Corpus Structure, Language Models, and Ad Hoc Information Retrieval

Oren Kurland and Lillian Lee
Department of Computer Science
Cornell University
Ithaca, NY 14853-7501
{kurland,llee}@cs.cornell.edu

ABSTRACT

Most previous work on the recently developed *language-modeling* approach to information retrieval focuses on document-specific characteristics, and therefore does not take into account the structure of the surrounding corpus. We propose a novel algorithmic framework in which information provided by document-based language models is enhanced by the incorporation of information drawn from *clusters* of similar documents. Using this framework, we develop a suite of new algorithms. Even the simplest typically outperforms the standard language-modeling approach in precision and recall, and our new *interpolation* algorithm posts statistically significant improvements for both metrics over all three corpora tested.

Categories and Subject Descriptors

H3.3 [**Information Search and Retrieval**]: Language models, clustering, smoothing

General Terms

Algorithms, Experiments

Keywords

language modeling, aspect models, interpolation model, clustering, smoothing, cluster-based language models

1. INTRODUCTION

As is well known, a basic problem in information retrieval is to determine how relevant a particular document is to a query. In the automatic ad hoc retrieval setting, examples of relevant documents are not supplied. Given this absence of explicit relevance evidence, it is important to consider what other information sources can be exploited.

In methods patterned after the classic tf.idf document-vector approach to text representation, the focus is mostly on utilizing within-document features, such as term frequencies. Information drawn from the corpus as a whole generally consists of aggregates of statistics gathered from each document considered in isolation; for example, the inverse document frequency is based on checking, for each document, whether that document contains a particular term.

Recent work has demonstrated the effectiveness of an alternative approach wherein probabilistic models of text generation are constructed from documents, and these induced *language models* (LMs) are used to perform document ranking [15, 5]. Like tf.idf and related techniques, though, language-modeling methods typically use only individual-document features and corpus-wide aggregates of the same. (Corpus term counts are generally employed for *smoothing*, so that unseen text can be assigned non-zero probability.)

Neither of the aforementioned approaches typically makes use of a potentially very powerful source of information: the *similarity structure* of the corpus. Clusters are a convenient representation of similarity whose potential for improving retrieval performance has long been recognized [4, 16]. From our point of view, one key advantage is that they provide smoothed, more representative statistics for their elements, as has been recognized in statistical natural language processing for some time [3]. For example, we could infer that a document not containing a certain query term is still relevant if the document belongs to a cluster whose component documents generally do contain the term.

However, relying on clusters alone has some potential drawbacks. Clustering at retrieval time can be very expensive, but off-line clustering seems, by definition, query-independent and therefore may be based on factors that are irrelevant to user information need. Also, cluster statistics may overgeneralize with respect to specific member documents.

We therefore propose a framework for incorporating both corpus-structure information — using pre-computed, overlapping clusters — and individual-document information. Importantly, although cluster formation is query-independent, within our framework the *choice* of which clusters to incorporate *can* depend on the query. We then consider several of the many possible algorithms arising as specific instantiations of our framework. These include both novel methods and, as special cases, both the standard, non-cluster-based LM approach and a variant of the cluster-based aspect model [9].

Our empirical evaluation consists of experiments in an array of settings created by varying several parameters and meta-parameters; these include the corpus, the information

SIGIR'04, July 25–29,2004, Sheffield, South Yorkshire, UK.

representation (e.g., language models versus tf.idf-style vectors), and, when applicable, the smoothing method selected. We find that even the worst-performing of our novel algorithms is competitive with the LM approach, and indeed always provides substantial improvement in recall. In general, our algorithms provide good performance in comparison to a number of recently proposed methods, thus demonstrating that our integration approach to incorporating document and corpus-structure information is an effective way to improve ad hoc retrieval.

Notational conventions. We use d, q, c and $\mathcal{C}$ to denote a document, query, cluster, and corpus, respectively. A fixed vocabulary is assumed. We use the notation $p_d(\cdot)$ for the *language model* — which assigns probabilities to text strings over the fixed vocabulary — induced from d by some pre-specified method, and $p_c(\cdot)$ for the language model induced from c. (Section 5 describes the induction methods we used in our experiments.)

It is convenient to use *Kronecker delta* notation $\delta[s]$ to set up some definitions. The argument s is a statement; $\delta[s] = 1$ if s holds, 0 otherwise.

2. RETRIEVAL FRAMEWORK

As noted above, when we rank documents with respect to a query, we desire per-document scores that rely both on information drawn from the particular document's contents and on how the document is situated within the similarity structure of the ambient corpus.

