Better than the real thing? Iterative pseudo-query processing using cluster-based language models

Oren Kurland [1,3]
kurland@cs.cornell.edu

Lillian Lee [1,2,3]
llee@cs.cornell.edu

Carmel Domshlak [4]
dcarmel@ie.technion.ac.il

1. Computer Science Department, Cornell University, Ithaca NY 14853, U.S.A.
2. Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA 15213, U.S.A.
3. Computer Science Department, Carnegie Mellon University, Pittsburgh PA 15213, U.S.A.
4. William Davidson Faculty of Industrial Engineering and Management, Technion, Haifa 32000, Israel

ABSTRACT

We present a novel approach to pseudo-feedback-based ad hoc retrieval that uses language models induced from both documents and clusters. First, we treat the pseudo-feedback documents produced in response to the original query as a set of *pseudo-queries* that themselves can serve as input to the retrieval process. Observing that the documents returned in response to the pseudo-queries can then act as pseudo-queries for subsequent rounds, we arrive at a formulation of pseudo-query-based retrieval as an iterative process. Experiments show that several concrete instantiations of this idea, when applied in conjunction with techniques designed to heighten precision, yield performance results rivaling those of a number of previously-proposed algorithms, including the standard language-modeling approach. The use of *cluster-based* language models is a key contributing factor to our algorithms' success.

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

General Terms: Algorithms, Experimentation

Keywords: language modeling, clustering, pseudo-feedback, pseudo-queries, rendition, query drift, cluster-based language models, aspect recall

1. INTRODUCTION

Statistical language models have become an important tool in information retrieval, and have been applied to many settings [7]. In the case of fully automatic ad hoc IR, where the task is to find documents relevant to a query q without access to relevance-feedback information, a great deal of recent research builds upon Ponte and Croft's initial proposal [24] wherein the rank of a document d is based on the probability assigned to q by a language model constructed from d. We can gloss this ranking principle as, "retrieve the documents that are the best *renderers* of the query".[1]

The work presented in this paper is partly motivated by the following hypothesis: documents that are the best *renderers* of a query may be good alternate *renditions* of it. Indeed, a basic premise behind query-expansion techniques utilizing pseudo-feedback is that top-retrieved documents may reveal dimensions of the user's information need that are not obvious from the original (short) query [26].[2] Assuming for now that the hypothesis is true (we discuss it further below), we therefore propose a type of pseudo-feedback approach in which query "expansion" consists of wholesale replacement of q with a list of *pseudo-queries* consisting of the query's best renderers.

Pseudo-queries are clearly a form of pseudo-feedback. However, the former term suggests that once we have created pseudo-queries from the initial query, we can in principle repeat the process, this time seeking the top renderers of the pseudo-queries. And if the pseudo-queries are indeed more informative than their predecessor(s), then we expect this repetition to improve the retrieval results. We thus arrive at an iterative *boot-strapping* approach in which the previously-retrieved best renderers become the pseudo-queries for the next round.

Unfortunately, pseudo-feedback quality can suffer from problems with both precision and "aspect recall". The currently unavoidable phenomenon of non-relevant documents appearing in the retrieval results leads to *query drift*, "the alteration of the focus of a search topic caused by improper expansion" [21][3]. As for recall, key aspects of the user's information need may be completely missing from the pool of top-retrieved documents, due to both small pool size (in order to keep precision reasonable) and selection for docu-

[1] Our choice of terminology — "renderers" rather than "generators" — reflects the fact that we do not assume that documents (or their induced language models) are the source that "generates" q: A monkey randomly striking typewriter keys may produce a word-for-word copy of *Hamlet*, but we do not therefore say that it is the author of the play.

[2] This is another motivation behind our terminology: in the hands of a skilled artist, a rendition of a particular piece may be faithful to the original in many respects, and yet still be superior overall.

[3] We are not referring to "query drift" in the sense of user interests changing over time [1].

SIGIR'05, August 15–19, 2005, Salvador, Brazil.

ments most resembling the short — and hence potentially not completely informative — initial query. The inability to cope with this missing-aspect problem is viewed as a major failing of current systems [4, 10].

To increase aspect recall, we use the structure of the corpus, as manifested through language models built on document clusters, to suggest and represent potential facets of the user's needs [13, 20]. In particular, we consider finding good renderers among a set of *clusters* rather than the set of documents: a cluster that is a good renderer may contain documents that, while relevant, superficially don't match the query string precisely because they include aspects not immediately evident in q. (Such documents could be present in the cluster by dint of being similar to other relevant documents with respect to non-query terms.)

As for query drift, the problem would seem to be exacerbated by the multiple iterations performed by our algorithm, since poor-quality input early in the pipeline has a potentially disastrous effect on retrieval results later on. To cope with this difficulty, we provide a number of methods that "re-anchor" pseudo-queries to the original query.

Experiments with several large corpora reveal significant improvements in both average precision and recall over the standard language-modeling approach (which corresponds to a degenerate version of our methods). This finding suggests that pseudo-queries may indeed be better than the original query as a basis for retrieval. Moreover, comparisons against two highly effective techniques incorporating pseudo-relevance feedback — Rocchio on pseudo-feedback and Lavrenko and Croft's language-model-based *relevance model* [18] — show that our cluster-based methods can often provide competitive or superior performance.

2. RETRIEVAL FRAMEWORK

We now present a suite of fully automatic iterative algorithms for processing pseudo-queries. As mentioned above, all our algorithms conform to the same general format: find the best renderers of a current set of pseudo-queries; then, repeat the process using these best renderers as the new pseudo-queries.

