Cent(c) + (1-\lambda)\sum_{d_i \in c} p_{d_i}(c)Cent(d_i)$
ClustQueryGen $\wedge$ DocQueryGen	$\lambda p_c(q) + (1-\lambda)\sum_{d_i \in c} p_{d_i}(q)p_{d_i}(c)$
ClustRanker	$\lambda Cent(c)p_c(q) + (1-\lambda)\sum_{d_i \in c} p_{d_i}(q)p_{d_i}(c)Cent(d_i)$

Table 1: Summary of methods for ranking clusters.

2.2.1 *Document and cluster biases*

Following common practice in work on language-model-based retrieval we can use uniform distribution for the document prior $p(d)$ [8], and similarly assume a uniform distribution for the cluster prior $p(c)$. Such practice would have been a natural choice if the clusters we want to rank were produced in a *query-independent* fashion. However, we would like to exploit the fact that the clusters in $Cl(\mathcal{D}_{\text{init}})$ are composed of documents in the initially retrieved list $\mathcal{D}_{\text{init}}$. Case in point, since $\mathcal{D}_{\text{init}}$ was retrieved in response to q, documents in $\mathcal{D}_{\text{init}}$ that are considered as "reflecting" $\mathcal{D}_{\text{init}}$'s content might be good candidates for generating the label q [20]; a similar argument can be made for clusters in $Cl(\mathcal{D}_{\text{init}})$ that "reflect" its content. Therefore, instead of using "true" *prior* distributions, we use *biases* that represent the *centrality* [20] of documents with respect to $\mathcal{D}_{\text{init}}$ and the centrality of clusters with respect to $Cl(\mathcal{D}_{\text{init}})$.[4]

Specifically, we adopt a recently suggested approach to inducing document centrality [20] that is based on measuring the similarity of a document (in $\mathcal{D}_{\text{init}}$) to other central documents in $\mathcal{D}_{\text{init}}$. To quantify this recursive centrality definition, we compute PageRank's [3] stationary distribution over a graph wherein vertices represent documents in $\mathcal{D}_{\text{init}}$ and edge-weights represent inter-document language-model-based similarities [20]. We then set $p(d) \overset{def}{=} Cent(d)$ for $d \in \mathcal{D}_{\text{init}}$ and 0 otherwise, where $Cent(d)$ is d's PageRank score; hence, $p(d)$ is a probability distribution over the entire corpus $\mathcal{D}$.

Analogously, we set $p(c) \overset{def}{=} Cent(c)$ for $c \in Cl(\mathcal{D}_{\text{init}})$, where $Cent(c)$ is c's PageRank score as computed over a graph wherein vertices are clusters in $Cl(\mathcal{D}_{\text{init}})$ and edge-weights represent language-model-based inter-cluster similarities. ($p(c)$ is thereby a probability distribution over the given set of clusters $Cl(\mathcal{D}_{\text{init}})$.)

The construction method of the document and cluster graphs follows an approach for constructing document-solely graphs [20], and is elaborated in Appendix A.

Using the document and cluster induced biases, we can now fully instantiate Equation 3 to derive **ClustRanker**, our primary cluster-ranking algorithm:

$$Score_{ClustRanker}(c) \overset{def}{=} \lambda Cent(c)p_c(q) + (1-\lambda)\sum_{d_i \in c} p_{d_i}(q)p_{d_i}(c)Cent(d_i). \quad (4)$$

2.2.2 *Criteria for ranking clusters*

The ClustRanker algorithm ranks cluster c by integrating several criteria; specifically, (i) **ClustCent** — c's centrality ($Cent(c)$), (ii) **ClustQueryGen** — the possibility to generate the label q directly from c as measured by $p_c(q)$, (iii) **DocCent** — the centrality of c's constituent documents ($Cent(d)$), and (iv) **DocQueryGen** — the possibility to generate q by c's constituent documents as measured by $p_d(q)$. (Note that the latter two are combined with the cluster-document association strength $p_d(c)$).