Structure representation via overlapping clusters. Document clusters are an attractive choice for representing corpus similarity structure (see [16, chapter 3] for extended discussion). Clusters can be thought of as *facets* of the corpus that users might be interested in. Given that a particular document can be relevant to a user for several reasons, or to different users for different reasons, we believe that a set of overlapping clusters[1] forms a better model for similarity structure than a partitioning of the corpus. Furthermore, employing intersecting clusters may reduce information loss due to the generalization that clustering can introduce [16, pg. 44].

Information representation. Motivated by the empirical successes of language-modeling-based approaches [15, 5], we use language models induced from documents and clusters as our information representation. Thus, $p_d(q)$ and $p_c(q)$ specify our initial knowledge of the relation between the query q and a particular document d or cluster c, respectively. (However, Section 6 shows that using a tf.idf representation also yields performance improvements with respect to the appropriate baseline, though not to the same degree as using language models does.)

Information integration. To assign a ranking to the documents in a corpus $\mathcal{C}$ with respect to q, we want to score each $d \in \mathcal{C}$ against q in a way that incorporates information from query-relevant corpus facets to which d belongs. While one could compute clusters specific to q at retrieval time, efficiency considerations compel us to create Clusters($\mathcal{C}$), the set of clusters, in advance, and hence in a query-independent fashion. To compensate, at retrieval time we base the *choice* of appropriate facets on the query.

How might cluster information be used? Our discussion above indicates that clusters can serve two roles. Insofar as they approximate true facets of the corpus, they can aid in the *selection* of relevant documents: we would want to retrieve those that belong to clusters corresponding to facets of interest to the user. On the other hand, clusters also have the capacity to *smooth* individual-document language models, since they pool statistics from multiple documents. Finally, we must remember that over-reliance on $p_c(q)$ can over-generalize by failing to account for document-specific information encoded in $p_d(q)$.

These observations motivate the algorithm template shown in Figure 1. This template is fairly general: both the standard language-modeling approach [15] and the aspect model [9] are concrete instantiations. In the template, the choice of Facets$_q(d)$ corresponds to utilizing clusters in their selection role. The scoring step can be thought of as integrating $p_d(q)$ with cluster-based language models in their smoothing role. The optional re-ranking step is used as a way to further bias the final ranking towards document-specific information, if desired. Note that re-ranking can change the average non-interpolated precision but not the absolute precision or recall of the retrieval results; we therefore use it, when necessary, to enhance average precision. (Section 6 reports experiments studying its efficacy.)

Offline: Create Clusters($\mathcal{C}$)
Given q and N, the number of documents to retrieve:
For each $d \in \mathcal{C}$,
Choose a cluster subset Facets$_q(d) \subseteq$ Clusters($\mathcal{C}$)
Score d by a weighted combination of $p_d(q)$ and the $p_c(q)$'s for all $c \in$ Facets$_q(d)$
Set TopDocs(N) to the rank-ordered list of N top-scoring documents
Optional: re-rank $d \in$ TopDocs(N) by $p_d(q)$
Return TopDocs(N)

Figure 1: Algorithm template.

In the next section, we describe a number of specific algorithms arising from this template, concentrating on their degree of dependence on cluster-induced language models.

3. RETRIEVAL ALGORITHMS

Table 1 summarizes the algorithms we consider, which represent a few choices out of the many possible ways to instantiate the template of Figure 1. Our preference in picking these algorithms has been towards simpler methods, so as to focus on the impact of using cluster information (as opposed to the impact of tuning many weighting parameters).

First step: Cluster formation and selection. There are many algorithms that can be used to create Clusters($\mathcal{C}$), the set of overlapping document clusters required by Figure 1's template. In our experiments, we simply have each document d form the *basis* of a cluster Cohort(d) consisting of d and its $k-1$ nearest neighbors, where k is a free parameter. (Note that two clusters with different basis documents may

[1] We include soft or probabilistic clusters in this category.

	$\text{Facets}_q(d)$	Score	Re-rank by $p_d(q)$?
LM	N/A	$p_d(q)$	(redundant)
basis-select	$\{\text{Cohort}(d)\} \cap \text{TopClusters}_q(m)$	$p_d(q) \cdot \delta[\lvert\text{Facets}_q(d)\rvert > 0]$	(redundant)
set-select	$\{c : d \in c\} \cap \text{TopClusters}_q(m)$	$p_d(q) \cdot \delta[\lvert\text{Facets}_q(d)\rvert > 0]$	(redundant)
bag-select	$\{c : d \in c\} \cap \text{TopClusters}_q(m)$	$p_d(q) \cdot \lvert\text{Facets}_q(d)\rvert$	yes
uniform-aspect-x	$\{c : d \in c\} \cap \text{TopClusters}_q(m)$	$\sum_{c \in \text{Facets}_q(d)} p_c(q)$	yes
aspect-x	$\{c : d \in c\} \cap \text{TopClusters}_q(m)$	$\sum_{c \in \text{Facets}_q(d)} p_c(q) \cdot p_c(d)$	yes
interpolation	$\{c : d \in c\} \cap \text{TopClusters}_q(m)$	$\lambda \cdot p_d(q) + (1-\lambda) \sum_{c \in \text{Facets}_q(d)} p_c(q) \cdot p_c(d)$	no

Table 1: Algorithm specifications.

contain the same set of documents.) Inter-document distance is measured by the Kullback-Leibler (KL) divergence between the corresponding (smoothed) language models, as in [11].