We begin by establishing some notation and conventions in Section 2.1. Then, we discuss the two main axes along which our algorithms vary. The first such axis, detailed in Section 2.2, is the basic definition of a good renderer; the options we consider are: (1) a document that is a good renderer of at least one pseudo-query; (2) a document that is a good renderer of multiple pseudo-queries; and (3) a cluster that is a good renderer of multiple pseudo-queries. The second main axis of algorithm variation, discussed in Section 2.3, is the choice of mechanism for preventing query drift. One idea we pursue is to incorporate the rendition probability assigned to the original query.

2.1 Notation and Conventions

Throughout this section, $\mathcal{D}$ denotes a given document set, which induces a fixed vocabulary. The notation q^* indicates the user's initial query, and N stands for the number of documents to be returned in response to q^* when the retrieval process terminates. Lower-case Greek letters indicate algorithm parameters that were varied in our experiments.

We use $\mathcal{C}$ to refer to a set of clusters of the documents in $\mathcal{D}$ (in our work, $\mathcal{C}$ is computed prior to retrieval time). We freely switch between thinking of a cluster as a subset of $\mathcal{D}$ and thinking of it as the single text string created by concatenating its constituent documents in some pre-defined order[4] — this allows us to treat queries, documents, and clusters uniformly as sequences of terms.

The algorithms we present engage in iterative processing of pseudo-queries. In what follows, we assume that in each of the ρ rounds, the pseudo-query input consists of a ranked list $\widehat{Q} = \widehat{q}_1, \widehat{q}_2, \ldots$ together with a *weight* function $\widehat{w} : \widehat{Q} \to [0, 1]$ such that $\widehat{w}(\widehat{q}_i) \geq \widehat{w}(\widehat{q}_{i+1})$. In the first iteration, $\widehat{Q} = q^*$, and we set $\widehat{w}(q^*)$ to be 1. In subsequent iterations, $\widehat{Q}$ is an ordering of the documents in $\mathcal{D}$.

A key concept in our work is that of a renderer r's *repertoire* among a set of text strings, by which we mean the subset of the strings that r is a *top renderer* of. We therefore make the following definitions. Let $p_r(x)$ denote an estimate of the probability that r (in our work, either a document or a cluster) renders the text sequence x.

DEFINITION 1. *Let x be a text sequence, and let R be a finite set of potential renderers. Then, x's* top k renderers *in R, denoted* $\mathrm{TopRen}(x; R, k)$, *is the set of k items $r \in R$ that yield the highest*[5] $p_r(x)$.

DEFINITION 2. *For renderer r, set of text strings X, set of potential renderers $R \ni r$, and positive integer k, we define the* repertoire of r in X with respect to R and k *as*

$$\mathrm{Rep}(r; X \mid R, k) \stackrel{def}{=} \{x \in X : r \in \mathrm{TopRen}(x; R, k)\}.$$

For compactness, we suppress X, R, and/or k in our notation when no confusion can result.

Note that repertoires are sets, not sorted lists; thus, when we say that a pseudo-query $\widehat{q} \in \mathrm{Rep}(r)$ is "highly ranked", we mean that it occurs early in $\widehat{Q}$, as opposed to, say, that its rendition probability $p_r(\widehat{q})$ is large with respect to the other members of $\mathrm{Rep}(r)$.

2.2 Basic Methods for Scoring Renderers

We now present three basic options for determining the best renderers of a given iteration's pseudo-queries. Each such method M takes as input the ranked list of pseudo-queries $\widehat{Q}$ and the associated pseudo-query weights $\widehat{w}(\widehat{q}_i)$ and produces a score $\mathrm{Score}_M(d)$ for each document d. The input to the next round can then be created by setting $\widehat{w}(d) = \mathrm{Score}_M(d)$ and sorting all the documents in $\mathcal{D}$ by this quantity, unless additional mechanisms for coping with query drift are applied (see Section 2.3).

We begin by restricting our consideration of possible renderers to documents. The **Viterbi Doc-Audition** scoring method is a straightforward procedure that ranks those documents with repertoires containing a highly-weighted pseudo-query above those that are top renderers only of lower-weighted ones. Specifically, in each round, it first returns the top τ renderers of $\widehat{q}_1$, then the top τ renderers of $\widehat{q}_2$ (repeated renderers discarded), and so on, with the list of top renderers d of each $\widehat{q}_i$ sorted in descending order of $p_d(\widehat{q}_i)$.[6] Hence, suppose we have two top renderers of $\widehat{q}_1$, d and d',

[4]In our work, the concatenation order is irrelevant since we use unigram language models.

[5]Throughout this paper, we assume that the corpus documents and clusters are identified by numeric labels, with ties broken in favor of lower-numbered items.

[6]Although it does not convey more insight than the English

such that $p_d(\widehat{q}_1) > p_{d'}(\widehat{q}_1)$. Then d would be ranked above d' even if $p_d(\widehat{q}_i) \ll p_{d'}(\widehat{q}_i)$ for every other $\widehat{q}_i$.

If we were confident that $\widehat{q}_1$ is indeed the best representation of the user's information need, then such behavior is not unreasonable. In practice, however, such confidence may not be warranted; rather, we might consider a document that renders many of the pseudo-queries to be potentially as good or better a candidate for retrieval than a document that renders only one. Hence, we define an alternative scoring method, **Doc-Audition**. In essence, it rewards a potential renderer $d \in \mathcal{D}$ for every pseudo-query in its repertoire, although less credit is assigned for low-weight $\widehat{q}_i$ and for $\widehat{q}_i$ that d assigns a low rendition probability to. Specifically, ranks are induced by the following function:

$$\mathrm{Score}_{\mathrm{Doc}}(d) \stackrel{def}{=} \sum_{\widehat{q} \in \mathrm{Rep}(d|\tau)} \widehat{w}(\widehat{q}) \cdot \frac{p_d(\widehat{q})}{K(\widehat{q}; m)}, \quad (1)$$

where the re-scaling term $K(\widehat{q}; m) = \sum_{d' \in \mathrm{TopRen}(\widehat{q}; \mathcal{D}, m)} p_{d'}(\widehat{q})$ serves to compare $p_d(\widehat{q})$ to the rendition probabilities of the top m renderers[7] of $\widehat{q}$. Observe that in the first round, only the top renderers of the original query can receive non-zero scores.