To study the effectiveness of each of these criteria (and some of their combinations) for ranking clusters, we apply the following manipulations to the ClustRanker algorithm: (i) setting λ to 1 (0) to have only the cluster (documents) generate q, (ii) using uniform distribution for $Cent(c)$ (over $Cl(\mathcal{D}_{\text{init}})$) and/or for $Cent(d)$ (over $\mathcal{D}_{\text{init}}$) hence assuming that all clusters in $Cl(\mathcal{D}_{\text{init}})$ and/or documents in $\mathcal{D}_{\text{init}}$ are central to the same extent (we assume that the number of clusters in $Cl(\mathcal{D}_{\text{init}})$ is the same as the number of documents in $\mathcal{D}_{\text{init}}$, as is the case for the clustering method that we employ in Section 4), and (iv) setting $p_c(q)$ ($p_d(q)$) to the same constant value thereby assuming that all clusters in $Cl(\mathcal{D}_{\text{init}})$ (documents in $\mathcal{D}_{\text{init}}$) have the same probability of directly generating q. For instance, setting λ to 0 and $p_d(q)$ to some constant, we rank c by DocCent — the weighted-average of the centrality values of its constituent documents: $\sum_{d_i \in c} p_{d_i}(c)Cent(d_i)$. Table 1 presents the resultant cluster-ranking methods that we explore. ("$\wedge$" indicates that a method utilizes two criteria.)

3. RELATED WORK

Query-specific clusters are often used to visualize the results of search so as to help users to quickly detect the relevant documents [5, 14, 28, 27, 33, 36]. Leuski [27], for example, orders (hard) clusters in an interactive retrieval system by the highest query-similarity exhibited by any of their constituent documents. The cluster that contains the document most similar to the query is always ranked first, as is the case in [36]. Hence, such a ranking approach cannot be naturally employed with overlapping clusters, which are used in our experiments in Section 4 following previous work

[4]The biases are not "true" *prior* distributions, because of the virtue by which $\mathcal{D}_{\text{init}}$ was created, that is, in response to the query. However, we take care that the biases form valid probability distributions as we show later.

on cluster-based retrieval [13, 19, 21, 31, 30]. In contrast, our framework is not committed to any specific clustering technique, and the ClustRanker algorithm leverages information from *all* constituent documents of a cluster.

Some work uses information from query-specific clusters to *smooth* language models of documents in the initial list so as to improve the document-query similarity estimate [29, 17]. In a related vein, graph-based approaches for re-ranking the initial list utilize inter-document similarity information [20, 21, 9]. These approaches can potentially help to improve the performance of our ClustRanker algorithm, as they provide a higher-quality document ranking to begin with.

Ranking (both query-specific and *query-independent*) clusters in response to a query has traditionally been based on comparing a cluster representation with that of the query [15, 7, 43, 19, 29, 31, 30]. The ClustQueryGen criterion, which was used in prior work on ranking (hard) query-specific clusters in the language model framework [29], is a language-model manifestation of this ranking approach; we compare its effectiveness with that of the other ranking methods from Table 1 in Section 4.3.

Some previous cluster-based document-ranking models [19, 17] can be viewed as the conceptual "opposite" of our ClustRanker algorithm as they use clusters as proxies for ranking documents. However, these models use only query-similarity information while ClustRanker integrates such information with centrality information. In fact, we show in Section 4.3 that centrality information is more effective than query-similarity (generation) information for ranking query-specific clusters, and that their integration yields better performance than that of using each alone.

Recently, researchers have identified some properties of query-specific clusters that contain a high percentage of relevant documents [31, 18]; among which are the cluster-query similarity (ClustQueryGen) [31], the query similarity of the cluster's constituent documents (DocQueryGen) [31, 18], and the differences between the two [31]. These properties were utilized for automatically deciding whether to employ cluster-based or document-based retrieval in response to a query [31], and for ranking query-specific clusters [18]. The latter approach [18] relies on rankings induced by clusters' models over the entire corpus, in contrast to our approach that focuses on the "context" within the initially retrieved list. However, our centrality-based criteria can potentially be incorporated in this cluster-ranking framework [18].

Some recent work on ranking query-specific clusters resembles ours in that it utilizes cluster-centrality information [21]; in contrast to our approach, centrality is induced based on cluster-document similarities. We further discuss this approach and compare it to ours in Section 4.3.

4. EVALUATION

We next evaluate the effectiveness (or lack thereof) of our cluster-ranking methods in detecting query-specific clusters that contain a high percentage of relevant documents.

4.1 Language-Model Induction

For language model induction, we treat documents and queries as term sequences. While there are several possible approaches to represent clusters [27, 31], our focus here is on the underlying principles of our ranking framework. Therefore, we adopt an approach commonly used in previous work on cluster-based retrieval [19, 29, 21], and represent a cluster by the term sequence that results from concatenating its constituent documents. (The order of concatenation has no effect since we only define unigram language models that assume term independence.)