The idea behind our use of cohorts is that a document's nearest neighbors in similarity space represent a local "fragment" or "tile" of the overall similarity structure of the corpus. Our evaluation results show that even this relatively unsophisticated way to approximate facets enables effective leveraging of corpus structure; at the very least, it serves as a form of nearest-neighbor smoothing (see below).

The first retrieval-time action specified by our algorithm template is to choose $\text{Facets}_q(d)$, a query-dependent subset of $\text{Clusters}(\mathcal{C})$. In all the algorithms described below except the baseline (which doesn't use cluster information), there is a document-selection aspect to this subset, in that only documents in some $c \in \text{Facets}_q(d)$ can appear in the final ranked-list output. Ideally, we would use the clusters best approximating those (true) facets of the corpus that are most representative of the user's interests, as expressed by q; therefore, we require that $\text{Facets}_q(d)$ be a subset of $\text{TopClusters}_q(m)$, the top m clusters c with respect to $p_c(q)$. But we also want to evaluate d only with respect to the facets it actually exhibits. Thus, in what follows (except for the baseline), $\text{Facets}_q(d)$ is always defined to be a subset of $\{c : d \in c\} \cap \text{TopClusters}_q(m)$; we assume m is large enough to produce the desired number of retrieved documents N.

Baseline method. The baseline for our experiments, denoted **LM**, is to simply rank documents by $p_d(q)$ — no cluster information is used. Details of our particular implementation are given in Section 5.

Selection methods. In this class of algorithms, the cluster-induced language models play a very small role once the set $\text{Facets}_q(d)$ is selected. In essence, the standard language-modeling approach (that is, ranking by $p_d(q)$) is invoked to rank (some of) the documents comprising the clusters in $\text{Facets}_q(d)$. This method of scoring is intended to serve as a precision-enhancing mechanism, downgrading documents that happen to be members of some $c \in \text{Facets}_q(d)$ by dint of similarity to d in respects not pertinent to q.

In the **basis-select** algorithm, the net effect of the definition given in Table 1 is that only the *basis* documents of the clusters in $\text{TopClusters}_q(m)$ are allowed to appear in the final output list. Thus, this algorithm uses the pooling of statistics from documents in $\text{Cohort}(d)$ simply to decide whether d is worth ranking; the rank itself is based solely on $p_d(q)$.

The **set-select** algorithm differs in that *all* the documents in the clusters in $\text{TopClusters}_q(m)$ may appear in the final output list — the "set" referred to in the name is the union of the clusters in $\text{TopClusters}_q(m)$. The idea is that any document in a "best" cluster, basis or not, is potentially relevant and should be ranked. Again, the ranking of the selected documents is by $p_d(q)$.[2]

Another natural variant of the same idea is that documents appearing in more than one cluster in $\text{TopClusters}_q(m)$ should get extra consideration, given that they appear in several (approximations of) facets thought to be of interest to the user. This idea gives rise to the **bag-select** algorithm, so named in reference to the incorporation of a document's multiplicity in the *bag* formed from the "multi-set union" of all the clusters in $\text{Facets}_q(d)$. First, each selected document d is assigned a score consisting of the product of its language-modeling score $p_d(q)$ and the number of "top" clusters it belongs to. The N top-scoring documents are then re-ranked via $p_d(q)$ and presented in the new sorted order.

Aspect-x methods. We now turn to algorithms making more explicit use of clusters as smoothing mechanisms. In particular, we study what we term "aspect-x" methods. Our choice of name is a reference to the work of Hofmann and Puzicha [9], which conceives of clusters as explanatory latent variables underlying the observed data. (The "x" stands for "extended"). In our setting, this idea translates to using $p_c(q)$ as a proxy for $p_d(q)$, where the degree of dependence on a particular $p_c(q)$ is based on the strength of association between d and c. The **aspect-x** algorithm measures this association by $p_c(d)$; the **uniform-aspect-x** algorithm assumes that every $d \in c$ has the same degree of association to c. In both cases, re-ranking by $p_d(q)$ is applied.