Like all pseudo-feedback techniques, both scoring methods just presented can help ameliorate the crucial "aspect recall" problem. We expect that the most improvement relative to retrieval based directly on the original query q^* will occur when the pseudo-queries contain effective search terms not appearing in q^*, and among the best renderers of these pseudo-queries are relevant documents containing the missing terms but not having high overlap with q^*. Our third basic scoring method, **Cluster-Audition**, makes use of document clusters to take more explicit advantage of such situations. In particular, we choose renderers of the pseudo-queries from the set $\mathcal{C}$ of document *clusters* rather than from the set of documents. Ideally, the best renderers would be clusters consisting only of relevant documents (thus increasing recall), and some of these clusters would represent information whose importance is implied by the query but not explicitly mentioned by it (thus increasing "aspect recall"). In any event, integrating clusters into language-model-based retrieval has recently been shown to yield substantial performance improvements [13, 20].

While there are a huge number of clustering methods to choose from, some quite sophisticated, we simply create one cluster for every document by grouping together that document's top γ renderers. This method is convenient for us since we already need to compute top renderers; moreover, it has proven effective in previous work, perhaps because the highly overlapping clusters can be seen as representing different *facets* of the similarity structure of the corpus [13]. Indeed, there is a history of successful applications of the general nearest-neighbor approach (e.g., [9]).

Within each iteration, Cluster-Audition scoring consists of two phases. In the first, each cluster c is credited for every pseudo-query in its repertoire. Note that in computing repertoires, we chose to restrict the set of possible top renderers of a pseudo-query $\widehat{q}$ to $\mathcal{C}(\widehat{q})$, the set of clusters containing $\widehat{q}$: a cluster is only "allowed" to render its constituent documents[8]. (For the sake of readability, we suppress this restriction in the repertoire notation below.) We thus have:

$$\mathrm{Score}_{\mathrm{Clust}}(c) \stackrel{def}{=} \sum_{\widehat{q} \in \mathrm{Rep}(c|\tau)} \widehat{w}(\widehat{q}) \cdot \frac{p_c(\widehat{q})}{K_1(\widehat{q})}, \quad (2)$$

where $K_1(\widehat{q}) = \sum_{c' \in \mathcal{C}(\widehat{q})} p_{c'}(\widehat{q})$ re-scales c's rendition probability with respect to the set of clusters containing $\widehat{q}$.

The purpose of the second phase of scoring is to convert the implicit cluster ranking just computed into a document ranking, since the output of each round should be a legal set of final retrieval results. This is achieved by crediting each document for every cluster it is one of the top σ renderers of, with the restriction (again suppressed in the repertoire notation below to enhance readability) that a document may only render a cluster that it belongs to:

$$\mathrm{Score}_{\mathrm{Clust}}(d) \stackrel{def}{=} \sum_{c \in \mathrm{Rep}(d|\sigma)} \mathrm{Score}_{\mathrm{Clust}}(c) \cdot \frac{p_d(c)}{K_2(c)}, \quad (3)$$

where $K_2(c) = \sum_{d' \in c} p_{d'}(c)$ re-scales d's rendition probability with respect to the set of documents within c.

Remarks. If the desired values of τ and σ (the parameters controlling the sizes of the top-renderer sets) and γ (the parameter for cluster size) are known beforehand, then the clusters and top-renderer sets can be computed off-line, greatly reducing the amount of computation required at retrieval time. However, even if these parameters are not pre-specified, one can pre-compute a ranking of all possible renderers of each document; this still results in significant computational savings at run-time.

We note that the iterative processes based upon the latter two of the basic scoring schemes we have just described can be conceptualized as a fixed-length random walk on a graph corresponding to a Markov chain whose structure is determined by top-renderer relationships. In fact, the Cluster-Audition scoring method is reminiscent of the term-to-document Markov chain used by Lafferty and Zhai [15] for query expansion. However, in our case it is not clear what insight is gained by such a formulation; for instance, how any stationary distribution should be interpreted in the context of the retrieval task at hand is not obvious.

Finally, notice that all three methods just outlined contain the standard language-modeling approach [24] as a degenerate one-iteration (or half-iteration, for Cluster-Audition) case where $\tau = |\mathcal{D}|$, and for Cluster-Audition, the cluster set $\mathcal{C}$ corresponds to a partition of $\mathcal{D}$ into single-element sets.

description just given, for the sake of completeness, here is a formulation of the Viterbi Doc-Audition scoring method in terms of an explicit score function. For a given document $d \in \mathcal{D}$, let $\widehat{q}^+$ be the highest-ranked pseudo-query $\widehat{q}$ in $\mathrm{Rep}(d|\tau)$, and let i^+ be $\widehat{q}^+$'s rank. (If d is not a top renderer of any pseudo-query, we set i^+ to $|\mathcal{D}| + 1$ and $\widehat{q}^+$ to the dummy value q^*.) Then, we define

$$\mathrm{Score}_{\mathrm{VDoc}}(d) \stackrel{def}{=} \frac{p_d(\widehat{q}^+) + 2(|\mathcal{D}| - i^+ + 1)}{1 + 2|\mathcal{D}|}.$$

[7]In our experiments, we fixed m to a value guaranteed to be greater than τ to handle cases where more than τ documents had high rendition probabilities for q. However, preliminary experiments indicated that choosing $m = \tau$ did not substantially alter results.