We use $p_x^{Dir[\mu]}(\cdot)$ to denote the unigram Dirichlet-smoothed language model induced from term sequence x (μ is the smoothing parameter) [46]. To avoid underflow issues when assigning language-model probabilities to long texts (as is the case for $p_d(c)$) [24, 19, 20], we adopt the following measure [19, 20, 21]:

$$p_y(x) \overset{def}{=} \exp\left(-D\left(p_x^{Dir[0]}(\cdot)\ \middle\|\ p_y^{Dir[\mu]}(\cdot)\right)\right),$$

where x and y are term sequences, and D is the Kullback-Leibler (KL) divergence. This estimate was mathematically shown to compensate for length issues [22, 20] and empirically demonstrated to be effective in settings wherein long texts are assigned with language-model probabilities [19, 20, 21].

Although the estimate just described does not constitute a probability distribution — as is the case for unigram language models — some previous work demonstrates the merits in using it as is without further normalization [20, 21].

4.2 Experimental Setup

We conducted our experiments on the following TREC corpora:

corpus	# of docs	queries	disk(s)
AP	242,918	51-64, 66-150	1-3
TREC8	528,155	401-450	4-5
WSJ	173,252	151-200	1-2

These data sets were used in some previous work on ranking query-specific clusters [21] with which we compare our methods. We used the titles of TREC topics for queries. We applied tokenization and Porter stemming via the Lemur toolkit (www.lemurproject.org), which was also used for language model induction.

We set $\mathcal{D}_{\text{init}}$, the list upon which clustering is performed, to be the 50 highest-ranked documents by an *initial ranking* induced over the entire corpus using $p_d^{Dir[\mu]}(q)$ — i.e., a standard language-model approach[5].

To produce the set $Cl(\mathcal{D}_{\text{init}})$ of query-specific clusters, we use a simple nearest-neighbor clustering approach that is known to yield (some) clusters that contain a high percentage of relevant documents [17, 30], and, more generally, was shown to be effective for cluster-based retrieval [13, 19, 17, 31, 30]. Specifically, given $d \in \mathcal{D}_{\text{init}}$ we define a cluster that contains d and the $k-1$ documents $d_i \in \mathcal{D}_{\text{init}}$ ($d_i \neq d$) that yield the highest language-model similarity $p_{d_i}(d)$. (We break ties by document IDs.)

We posed our cluster-ranking methods as means for increasing precision at the very top ranks of the returned document list. Thus, we evaluate a cluster-ranking method by the percentage of relevant documents in the most highly ranked cluster. Specifically, we use **p@k** to denote the percentage of relevant documents in a cluster of size k (either

[5]To create an initial ranking of a reasonable "quality", we set the smoothing parameter (μ) to a value that results in optimized MAP (calculated at a 1000 cutoff) performance. This also facilitates the comparison with some previous work on cluster-ranking [21], which employs the same approach for creating an initial list of 50 documents to be clustered.

5 or 10), because it is exactly the precision of the top k documents that is obtained if the cluster's (k) constituent documents are positioned at the top ranks of the results[6].

We optimize p@k performance for clusters of size k by selecting the value of λ, the interpolation parameter in the ClustRanker algorithm, from $\{0, 0.1, \ldots, 1\}$, and the values of the (two) parameters controlling the graph-construction methods (for inducing the document and cluster biases) from previously-suggested ranges [20]. (See Appendix A for further details on graph construction.) The value of μ, the document language model smoothing parameter, is set to 2000 following previous recommendations [46], except for estimating $p_d(q)$ where we use the value chosen for creating $\mathcal{D}_{\text{init}}$ so as to maintain consistency with the initial ranking.

Finally, we note that the computational overhead of our approach on top of the initial search is not significant. Case in point, clustering of top-retrieved documents (50 in our case) can be performed quickly [45] (our framework is not committed to a specific clustering approach), and computing PageRank scores over a graph of 50 documents (clusters) to induce document (cluster) centrality takes only a few iterations of the Power method [12].

4.3 Experimental Results

In what follows, we determine statistically significant differences of p@k performance by using Wilcoxon's two-sided test at a confidence level of 95%.

Comparison to document-based retrieval. The first question we are interested in is the effectiveness (or lack thereof) of the ClustRanker algorithm in comparison to that of the initial document ranking from which $\mathcal{D}_{\text{init}}$ is derived. Recall that ClustRanker ranks clusters of documents from $\mathcal{D}_{\text{init}}$, and is evaluated by the percentage of relevant documents in the cluster most highly ranked. As we can see in Table 2, ClustRanker posts performance that is (substantially) better than that of the initial ranking in all relevant comparisons (corpus $\times$ evaluation measure).