The scoring function we use for our aspect-x algorithm can be motivated by appealing to the probabilistic derivation of the aspect model [9], as follows. It is a fact that

$$p(q|d) = \sum_c p(q|d,c)p(c|d). \qquad (1)$$

The aspect model assumes that a query is conditionally independent of a document given a cluster (which is a way of using clusters to smooth individual-document statistics), in which case $p(q|d) = \sum_c p(q|c)p(c|d)$. If we further assume that $p(d)$ and $p(c)$ are constant, we can write $p(q|d) =$

[2] Because our implementation treats clusters and their component documents in a "fifo" manner, it deviates slightly from the template. Let N' be the number of documents in the $m-1$ highest-ranked clusters. Then, only the $N - N'$ documents in the m'th cluster that are closest, in the KL-divergence sense, to the cluster's basis are allowed into $\text{TopDocs}(N)$.

$\alpha \sum_c p(q|c)p(d|c)$, where α is a constant that doesn't affect ranking. Our aspect-x algorithm then arises by replacing the conditional probabilities with the corresponding language models and only summing over the clusters in $\text{Facets}_q(d)$. Constraining which clusters participate in the sum to those of relatively high rank is important: experiments indicate that using a large number of clusters could be detrimental. We note, however, that it appears difficult within the strictly probabilistic framework of the original aspect model to incorporate such a constraint: a particular cluster's rank depends on all the other clusters, but none of the terms in the basic aspect-model equation explicitly conditions on them.

A hybrid algorithm. The selection-only algorithms emphasize $p_d(q)$ in scoring a document d; in contrast, the aspect-x algorithms rely on $p_c(q)$. We created the **interpolation** algorithm to combine the advantages of these two approaches.

The algorithm can be derived by dropping the original aspect model's conditional independence assumption — namely, that $p(q|d,c) = p(q|c)$ — and instead setting $p(q|d,c)$ in Equation 1 to $\lambda p(q|d) + (1-\lambda)p(q|c)$, where λ indicates the degree of emphasis on individual-document information. If we do so, then via some algebra we get $p(q|d) = \lambda p(q|d) + (1-\lambda)\sum_c p(q|c)p(c|d)$. Finally, applying the same assumptions as described in our discussion of the aspect-x algorithm yields a score function that is the linear interpolation of the score of the standard LM approach and the score of the aspect-x algorithm. Note that no re-ranking step occurs; as we shall see, the interpolation algorithm's incorporation of document-specific information yields higher precision.

4. RELATED WORK

Document clustering has a long history in information retrieval [4, 16]; in particular, approximating topics via clusters is a recurring theme [?]. Arguably the work most related to ours by dint of employing both clustering and language modeling in the context of ad hoc retrieval[3] is that on latent-variable models, e.g., [9, 8, 12, 2], of which the classic aspect model is one instantiation. Such work takes a strictly probabilistic approach to the problems we have discussed with standard language modeling, as opposed to our algorithmic viewpoint. Also, a focus in the latent-variable work has been on sophisticated cluster induction, whereas we find that a very simple clustering scheme works rather well in practice. Interestingly, Hofmann [8] linearly interpolated his probabilistic model's score, which is based on (soft) clusters, with the usual cosine metric; this is quite close in spirit to what our interpolation algorithm does.

Implicit corpus structure is also exploited by Lafferty and Zhai's *expanded query language model* [11]. Their method uses interleaved document-term Markov chains (which can be thought of as tracing "paths" between related documents) to enhance language models built from queries. This is similar conceptually to our framework's use of inter-document similarities to enhance the performance of document language models, although in our work the notion of similarity is more explicit.

[3]See e.g., [3], [10], and [6] for applications of clustering in related areas.

5. EXPERIMENTAL SETUP

Data. We conducted our experiments on TREC data. We used titles (rather than full descriptions) as queries, resulting in an average length of 2-5 terms. Some characteristics of our three corpora are summarized in the following table.

corpus	# of docs	queries	previous work
AP89	84,678	1-46,48-50	Lafferty & Zhai [11]
AP88+89	164,597	101-150	Lavrenko & Croft [13]
LA+FR	187,526	401-450	-

The first two data corpora, AP89 and AP88+89, were chosen because they have served as data for previous research on state-of-the-art algorithms somewhat related to but considerably extending the basic LM approach. We used the same stemming and stopword-removal policies as in those previous experiments; hence, we applied the Porter stemmer to the AP89 collection (disk one), and we ran the Krovetz stemmer on AP88+89 and removed both INQUERY stopwords [1] and length-one tokens. LA+FR (disk 5 and 4, respectively), which is part of the TREC-8 corpus (we used TREC-8 ad hoc queries), was neither stemmed nor subjected to stopword removal. This corpus is more heterogeneous than the other two.