[8]In the first iteration only, we treat the original query as a document belonging to all clusters.

2.3 Coping with Query Drift

We have previously mentioned that one of our main interests is increasing so-called "aspect recall". Naturally, we want to simultaneously retain high precision as well. However, a potential drawback of our iterative approach to pseudo-query processing is that engaging in multiple rounds threatens to exacerbate query drift: early contamination of the set of pseudo-queries with non-relevant documents can seriously skew downstream pseudo-query sets away from the user's true information needs. Using clusters adds even more risk of overgeneralization. We therefore propose a number of methods for addressing the query-drift problem. All follow the same general strategy: ensure that information from the original query q^* plays a large role.

Two indirect techniques all our algorithms employ involve choosing appropriate values for certain parameters. First, we limit ρ, the total number of rounds, to a relatively small number. Second, since the initial iteration is the one that is "closest" to the original query, we give it privileged status by considering the top τ_1, rather than τ, renderers of q^*.

We also consider a number of *re-scoring* techniques; these directly use $p_d(q^*)$ in some combination with the output of one of the three basic scoring methods M introduced above. Recall that without re-scoring, the pseudo-query weights $\widehat{w}(d)$ for round $t+1$ would simply be the score assigned by M to document d at the end of round t.

We borrow two re-scoring techniques from research on cluster-based retrieval within the language-modeling framework [13, 20]. Both affect only the output of the final round, since they were introduced in the context of non-iterative methods. The **$\mathbf{P}^*$ Interpolation** technique derives a new final score for each document d by linear interpolation of $\text{Score}_M^{(\rho)}(d)$, d's score in the final round according to M, with the rendition probability that d assigns to the original query (after rescaling both quantities with respect to their maximum values to ensure comparability):

$$\lambda \text{Score}_M^{(\rho)}(d) + (1-\lambda) p_d(q^*).$$

It thus integrates our iterated estimate of document relevance with surface document-query similarity.

In contrast, the **Truncated $\mathbf{P}^*$ Re-rank** method first discards all but the top N documents according to $\text{Score}_M^{(\rho)}(d)$; each remaining document d is then given the new score $p_d(q^*)$. Note that this method does not affect recall, since only the order of the retrieved results is changed.

Alternatively, we could alter the scores at the end of every round, rather than just the final one, as an attempt to counteract query drift early in the process. One idea, implemented in the **Iterated Truncation** technique, is to consider only the top N documents in a given iteration or pass to be likely to be informative pseudo-queries for the next round; the scores of all the other documents are therefore zeroed. The **Iterated Truncated $\mathbf{P}^*$ Re-rank** technique goes even further by additionally changing the scores of the top N documents to $p_d(q^*)$. That is, the Truncated P* Re-rank technique is applied to each round, rather than just to the final one. Similarly, we can apply P* Interpolation at each round, thus yielding the **Iterated $\mathbf{P}^*$ Interpolation** technique. Note that this method is more conservative than the original P* Interpolation technique because it tends to prevent pseudo-queries with low surface similarity to the query from being assigned high scores in early rounds.

2.4 Estimating Rendition Probabilities

Rendition probabilities are the foundation upon which all our algorithms are built. To describe the method by which we estimate them, we first introduce some preliminary concepts. Let y be either a text string or a set of text strings. Denoting the number of times a term w occurs in y by $\text{tf}(w \in y)$, for an n-term text sequence $w_1 w_2 \cdots w_n$ we define

$$p_y^{ML}(w_1 w_2 \cdots w_n) \stackrel{def}{=} \prod_{j=1}^{n} \frac{\text{tf}(w_j \in y)}{\sum_{w'} \text{tf}(w' \in y)};$$

this is commonly known as y's *maximum likelihood estimate* (MLE) for the sequence. The *Dirichlet-smoothed* version of the MLE is defined as

$$p_y^{[\mu]}(w_1 w_2 \cdots w_n) \stackrel{def}{=} \prod_{j=1}^{n} \frac{\text{tf}(w_j \in y) + \mu \cdot p_{\mathcal{D}}^{ML}(w_j)}{\sum_{w'} \text{tf}(w' \in y) + \mu},$$

where the smoothing parameter μ controls the degree of reliance on relative frequencies in the corpus rather than on the counts in y.

While the Dirichlet-smoothed unigram language model just defined has been used directly [32, 20], we adopt the following variant: for renderer r and text sequence x, we set

$$\begin{aligned} p_r(x) &\stackrel{def}{=} \exp\left(-D\left(p_x^{ML}(\cdot) \,||\, p_r^{[\mu]}(\cdot)\right)\right) \\ &\propto \sqrt[|x|]{p_r^{[\mu]}(x)}, \end{aligned}$$

where D is the KL divergence, which has formed the basis for other ranking principles as well [30, 15, 22, 13]; the two arguments to D are treated as distributions over terms rather than term sequences; and the omitted factor (which is of independent interest in other contexts [14]) drops out in the re-scaling performed by the Doc-Audition and Cluster-Audition scoring methods. Our formulation provides some mathematical justification for Lavrenko et al.'s "heuristic adjustment" [17], proposed to handle underflow problems in processing long documents, to take the *geometric mean* of $p_r^{[\mu]}(x)$ rather than $p_r^{[\mu]}(x)$ itself.