ClustRanker also tends to outperform a document-based language model approach, which ranks all documents in the corpus by $p_d(q)$ with the smoothing parameter μ set to optimize precision at top ranks. Case in point, the p@5 performance of such a p@5-optimized document-based retrieval approach is 46.5, 51.2, and 56.0 for AP, TREC8 and WSJ, respectively. The p@10 performance of such a p@10-optimized approach is 43.9, 46.4, and 49.4, respectively.

Comparison to pseudo-feedback-based retrieval. Our ClustRanker algorithm looks for relevant documents in $\mathcal{D}_{\text{init}}$ by exploiting clustering information. Pseudo-feedback-based query expansion approaches, on the other hand, define a query model based on $\mathcal{D}_{\text{init}}$ and use it for (re-)ranking the entire corpus [4, 44]. To contrast the two paradigms, we use the *relevance model* RM3 [25, 1, 10] — a state-of-the-art pseudo-feedback-based query-expansion approach. We set the values of the (three) free parameters of RM3 so as to independently optimize p@5 and p@10 performance. (See Appendix B for more details.)

	AP		TREC8		WSJ	
	p@5	p@10	p@5	p@10	p@5	p@10
init. rank.	45.7	43.2	50.0	45.6	53.6	48.4
ClustRanker	**52.7***	**50.6***	**57.6**	**50.6**	**56.0**	**51.2**

Table 2: Comparison of ClustRanker with the initial document ranking. Boldface: best result in a column; '*' marks statistically significant differences with the initial ranking.

	AP		TREC8		WSJ	
	p@5	p@10	p@5	p@10	p@5	p@10
RM3	50.3	48.6	53.6	48.2	**58.8**	**53.4**
ClustRanker	**52.7**	**50.6**	**57.6**	**50.6**	56.0	51.2

Table 3: Comparison of ClustRanker with a relevance model (RM3) [25, 1]. Boldface: best result in a column.

We can see in Table 3 that ClustRanker outperforms RM3 on AP and TREC8 and underperforms it on WSJ. (The performance differences, however, are not statistically significant.) These results are gratifying: we have shown that a method (ClustRanker) based on retrieving a cluster in its entirety can outperform both standard document-based retrieval (see the above), and can favorably compete with a state-of-the-art pseudo-feedback-based retrieval approach.

Criteria for ranking clusters. We now turn to analyze the performance of the various cluster-ranking criteria (methods) that ClustRanker integrates so as to study their relative contribution to its overall effectiveness. (Refer back to Table 1 for specification of the different methods.) The performance numbers are presented in Table 4.

Our first observation based on Table 4 is that using either of the two types of information (i.e., centrality and query-similarity (generation)), or both, for the cluster's constituent documents, yields in most relevant comparisons (corpora $\times$ evaluation metric) superior performance to that of using the same type(s) of information for the cluster as a whole. (Compare DocCent with ClustCent, DocQueryGen with ClustQueryGen, and DocCent $\wedge$ DocQueryGen with ClustCent $\wedge$ ClustQueryGen.) Nevertheless, using information induced both from the cluster as a whole and from its constituent documents yields performance that in many of the relevant comparisons transcends that of using information induced from either — e.g., compare ClustCent and DocCent with ClustCent $\wedge$ DocCent. These findings attest to the importance of integrating information induced from the cluster with that induced from its proxy documents — the idea behind our framework.

In comparing the two types of information used for ranking, that is, centrality and query-similarity (generation), we see that the former yields in most relevant comparisons superior performance to that of the latter. (Compare ClustCent with ClustQueryGen, DocCent with DocQueryGen, and ClustCent $\wedge$ DocCent with ClustQueryGen $\wedge$ DocQueryGen.) Nevertheless, combining both types of information

[6]An alternative previously-proposed evaluation approach is based on converting a ranking over *all* clusters in $Cl(\mathcal{D}_{\text{init}})$ to a ranking over *all* documents in $\mathcal{D}_{\text{init}}$ [27, 29, 21, 31]. However, the "quality" of the resultant document ranking heavily depends on the induced intra-cluster document ordering [27, 21] and on the overlap between clusters [21], and hence does not enable a "clean" evaluation of the percentage of relevant documents within clusters.