Induction of base language models. Unless otherwise specified, we use unigram Dirichlet-smoothed language models (which were previously shown to yield good performance for short queries [19]) in the following manner. For the purposes of this discussion, we use the term "document" and notation d to refer either to a true document in the corpus $\mathcal{C}$ or to a query. Let $f(x \in y)$ be the number of times word x occurs in item y. For a text sequence $\vec{w} = w_1 w_2 \cdots w_n$, the Dirichlet-smoothed language model induced from d assigns the following probability to $\vec{w}$:

$$p_d^{Dir}(\vec{w}) \stackrel{def}{=} \prod_{i=1}^{n} \frac{f(w_i \in d) + \mu \cdot p_{\mathcal{C}}^{ML}(w_i)}{\sum_w f(w \in d) + \mu},$$

where the free parameter μ controls the degree to which the document's statistics are altered by the overall corpus statistics, and "ML" indicates the maximum-likelihood estimate. Then, for any two documents d and d', we set $p_d(d')$ to

$$\exp\left(-D\left(p_{d'}^{ML}(\cdot)||p_d^{Dir}(\cdot)\right)\right)$$

(normalizing when appropriate), where D is the Kullback-Leibler divergence. This formulation is actually equivalent to a log-likelihood criterion under certain assumptions [11], but in practice is less sensitive than $p_d^{Dir}(d')$ to variations in the length of d'.

For a given cluster c, the corresponding language model $p_c(\cdot)$ is induced by concatenating c's component documents and then applying the document-LM induction method to the new "document".

Reference comparisons. While one of our goals is to demonstrate that incorporating corpus structure as in our retrieval framework can provide improvements over the performance of the standard LM algorithm, we also wish to detemine whether our algorithms are competitive with state-of-the-art language-modeling-based algorithms. One natural choice for

	Baseline: LM	basis-S	set-S	bag-S	uniform	aspect-x	interp.	Pseudo-feedback Markov chains
Avg. Prec.	21.03%	_22.1%_*	_22.45%_*	_22.3%_*	_21.7%_	_22.6%_*	**24.9%***	_23.2%_
Prec. at 0	57.4%	**58.5%**	**58.5%**	_58.1%_	57.2%	_58.2%_	55.8%	53.4%
Recall	48.67%	_54.86%_	_56.15%_	_62.77%_*	_57.31%_	_62.16%_*	**63.62%***	_61.91%_

Table 2: AP89 results (3261 relevant documents). Cluster size $k = 40$; interpolation parameter $\lambda = 0.4$.

	Baseline: LM	basis-S	set-S	bag-S	uniform	aspect-x	interp.	Relevance model
Avg. Prec.	24.37%	_26.58%_*	_28.11%_*	_26.65%_*	_24.92%_	_27.5%_*	**31.28%***	_26.17%_
Prec. at 0	65.52%	_65.77%_	65.32%	65.17%	**67.5%**	_65.9%_	_65.82%_	61.61%
Recall	66.53%	_71.86%_	_76.48%_*	_79.23%_*	_69.05%_	_78.9%_*	**80.83%***	_77.69%_

Table 3: AP88+89 results (4805 relevant documents). Cluster size $k = 40$; interpolation parameter $\lambda = 0.6$.

comparison is Lafferty and Zhai's *pseudo-feedback Markov chains* algorithm [11], which extends the expanded query language model described above by forcing the chains to pass through top-ranked documents, as determined using the standard LM approach. Another obvious candidate is Lavrenko and Croft's *relevance model* [13] which was the first method to explicitly incorporate relevance into the language-modeling framework, and which demonstrated excellent performance. Note that both algorithms, in contrast to our framework, depend on pseudo-feedback mechanisms to cope with the lack of true user feedback.

Implementation. We used the Lemur toolkit [14] to run our experiments. Our implementations of the baseline used optimized smoothing-parameter settings with respect to average non-interpolated precision[4], computed via line search. For our novel algorithms, we optimized the cluster-size parameter k and the interpolation algorithm's interpolation parameter λ, but the other parameters were set to default values suggested in the previous literature [19]; thus, the baseline algorithm was given an extra advantage.

Rather than re-implement the pseudo-feedback Markov-chain and relevance-model algorithms described above, we report results presented in the previous literature [11, 13]. We do realize that minor differences in performance could stem from specific implementation issues, but as stated above, our goal was to test the competitiveness of our algorithms' performance with respect to that of other prominent algorithms, not to prove our algorithms' superiority.

6. EXPERIMENTAL RESULTS

For our evaluation measures, we used average non-interpolated precision, interpolated precision at 0, and recall, all for $N = 1000$ selected documents. Our main experimental results are given by Tables 2, 3, and 4 and Figure 2.

In the tables, for each evaluation metric, the strongest performance is boldfaced and all results above the baseline (LM) are italicized. Also, the Wilcoxon two-sided test was employed with significance threshold $p = 0.05$ — all statistically significant performance improvements and degradations for our algorithms relative to the baseline are marked with a star (*).