3. RELATED WORK

The fundamental principle underlying pseudo-feedback-based methods is that the top-ranked documents retrieved in response to a query may contain additional information regarding the user's information need. The canonical approach is to treat the pseudo-feedback documents as if they had actually been deemed relevant by the user, and then apply relevance-feedback techniques [26] to them. One such line of work is to use the feedback documents to re-weight query terms and/or to identify additional terms with which to augment the query; since our wholesale replacement of the query with pseudo-queries can be considered an extension of this idea, in Section 4 we compare against one well-known instantiation, namely, Rocchio [25]. Within the language-modeling retrieval framework, treating the feedback documents as relevant often means estimating rendition probabilities using the feedback pool (where the members may be differentially weighted) as data [18, 16, 15, 31, 29].

Lafferty and Zhai [15] proposed an iterative probability-estimation sub-routine that alternates between terms and documents, which is remiscent of the shifting between clusters and documents that our Cluster-Audition algorithm represents; but their intended application is not a direct scoring of potential retrieval candidates. Lavrenko and Croft's *relevance model* algorithm [18] has the same goal as ours, is also based on language models, and posts state-of-the-art performance; Section 4 describes it in more detail and reports the results of our experimental comparisons against it.

Query drift has long been recognized as a key concern for pseudo-feedback approaches [21, 6], and hence a number of coping techniques have been previously introduced. One example is the application of boolean filters and term co-occurrence analysis [21]. The techniques we adopted focus instead on incorporating rendition probabilities for the original query, borrowing from previous work [31, 20, 13].

4. EXPERIMENTS

To examine the effectiveness of our algorithms and to determine how much various aspects of our proposed retrieval framework contribute, we designed a number of evaluation experiments.

First, we compare the performance of our algorithms to that of a language-model-based approach (henceforth *baseline*) in which documents are ranked according to $p_d(q^*)$. This comparison serves not only to see whether our methods can outperform an effective retrieval system, but also to highlight the merits (or lack thereof) of engaging in multiple iterations of pseudo-query processing, since, as noted above, conceptually the basic language-modeling approach corresponds to a single round or pass of our algorithms.

We also test our techniques for pseudo-query-processing against a well known and highly effective pseudo-feedback method, the Rocchio algorithm [25] as applied to top-retrieved documents.

Finally, we study whether our particular ways of utilizing language models are beneficial by testing how well they perform against the *relevance model* [18, 19]. The latter approach takes a generative perspective: assuming that there is a single *relevance* language model $\mathcal{R}$ underlying the creation of both q^* and the documents relevant to q^*, documents are ranked by their degree of "match" with $\mathcal{R}$, rather than by how well they directly match the query or set of pseudo-queries. In implementation, Lavrenko and Croft estimate $\mathcal{R}$ by combining the language models of those documents assigning the highest rendition probabilities to q^*. Thus, the relevance-model, similarly to our algorithms, is a pseudo-feedback-based language-modeling approach, but clearly the specific way in which document-based language models are used is quite different from the ways our algorithms employ them, and Lavrenko and Croft made no explicit mention of clusters.

Although our reference comparison models operate in different spaces (vector space vs. the probability simplex), in [19] it is observed that if *i.i.d sampling* sampling is used for relevance model estimation, then both the pseudo-feedback version of Rocchio and the relevance model utilize a linear combination of the top-retrieved documents' models to construct an *expanded query model.*

We conducted our experiments on the following three corpora, drawn from TREC data:

corpus	# of docs	queries	# of relevant docs
AP89	84,678	1-46,48-50	3261
AP88+89	164,597	101-150	4805
LA+FR	187,526	401-450	1391

The AP89 corpus was pre-processed with the Porter stemmer. For AP88+89 the Krovetz stemmer was used, and both INQUERY stopwords [3] and length-one tokens were removed to comply with the processing policy in [18]. For LA+FR, which is part of the TREC-8 corpus, neither stemming nor stopword removal was applied. It is relatively heterogeneous, and the LA dataset with TREC8 queries is considered to be difficult [11].

For queries, we used the titles of TREC topics rather than the full descriptions, resulting in short queries containing 2-5 terms on average.

We use both average non-interpolated precision and recall at $N = 1000$ as our evaluation measures. Statistically significant differences in performance are determined using the two-sided Wilcoxon test at the 95% confidence level.

4.1 Implementation

We employed the Lemur toolkit [23] for a number of our experiments. To collect pseudo-feedback, we used $p_d(q)$ to create an initial ranking All parameters were set to values optimizing average non-interpolated precision.

Our implementation of Rocchio used the vector-space model with log tf.idf term weighting to represent queries and documents. Similarity was measured via the inner product. The free parameters were: (i) τ_1 - the number of top-retrieved documents used for feedback, (ii) the number of terms to augment the original query with, and (iii) the weighting coefficient for the augmenting terms (we only used positive feedback). Note that while the number of (augmenting) terms is not modeled in Rocchio's original method, we varied it to obtain better performance and comply with the optimization steps we implemented for the relevance model (details further below). Our implementation yielded accuracies consistent with previously reported (optimized) results.

Our Lemur-based implementation of the relevance model utilized i.i.d sampling (following [19]) to construct $\mathcal{R}$; the divergence $D\left(\mathcal{R} \mid\mid p_d(\cdot)\right)$ served as ranking criterion. The free parameters were τ_1 (the number of top-retrieved documents) and an interpolation parameter controlling the evaluation of the top retrieved documents' language models.

We also experimented with *clipping* the relevance model [5, 8] to assign non-zero probability to only a restricted number of terms (up to a maximum of several hundred). This modification can be viewed as regulating the degree of query expansion, or as an efficiency-improving heuristic.

Some of our algorithms have quite a few free parameters. To help prevent our algorithms from enjoying an unfair advantage due to this fact alone, we implemented the following policies. The language models forming the basis of both our methods and the baseline had the Dirichlet smoothing parameter value fixed at $\mu = 2000$, following [32]. All parameters shared by our methods and the relevance models (e.g., τ_1) were set identically for all the algorithms, with the following search ranges:

τ, the number of top renderers considered: $\{5, 10, 20, \ldots, 100\}$ for Viterbi Doc-Audition; $\{5, 10, 20, 30, 40\}$ for Doc-Audition; from $\{1, 2, 3, 4\}$ for Cluster-Audition.