	AP		TREC8		WSJ	
	p@5	p@10	p@5	p@10	p@5	p@10
init. rank.	45.7	43.2	50.0	45.6	53.6	48.4
ClustCent	51.7	48.6*	52.4	49.4	54.8	50.0
ClustQueryGen	39.2*	38.8*	39.6*	40.6*	44.0*	37.0*
ClustCent ∧ ClustQueryGen	49.7	48.0*	55.2	50.4	52.4	47.8
DocCent	52.9*	48.8	52.0	48.8	55.6	50.6
DocQueryGen	43.6	46.7	47.6	43.2	55.2	47.0
DocCent ∧ DocQueryGen	52.7*	**50.6***	54.8	49.0	**56.0**	51.2
ClustCent ∧ DocCent	**53.5***	48.8*	54.8	49.8	**56.0**	**51.4**
ClustQueryGen ∧ DocQueryGen	43.6	46.7	47.6	43.2	55.2	47.8
ClustRanker	52.7*	**50.6***	**57.6**	**50.6**	**56.0**	51.2

Table 4: Comparison of the cluster-ranking methods from Table 1. Boldface marks the best result in a column and '*' indicates a statistically significant difference with the initial ranking.

can improve performance over that of using each alone, as is the case for DocCent ∧ DocQueryGen with respect to DocCent and DocQueryGen.

It is not a surprise, then, that the ClustRanker algorithm, which integrates centrality information and query-similarity (generation) information that are induced from both the cluster as a whole and from its constituent documents, is (in most relevant comparisons) the most effective cluster-ranking algorithm among those presented in Table 4.

Comparison to past approaches. Most previous approaches to ranking (various types of) clusters compare a cluster representation with that of the query [15, 7, 19, 29, 31]. Specifically, in the language model framework, (hard) query-specific clusters were ranked by the probability assigned by their induced language models to the query [29, 31]. Note that this is exactly the ClustQueryGen criterion for ranking clusters.

An additional reference comparison that we consider, which yields state-of-the-art performance in detecting clusters that contain a high relevant-document percentage, is a recently-proposed (bipartite-)graph-based approach for ranking query-specific clusters [21]. Specifically, documents in $\mathcal{D}_{\text{init}}$ are vertices on one side, and clusters in $Cl(\mathcal{D}_{\text{init}})$ are vertices on the other side; an edge connects document d with the δ clusters c_i that yield the highest language-model similarity $p_{c_i}(d)$, which also serves as a weight-function for the edges. Then, Kleinberg's **HITS** (hubs and authorities) algorithm [16] is run on the graph, and clusters are ranked by their induced *authority* values. It was shown that the cluster with the highest authority value tends to contain a high percentage of relevant documents [21]. For implementation, we follow the details described in [21]; specifically, we choose the value of δ from $\{2, 4, 9, 19, 29, 39, 49\}$ so as to optimize p@k performance for clusters of size k.

Table 5 presents the comparison of ClustRanker with the reference comparisons just described (ClustQueryGen and HITS). We can see that ClustRanker outperforms both reference comparisons in all cases. Moreover, many of the performance differences are also statistically significant.

The HITS-based algorithm [21] utilizes cluster-centrality information as induced over a cluster-document graph. Our ClustRanker algorithm, on the other hand, integrates centrality information (induced over document-solely and cluster-solely graphs) with query-similarity (generation) information. In Table 6 we contrast the resultant performance of using the different notions of centrality utilized by the two algorithms. (We present the performance of our centrality-solely-based methods ClustCent, DocCent, and ClustCent ∧ DocCent and of the HITS approach [21].)

	AP		TREC8		WSJ	
	p@5	p@10	p@5	p@10	p@5	p@10
ClustQueryGen	39.2	38.8	39.6	40.6	44.0	37.0
HITS	49.5^c	47.2^c	50.8^c	46.6	53.6^c	49.0^c
ClustRanker	$\mathbf{52.7^c}$	$\mathbf{50.6^c}$	$\mathbf{57.6^{ch}}$	$\mathbf{50.6^c}$	$\mathbf{56.0^c}$	$\mathbf{51.2^c}$

Table 5: Comparison of ClustRanker with the ClustQueryGen [29] and HITS [21] methods for ranking clusters. Boldface: best performance in a column; 'c' and 'h' mark statistically significant differences with ClustQueryGen and HITS, respectively.

We can see in Table 6 that all our centrality-solely-based approaches outperform the HITS-based method [21] in all relevant comparisons. These results attest to the effective utilization of (a specific type of) centrality information by our framework. We hasten to point out, however, that ClustCent and DocCent incorporate two free parameters and ClustCent ∧ DocCent incorporates three, while the HITS-based approach incorporates one free parameter.