Clearly, at the indicated settings (given in the captions), even at worst our algorithms are always competitive with the baseline LM approach, and with occasional exceptions (mostly for precision at 0) generally do better. We also observe that the aspect-x and interpolation algorithms are competitive with the pseudo-feedback Markov-chains algorithm (see Table 2) and the relevance-model algorithm (see Table 3) with respect to all performance measures.

Figure 2 shows 11-point precision/recall curves for our algorithms and the baseline. In all three corpora, the interpolation algorithm does best overall. On AP88 and AP88+89, our cluster-based algorithms on the whole generally perform demonstrably better than the baseline. In LA+FR, however, the new algorithms, with the exception of the interpolation algorithm, seem difficult to distinguish from LM, as is borne out by the relative lack of statistical-significance indications in Table 4.

The fact that the aspect-x algorithm was usually superior to uniform-aspect-x indicates that incorporating within-cluster structure, as represented by $p_c(d)$, is important.

Finally, the generally high performance of our aspect-x and interpolation algorithms seems to support our claims as to the importance of using corpus-structural information in the particular ways we have suggested: specifically, in these two algorithms, clusters play both a selection and a smoothing role, and both document-specific information and intra-cluster structure are incorporated as well.

In what follows, we discuss the results of further experimental studies. For space reasons, we present only a subset of the performance figures for a selection of corpora.

Parameter selection. The cluster-size parameter k does have a noticeable impact on performance. A series of preliminary experiments (whose results are omitted due to space restrictions) indicate that small values of k (e.g., 5 or 10) yield better results than the baseline LM for all but the uniform-aspect-x method, demonstrating the usefulness of even tiny document clusters. However, increasing k to 40 resulted in superior performance on the AP89 and AP88+89 datasets, which suggests that the re-rank step of our algorithm template can compensate to a degree for the extra irrelevant documents that large clusters may bring into consideration.

We must also choose m, the number of clusters to be retrieved, recalling that we wish to return a fixed number $N = 1000$ of documents. In the experimental results reported above, two different schemes were used. For the al-

[4]Optimization with respect to recall yielded results which were statistically indistinguishable with respect to each of our performance metrics.

	Baseline: LM	basis-S	set-S	bag-S	uniform	aspect-x	interp.
Avg. Prec.	22.16%	21.92%	*22.52%*	22%	21.73%	*22.45%*	**23.88%***
Prec. at 0	57.37%	*57.91%*	*57.89%*	*58.16%*	57.28%	**58.25%**	*57.81%*
Recall	48.31%	*55.64%*	*58.09%*	*53.34%*	*58.09%*	**63.7%***	*61.18%**

Table 4: Results for LA+FR (1391 relevant documents). Cluster size $k = 10$; interpolation parameter $\lambda = 0.8$.

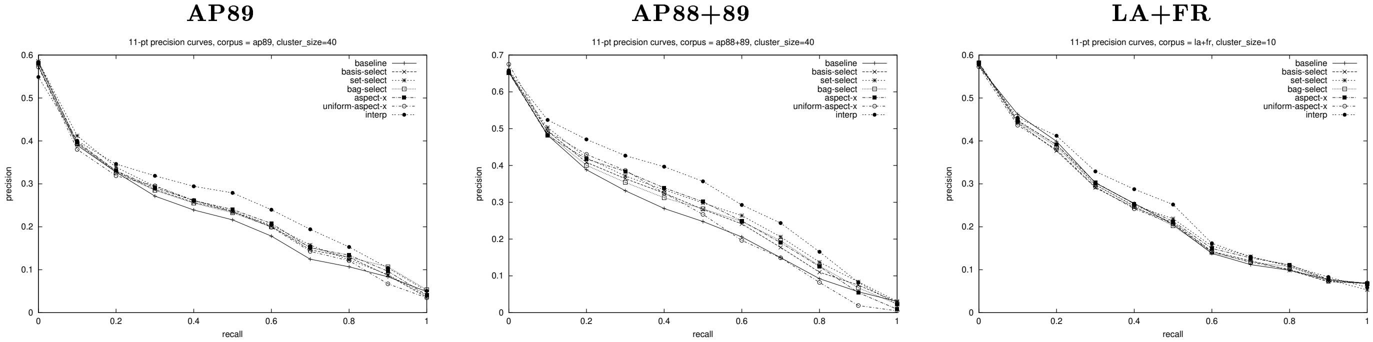

Figure 2: 11-point precision/recall curves. For AP89 and AP88+89, $k = 40$; for LA+FR, $k = 10$.

gorithms using clusters solely for selection, we set m to either 1000 or the minimum value needed for there to be 1000 documents receiving a non-zero score.[5] For the remaining algorithms (aspect-x, uniform-aspect-x, and interpolation), we set $m = 10000$. The former group of algorithms were more sensitive to choice of m than the latter, where as long as m did not exceed 10000, satisfactory improvements with respect to the baseline algorithms were observed. However, drawing upon more clusters than this — which in a sense is what the classic aspect model [9] does — was clearly detrimental for some of the data corpora.