τ_1, the number of best renderers retrieved at the first iteration: $\{5\} \cup \{10, 20, ..., 100\} \cup \{200, 300, 400, 500\}$.

σ, the number of documents to which a cluster's score is distributed (Equation 3): {5,10,20,30,40} for AP89 and AP88+89; {5,10} for LA+FR.

γ, cluster size: 40 for AP89 and AP88+89; 10 for LA+FR.

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
Baseline	20.74%	48.67%	24.26%	66.62%	21.72%	48.81%
VDoc	*23.12%**	*55.20%*	*28.28%**	63.68%	*22.07%*	47.81%
Doc	*22.81%**	*57.34%*	*28.27%**	*71.13%*	*22.48%**	**59.67%**
Int-Doc	*24.48%**	*59.77%**	*30.28%**	*75.57%**	*22.98%**	*56.00%**
Clust	*23.43%**	**63.57%***	*28.76%**	**80.15%***	*23.24%**	*56.79%*
Int-Clust	**24.56%***	*62.77%**	**31.09%***	*76.15%**	**23.40%**	*54.57%**

Table 1: Comparison against the baseline. Statistically significant differences with the baseline are marked with a star (*). Bold: best performance for each setting (column). Italics: results superior to the baseline.

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
Clust	23.43%	**63.57%**	28.76%$^{\mathcal{R}c}$	80.15%$^{\mathcal{R}c}$	23.24%$^{\eta}$	**56.79%**$^{\mathcal{R}}$
Int-Clust	24.56%$^{\eta}$	62.77%	31.09%	76.15%$^{\mathcal{R}c}$	**23.40%**$^{\eta}$	54.57%$^{\eta\mathcal{R}}$
Rocchio	22.85%	58.23%	30.69%	76.02%	18.21%	52.26%
RelModel	24.72%	58.08%	**32.72%**	81.60%	22.03%	46.59%
ClippedRelModel	**26.17%**	63.48%	32.51%	**82.10%**	22.34%	56.65%

Table 2: Comparison of the best pseudo-query algorithms against Rocchio, the relevance model, and the clipped relevance model. Statistically significant differences with these algorithms are marked with η, $\mathcal{R}$, and c, respectively. Bold: best performance for each setting (column).

ρ, the number of rounds: 1–2, Cluster-Audition; 1–5, Viterbi Doc-Audition and Doc-Audition.

4.2 Main results

We report results using the following abbreviations.

VDoc	Viterbi Doc-Audition
Doc	Doc-Audition
Clust	Cluster-Audition

The prefix "Int-" indicates that the (non-iterated) P* Interpolation technique was employed for coping with query drift; in all other cases, Truncated P* Re-rank was applied. Results for other query-drift amelioration mechanisms are reported later in this section.

Table 1 compares our algorithms' performance to that of the language-model baseline. We see that almost all our methods outperform the language model in both average precision and recall, often to a statistically significant degree. Moreover, it is clear that our cluster-based algorithm Cluster-Audition, in either its Truncated P* Re-rank or P* Interpolation version, yields the best results, outperforming not only the language-modeling approach but all the non-cluster-based algorithms we have proposed. This finding further reinforces conclusions previously drawn in the literature regarding the advantages of using clusters to represent cross-document contextual information [13, 20]; and the fact that we see especially large improvements in recall provides partial support towards our hypothesis that clusters can potentially alleviate the problem of aspect recall.

Moving on to our second main comparison, Table 2 shows that our cluster-based methods usually yield results that are better (sometimes to a significant degree) than those of Rocchio. Comparing our algorithms' performance to that of the two versions of the relevance model, we observe the following: on AP89, our cluster-based methods yield results that are, in a statistical sense, indistinguishable from those of the (clipped) relevance model; on LA+FR our methods tend to be superior to the (clipped) relevance model (sometimes significantly so); but on AP88+89, the (clipped) relevance model generally performs significantly better than our cluster-based methods. In interpreting these results, though, it is crucial to note that (1) our implementation of the (clipped) relevance model involved an extremely wide-ranging search over the parameter space, whereas as discussed in Section 4.1, our methods only explored moderate parameter-setting ranges; and (2) clipping is a heuristic that could potentially be adapted for use by our document- or cluster-based language models.

While some preliminary results indicate that the performance of our methods can be further improved by more exhaustive parameter tuning, we believe that the main message of Table 2 is that we can achieve performance competitive with optimized state-of-the-art pseudo-feedback methods with relatively little optimization effort.

4.3 Further analysis

Examination of the average-precision results in Table 1 reveals that the P* Interpolation technique is usually more effective at coping with query drift than Truncated P* Re-rank. Table 3 provides a more extensive comparison of the full set of query-drift-prevention techniques we have proposed. For simplicity, we report only the results of applying these methods in conjunction with the Doc-Audition algorithm. It is apparent that most of our techniques achieve comparable or better precision than is obtained by the original method by itself, and that P* Interpolation is the "winner", although the more conservative iterated version (Iterated P* Interpolation) ties it on two corpora. Investigation into the optimal parameter settings revealed that, while performance for Doc-Audition itself was optimized at a low number of iterations (which would have the effect of keeping precision from degrading), performance when prevention techniques were applied was best for a larger number of iterations, enabling increase in recall along with preservation of high precision rates.