Further analysis. The derivation of the ClustRanker algorithm is based on truncating the summation in Equation 2

	AP		TREC8		WSJ	
	p@5	p@10	p@5	p@10	p@5	p@10
HITS	49.5	47.2	50.8	46.6	53.6	49.0
ClustCent	51.7	48.6	52.4	49.4	54.8	50.0
DocCent	52.9^h	**48.8**	52.0	48.8	55.6	50.6
ClustCent ∧ DocCent	$\mathbf{53.5^h}$	**48.8**	**54.8**	**49.8**	**56.0**	**51.4**

Table 6: Comparison of our centrality-solely approaches for ranking clusters with the HITS-based method [21]. Boldface: best performance in a column; 'h' marks statistically significant differences with the HITS method.

	AP		TREC8		WSJ	
	p@5	p@10	p@5	p@10	p@5	p@10
init. rank.	45.7	43.2	50.0	45.6	53.6	48.4
$d \in \mathcal{D}_{\text{init}}$	49.5	47.6	54.0	49.8	52.8	49.6
$d \in c$	**52.7***	**50.6***	**57.6**	**50.6**	**56.0**	**51.2**

Table 7: Performance numbers of ClustRanker when either *all* documents in $\mathcal{D}_{\text{init}}$ serve as proxies for cluster c (denoted $d \in \mathcal{D}_{\text{init}}$), or when only c's constituent documents serve as its proxies, as in the original implementation (denoted $d \in c$). Boldface: best result in a column; '*' marks statistically significant differences with the initial ranking.

(Section 2) so as to allow only c's constituent documents to serve as its proxies. We now study a variant of ClustRanker wherein *all* documents in the initial list $\mathcal{D}_{\text{init}}$ can serve as c's proxies:

$$Cent(c)p_c(q) + (1-\lambda) \sum_{d_i \in \mathcal{D}_{\text{init}}} p_{d_i}(q) p_{d_i}(c) Cent(d_i).$$

As can be seen in Table 7, this variant (represented by the row labeled "$d \in \mathcal{D}_{\text{init}}$") posts performance that is almost always better than that of the initial document ranking from which $\mathcal{D}_{\text{init}}$ is derived. However, the performance is also consistently worse than that of the original implementation of ClustRanker (represented by the row labeled "$d \in c$") that lets only c's constituent documents to serve as its proxies; furthermore, the suggested variant never posts statistically significant improvements over the initial ranking as opposed to the original implementation of ClustRanker. (The performance differences between the two variants of ClustRanker, however, are not statistically significant.) Thus, as is mentioned in Section 2, using only the cluster's constituent documents as its proxies is not only computationally convenient, but also yields performance improvements.

5. CONCLUSION

We presented a novel language model approach to ranking *query-specific* clusters by the presumed percentage of relevant documents that they contain. Our model integrates information induced from the cluster as a whole unit with information induced from documents that are associated with the cluster. We demonstrated the superiority of our model to previous cluster-ranking methods in detecting clusters that contain a high percentage of relevant documents. Furthermore, we showed that posting the constituent documents of the cluster most-highly ranked by our model at the top of the returned results yields precision-at-top-ranks performance that is superior to that of the initial document ranking upon which clustering is performed; the performance also favorably competes with that of a state-of-the-art pseudo-feedback-based document-retrieval approach.

Acknowledgments The author thanks the anonymous reviewers for their helpful comments. The author also thanks Lillian Lee for discussions that led to ideas presented in this paper; specifically, the cluster-centrality induction method is a fruit of joint work with Lillian Lee. This paper is based upon work done in part while the author was at Cornell University and upon work supported in part by the National Science Foundation under grant no. IIS-0329064 and by a gift from Google. Any opinions, findings and conclusions or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsoring institutions.

6. REFERENCES

[1] N. Abdul-Jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, M. D. Smucker, and C. Wade. UMASS at TREC 2004 — novelty and hard. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC-13)*, 2004.

[2] L. Azzopardi, M. Girolami, and K. van Rijsbergen. Topic based language models for ad hoc information retrieval. In *Proceedings of International Conference on Neural Networks and IEEE International Conference on Fuzzy Systems*, pages 3281–3286, 2004.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the 7th International World Wide Web Conference*, pages 107–117, 1998.

[4] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using SMART: TREC3. In *Proceedings of the Third Text Retrieval Conference (TREC-3)*, pages 69–80, 1994.

[5] http://www.clusty.com.

[6] M. Connell, A. Feng, G. Kumaran, H. Raghavan, C. Shah, and J. Allan. UMass at TDT 2004. TDT2004 System Description, 2004.