An important regard in which the interpolation algorithm differs from the other methods we have introduced is in its inclusion of an additional free parameter λ, representing the degree of dependence on $p_d(q)$ relative to the aspect-x algorithm. Figure 3 plots the "trajectories" of the interpolation algorithm through performance space as λ is increased. This figure makes visually clear the interplay between cluster and document information: small λ's (emphasizing clusters) result in better recall but relatively poor precision; but large λ's (emphasizing documents) improve precision at the expense of recall. The performance of "average" values (around .6) shows that integrating document- and cluster-level information provides better performance than either can produce alone.

We note that the aspect-x algorithm can be viewed as a version of the interpolation algorithm in which $\lambda = 0$ and re-ranking is added to improve average precision. In return for some performance degradation relative to the interpolation algorithm, it offers the advantage of having one fewer parameter to tune, and is fairly robust to m's value as well.

The re-rank step. How important is the re-ranking step, in which the top-ranked documents are re-scored by their document-specific language models, to producing good precision? We ran several experiments to explore this issue.

First, we observed considerable degradation in average precision if we removed the $p_d(q)$ term from the score functions of the basis-select and set-select algorithms, for which re-ranking is redundant. Note that this version of the basis-select algorithm corresponds to applying the basic LM approach to the "document" Cohort(d) rather than d itself, and so can be thought of as a smoothing method wherein the document language model is created by backing off completely to a cluster language model.

[5] In the bag-select algorithm we chose $m = 1000$, although lower values would have sufficed.

	AP89		AP88+89		LA+FR	
re-rank?	yes	no	yes	no	yes	no
Avg. Prec.	22.6%	19.8%	27.5%	26.95%	22.45%	16.01%
Prec. at 0	58.2%	46.98%	65.9%	65.21%	58.25%	46.92%

Table 5: Effect of re-rank step on aspect-x precision. For AP89 and AP88+89, k=40; for LA+FR, k=10.

Next, we examined the role of the optional re-ranking step in the algorithms that explicitly incorporate it. When the aspect-x and uniform-aspect-x algorithms — the two cases in which the scoring function does not incorporate $p_d(q)$ — were run without the optional re-ranking phase, low average precision and precision at 0 resulted, implying that reliance on $p_c(\cdot)$ alone suffers from over-regularization; the results for the aspect-x algorithm are shown in Table 5. Furthermore, re-ranking is also required to achieve reasonable precision for the bag-select algorithm, even though its scoring function incorporates $p_d(q)$: when re-ranking is not applied, average precision suffers when clusters are small.

In the case of the interpolation algorithm, however, the additional re-rank phase is not needed as long as the interpolation weight λ for the document-based language model is large enough. This can be seen in Figure 4, where the difference between average precision without and with re-rank at different values of λ is reported — observe that for $\lambda > 0.4$, re-ranking degrades performance.

These results suggest that (1) for best results, it is important to strike the right balance between document-specific and inter-document information, and (2) for some algorithms, re-ranking creates this balance, but in others it can upset it.