	AP89		AP88+89		LA+FR	
	prec	recall	prec	recall	prec	recall
none	22.83%	51.43%	28.49%	69.99%	21.55%	53.20%
Truncated P* Re-rank	22.81%	57.34%	28.27%	71.13%	22.48%	**59.67%**
P* Interpolation	**24.48%**	**59.77%**	**30.28%**	**75.57%**	**22.98%**	56.00%
Iterated Truncation	19.71%	51.09%	26.86%	63.37%	16.84%	31.63%
Iterated Truncated P* Re-rank	22.76%	58.02%	27.96%	68.05%	22.49%	59.45%
Iterated P* Interpolation	24.27%	59.09%	**30.28%**	**75.57%**	**22.98%**	56.00%

Table 3: Comparison of techniques for query drift prevention, the Doc-Audition scoring method. Bold: best performance for a given evaluation setting (column). Note that Truncated P* Re-rank improves recall over the basic algorithm (none) as it achieves optimal precision with a different parameter setting.

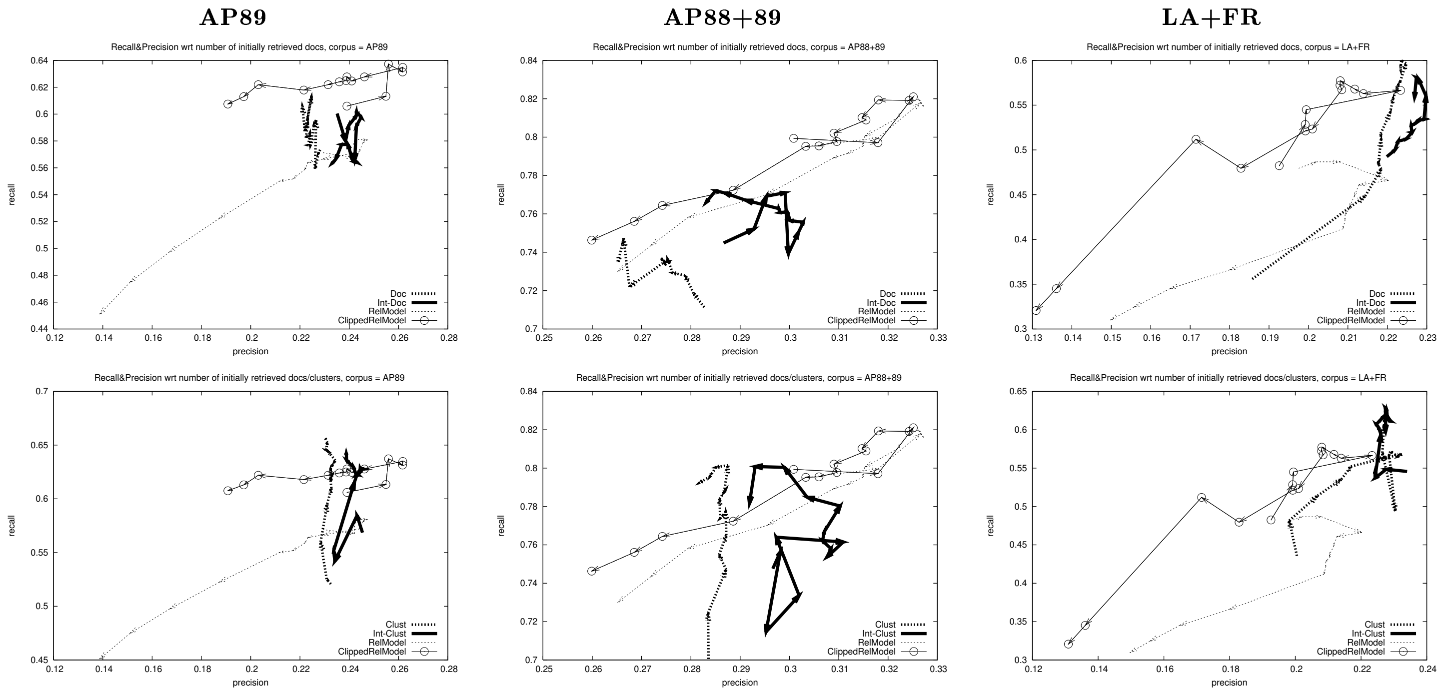

Figure 1: Average precision vs. recall as τ_1, the number of initially retrieved best renderers for the original query, takes the values 5,10,20,...100,200,...,500. Top and bottom rows show the best non-cluster- and cluster-based algorithms, respectively, against the (clipped) relevance model (note the relatively severe degradation in precision — "leftwards motion" — of the latter). To show detail, the plots are not to the same scale.

Finally, we explored a well-known weakness of language-model-based approaches using pseudo-feedback: sensitivity to the number τ_1 of documents initially retrieved [28]. First, we see from Figure 1 that the precision of our novel algorithms is much less affected by increases in τ_1 than the precision of the (clipped) relevance model, which indicates the merits of both of our query-drift prevention techniques (although clearly P* Interpolation almost always outperforms Truncated P* Re-rank at all values of τ_1). It is also interesting to observe that increasing τ_1 tends to have a positive influence on the recall of our cluster-based methods (which, after all, were posited to improve aspect recall), whereas eventually it has a negligible or negative influence on the (clipped) relevance model's recall. In short, the performance curves for our algorithms tend to move vertically, whereas the (clipped) relevance model's curves seem to exhibit more horizontal movement. These trends suggest that our methods and the relevance model have complementary strengths.

5. CONCLUSIONS

We presented a novel iterative pseudo-feedback approach to ad hoc information retrieval using cluster-based language models. Starting from the original query, our methods repeatedly seek potentially good renderers of a current set of pseudo-queries, guided by the hypothesis that documents that are the best renderers of a pseudo-query may be good alternate renditions of it.