[7] W. B. Croft. A model of cluster searching based on classification. *Information Systems*, 5:189–195, 1980.

[8] W. B. Croft and J. Lafferty, editors. *Language Modeling for Information Retrieval.* Number 13 in Information Retrieval Book Series. Kluwer, 2003.

[9] F. Diaz. Regularizing ad hoc retrieval scores. In *Proceedings of the Fourteenth International Conference on Information and Knowledge Managment (CIKM)*, pages 672–679, 2005.

[10] F. Diaz and D. Metzler. Improving the estimation of relevance models using large external corpora. In *Proceedings of SIGIR*, pages 154–161, 2006.

[11] F. Geraci, M. Pellegrini, M. Maggini, and F. Sebastiani. Cluster generation and cluster labeling for Web snippets: A fast and accurate hierarchical solution. In *Proceedings of the 13th international conference on string processing and information retrieval (SPIRE)*, pages 25–37, 2006.

[12] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, third edition, 1996.

[13] 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. Reprinted in Karen Sparck Jones and Peter Willett, eds., *Readings in Information Retrieval*, Morgan Kaufmann, pp. 365–373, 1997.

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

[15] N. Jardine and C. J. van Rijsbergen. The use of hierarchic clustering in information retrieval. *Information Storage and Retrieval*, 7(5):217–240, 1971.

[16] J. Kleinberg. Authoritative sources in a hyperlinked environment. Technical Report Research Report RJ 10076, IBM, May 1997.

[17] O. Kurland. *Inter-document similarities, language models, and ad hoc retrieval.* PhD thesis, Cornell University, 2006.

[18] O. Kurland and C. Domshlak. A rank-aggregation approach to searching for optimal query-specific clusters. In *Proceedings of SIGIR*, 2008.

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

[20] O. Kurland and L. Lee. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR*, pages 306–313, 2005.

[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. D. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of SIGIR*, pages 111–119, 2001.

[23] V. Lavrenko. *A Generative Theory of Relevance.* PhD thesis, University of Massachusetts Amherst, 2004.

[24] V. Lavrenko, J. Allan, E. DeGuzman, D. LaFlamme, V. Pollard, and S. Thomas. Relevance models for topic

detection and tracking. In *Proceedings of the Human Language Technology Conference (HLT)*, pages 104–110, 2002.

[25] V. Lavrenko and W. B. Croft. Relevance-based language models. In *Proceedings of SIGIR*, pages 120–127, 2001.

[26] V. Lavrenko and W. B. Croft. Relevance models in information retrieval. In Croft and Lafferty [8], pages 11–56.

[27] A. Leuski. Evaluating document clustering for interactive information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Managment (CIKM)*, pages 33–40, 2001.

[28] A. Leuski and J. Allan. Evaluating a visual navigation system for a digital library. In *Proceedings of the Second European conference on research and advanced technology for digital libraries (ECDL)*, pages 535–554, 1998.

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

[30] 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.

[31] X. Liu and W. B. Croft. Representing clusters for retrieval. In *Proceedings of SIGIR*, pages 671–672, 2006. Poster.

[32] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD international conference*, pages 490–499, 2007.

[33] C. R. Palmer, J. Pesenty, R. Veldes-Perez, M. Christel, A. G. Hauptmann, D. Ng, and H. D. Wactlar. Demonstration of hierarchical document clustering of digital library retrieval results. In *Proceedings of the 1st ACM/IEEE-CS joint conference on digital libraries*, page 451, 2001.

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

[35] S. E. Preece. Clustering as an output option. In *Proceedings of the American Society for Information Science*, pages 189–190, 1973.

[36] 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 the Twelfth Text Retrieval Conference (TREC-12)*, pages 152–160, 2003.

[37] L. Si, R. Jin, J. Callan, and P. Ogilvie. A language modeling framework for resource selection and results merging. In *Proceedings of the 11th International Conference on Information and Knowledge Managment (CIKM)*, pages 391–397, 2002.

[38] 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.

[39] P. Treeratpituk and J. Callan. Automatically labeling hierarchical clusters. In *Proceedings of the sixth national conference on digital government research*, pages 167–176, 2006.

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

[41] E. M. Voorhees. The cluster hypothesis revisited. In *Proceedings of SIGIR*, pages 188–196, 1985.

[42] X. Wei and W. B. Croft. LDA-based document models for ad-hoc retrieval. In *Proceedings of SIGIR*, 2006.