One of the major challenges facing today's retrieval engines is the problem of "aspect recall". To alleviate this problem, we proposed to take advantage of corpus structure via the consideration of cluster-based language models as potential renderers; the key idea is that clusters can serve as a rich source of information regarding corpus aspects. Likewise, we examined several techniques for reducing *query drift*, which is yet another obstacle that both traditional and language-modeling-based pseudo-feedback approaches need to overcome. As evidence that our techniques are effective, we saw that our algorithms showed significant improvements in performance with respect to a standard language-modeling approach, and produced results rivaling those of other state-of-the-art pseudo-feedback methods.

For future work, we plan to look into analyzing our methods in real-feedback settings, e.g., [27, 12, 2]. Furthermore, we would like to incorporate and examine additional clustering approaches for modeling corpus structure.

Acknowledgments. We thank David Fisher for technical assistance with Lemur, and Jon Kleinberg and the anonymous reviewers for valuable discussions and comments. We also thank CMU for its hospitality during the year. This paper is based upon work supported in part by the National Science Foundation (NSF) under grant no. IIS-0329064 and CCR-0122581; SRI International under subcontract no. 03-000211 on their project funded by the Department of the Interior's National Business Center; and by an Alfred P. Sloan Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views or official policies, either expressed or implied, of any sponsoring institutions, the U.S. government, or any other entity.

6. REFERENCES

[1] James Allan. Incremental relevance feedback for information filtering. In *Proceedings of SIGIR*, pages 270–278, 1996.

[2] James Allan. HARD track overview in TREC 2003: High accuracy retrieval from documents. In *Proceedings of the Twelfth Text Retrieval Conference (TREC-12)*, pages 24–37, 2003.

[3] James Allan, Margaret E. Connell, W. Bruce Croft, Fang-Fang Feng, David Fisher, and Xiaoyan Li. INQUERY and TREC-9. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 551–562, 2001. NIST Special Publication 500-249.

[4] Chris Buckley. Why current IR engines fail. In *Proceedings of SIGIR*, pages 584–585, 2004. Poster.

[5] Margaret Connell, Ao Feng, Giridhar Kumaran, Hema Raghavan, Chirag Shah, and James Allan. UMass at TDT 2004. TDT2004 System Description, 2004.

[6] W. Bruce Croft and D. J. Harper. Using probabilistic models of document retrieval without relevance information. *Journal of Documentation*, 35(4):285–295, 1979. Reprinted in Karen Sparck Jones and Peter Willett, eds., *Readings in Information Retrieval*, Morgan Kaufmann, pp. 339–344, 1997.

[7] W. Bruce Croft and John Lafferty, editors. *Language Modeling for Information Retrieval.* Number 13 in Information Retrieval Book Series. Kluwer, 2003.

[8] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. A language modeling framework for selective query expansion. Technical Report IR-338, Center for Intelligent Information Retrieval, University of Massachusetts, 2004.

[9] Alan Griffiths, H. Claire Luckhurst, and Peter 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.

[10] Donna Harman and Chris Buckley. The NRRC reliable information access (RIA) workshop. In *Proceedings of SIGIR*, pages 528–529, 2004. Poster.

[11] Xiao Hu, Sindhura Bandhakavi, and ChengXiang Zhai. Error analysis of difficult TREC topics. In *Proceedings of SIGIR*, pages 407–408, 2003. Poster.

[12] IJsbrand Jan Aalbersberg. Incremental relevance feedback. In *Proceedings of SIGIR*, pages 11–22, 1992.

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

[14] Oren Kurland and Lillian Lee. PageRank without hyperlinks: Structural re-ranking using links induced by language models. In *Proceedings of SIGIR*, 2005.

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

[16] Victor Lavrenko. Optimal mixture models in IR. In *European Conference on Information Retrieval*, pages 193–212, 2002.

[17] Victor Lavrenko, James Allan, Edward DeGuzman, Daniel LaFlamme, Veera Pollard, and Steven Thomas. Relevance models for topic detection and tracking. In *Proceedings of the Human Language Technology Conference (HLT)*, pages 104–110, 2002.

[18] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In *Proceedings of SIGIR*, pages 120–127, 2001.

[19] Victor Lavrenko and W. Bruce Croft. Relevance models in information retrieval. In Croft and Lafferty [7], pages 11–56.

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

[21] Mandar Mitra, Amit Singhal, and Chris Buckley. Improving automatic query expansion. In *Proceedings of SIGIR*, pages 206–214, 1998.

[22] Kenney Ng. A maximum likelihood ratio information retrieval model. In *Proceedings of the Eighth Text Retrieval Conference (TREC-8)*, pages 483–492, 2000.

[23] Paul Ogilvie and Jamie Callan. Experiments using the LEMUR toolkit. In *Proceedings of the Tenth Text Retrieval Conference (TREC-10)*, pages 103–108, 2001.

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

[25] Joseph John Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[26] Ian Ruthven and Mounia Lalmas. A survey on the use of relevance feedback for information access systems. *Knowledge Engineering Review*, 18(2):95–145, 2003.

[27] Ian Soboroff and Stephen E. Robertson. Building a filtering test collection for TREC 2002. In *Proceedings of SIGIR*, pages 243–250, 2003.

[28] Tao Tao and ChengXiang Zhai. A mixture clustering model for pseudo feedback in information retrieval. In *Proceedings of the International Federation of Classification Societies (IFCS)*, 2004. Invited paper.

[29] Tao Tao and ChengXiang Zhai. A two-stage mixture model for pseudo feedback. In *Proceedings of the 27th SIGIR*, pages 486–487, 2004. Poster.

[30] Jinxi Xu and W. Bruce Croft. Cluster-based language models for distributed retrieval. In *Proceedings of SIGIR*, pages 254–261, 1999.

[31] Chengxiang Zhai and John D. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of CIKM*, pages 403–410, 2001.

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