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

[44] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of SIGIR*, pages 4–11, 1996.

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

[46] 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.

APPENDIX

A. CENTRALITY INDUCTION

We briefly describe a previously proposed graph-based approach for inducing document centrality [20], which we use for inducing document and cluster centrality.

Let S (either $\mathcal{D}_{\text{init}}$ — the initial list of documents, or $Cl(\mathcal{D}_{\text{init}})$ — the set of their clusters) be a set of items, and $G = (S, S \times S)$ be the complete directed graph defined over S.

The weight $wt(s_1 \rightarrow s_2)$ of the edge $s_1 \rightarrow s_2$ ($s_1, s_2 \in S$) is defined as

$$wt(s_1 \rightarrow s_2) \overset{def}{=} \begin{cases} p_{s_2}(s_1) & \text{if } s_2 \in Nbhd(s_1; \delta), \\ 0 & \text{otherwise,} \end{cases}$$

where $Nbhd(s_1; \delta)$ is the set of δ items $s' \in S - \{s_1\}$ that yield the highest $p_{s'}(s_1)$. (Ties are broken by item ID.)

We use the PageRank approach [3] to smooth the edge-weight function:

$$wt^{[\nu]}(s_1 \rightarrow s_2) = \nu \cdot \frac{1}{|S|} + (1 - \nu) \cdot \frac{wt(s_1 \rightarrow s_2)}{\sum_{s' \in S} wt(s_1 \rightarrow s')};$$

ν is a free parameter.

Thus, G with the edge-weight function $wt^{[\nu]}$ constitutes an ergodic Markov chain, for which a stationary distribution exists. We set $Cent(s)$, the centrality value of s, to the stationary probability of "visiting" s.

Following previous work [20], the values of δ and ν are chosen from $\{2, 4, 9, 19, 29, 39, 49\}$ and $\{0.05, 0.1, \ldots, 0.9, 0.95\}$, respectively, so as to optimize the p@k performance of a given algorithm for clusters of size k. We use the *same* parameter setting for the document-graph ($S = \mathcal{D}_{\text{init}}$) and for the cluster-graph ($S = Cl(\mathcal{D}_{\text{init}})$), and therefore inducing document and cluster centrality in any of our methods is based on two free parameters.

B. RELEVANCE MODEL

To estimate the standard relevance model RM1, which was shown to yield better performance than that of the RM2 relevance model [26], we follow Lavrenko and Croft [26]. Let w denote a term in the vocabulary, $\{q_i\}$ be the set of query terms, and $p_d^{JM[\alpha]}(\cdot)$ denote a Jelinek-Mercer smoothed document language model with smoothing parameter α [46]. RM1 is then defined by

$$p_{RM1}(w; \alpha) \overset{def}{=} \sum_{d \in \mathcal{D}_{\text{init}}} p_d^{JM[\alpha]}(w) \frac{\prod_i p_d^{JM[\alpha]}(q_i)}{\sum_{d_j \in \mathcal{D}_{\text{init}}} \prod_i p_{d_j}^{JM[\alpha]}(q_i)}.$$

In practice, RM1 is *clipped* by setting $p_{RM1}(w; \alpha)$ to 0 for all but the β terms with the highest $p_{RM1}(w; \alpha)$ to begin with [6, 10]; further normalization is performed to yield a probability distribution, which we denote by $\tilde{p}_{RM1}(\cdot; \alpha, \beta)$. To improve performance, RM1 is anchored to the original query [1, 10] to yield the RM3 model:

$$p_{RM3}(w; \alpha, \beta, \gamma) \overset{def}{=} \gamma p_q^{MLE}(w) + (1 - \gamma) \tilde{p}_{RM1}(w; \alpha, \beta);$$

$p_q^{MLE}(w)$ is the maximum likelihood estimate of term w with respect to q. Documents in the corpus are then ranked by the KL divergence $D\left(p_{RM3}(\cdot; \alpha, \beta, \gamma) \,\middle\|\, p_d^{Dir[\mu]}(\cdot)\right)$.

The free-parameter values are chosen from the following ranges to independently optimize p@5 and p@10 performance: $\alpha \in \{0, 0.1, 0.3, \ldots, 0.9\}$, $\beta \in \{25, 50, 75, 100, 500, 1000, 5000, ALL\}$, where "$ALL$" stands for using all terms in the corpus (i.e., no clipping), and $\gamma \in \{0, 0.1, 0.2, \ldots, 0.9\}$; μ is set to 2000, as in our cluster-based algorithms, following previous recommendations [46].