AP89 **AP88+89**

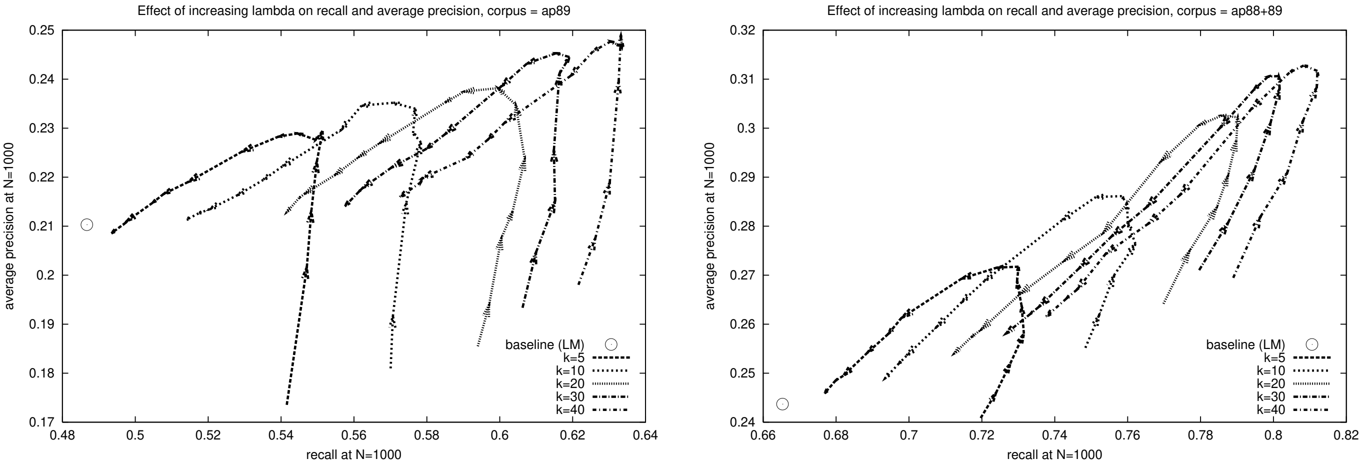

Figure 3: Interpolation algorithm's recall vs average precision as λ grows (increments of .1 until .9, then .925, .95, .975, .98, .99). Recall that $\lambda = 1$ would yield the baseline language-model scoring function. Similar patterns were observed on the LA+FR corpus; we omit the results for clarity.

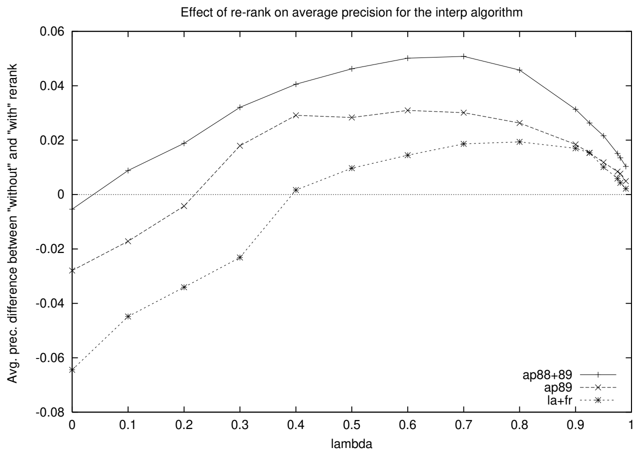

Figure 4: Effect of re-rank step on average precision for the interpolation algorithm as λ varies. For AP89 and AP88+89 k=40; for LA+FR k=10.

Smoothing. The sensitivity of the LM approach to choice of smoothing technique and smoothing parameters has prompted a great deal of research [19, 7, 17]. However, we found that for our algorithms, simply setting the Dirichlet smoothing parameter μ to a suggested value of 2000 [19] (or, as it turned out, randomly-chosen values within the neighborhood of 2000) outperformed the μ-optimized baseline. Moreover, experiments with Jelinek-Mercer and absolute discounting — two other well-known single-parameter smoothing methods [19] — yielded the same outcome of relative insensitivity to choice of parameter value for the underlying smoothing method employed.

Feature selection. Another interesting observation is that effective incorporation of cluster information somewhat obviates the need for feature selection. In particular, Table 6 shows one case where using the aspect-x and interpolation algorithms *without* a stemmer or stop-word list outperforms the baseline *with* access to the Porter stemmer with respect to average precision and recall. On the other hand, stemming led to degradation of precision at zero. Results for cluster sizes other than 40 and different corpora were consistent with these findings.

Is it all due to language modeling? Throughout this paper, we have used language models as our information representation. An interesting question is whether it is the representation (e.g., $p_c(\cdot)$), or the *source* of this representation (e.g. c itself) that matters most. We therefore explored the effect of using an alternative representation. Specifically, both the queries and the documents were represented using log-based tf.idf, with the inner product as distance measure. As before, clusters were treated as large documents formed by concatenating their contents. Altering our selection algorithms (basis-select, set-select, and bag-select) in this way led to improved performance with respect to the basic tf.idf retrieval algorithm, as shown in Table 7. On the other hand, these algorithms did not do as well as their original, LM-based counterparts. We thus see that our algorithmic framework can boost performance for other information representations over the structure-blind alternative, but language models do seem to have advantages, at least in comparison to tf.idf.

7. CONCLUSIONS

In summary, we have proposed a general framework that enables the development of a variety of algorithms for integrating corpus similarity structure, modeled via clusters, and document-specific information. Although our proposal is motivated by the recent language-modeling approach to information retrieval, and the specific algorithms presented here do use language models for representation purposes to good effect, we observed that the framework also can be used with basic classic IR techniques such as tf.idf.

An interesting direction for future work is to explore the effect of using alternative clustering algorithms. We would

	S-Baseline	U-Baseline	S-aspect-x	U-aspect-x	S-interpolation ($\lambda = 0.4$)	U-interpolation ($\lambda = 0.6$)
Avg. Prec.	21.03%	19.56%	*22.6%**	*21.51%**	**24.9%***	*24.08%**
Prec. at 0	57.4%	60.1%	*58.2%*	**60.9%**	55.8%	58.9%
Recall	48.67%	44.99%	*62.16%**	*60.47%**	**63.62%***	*60.66%**

Table 6: Stemming comparison on AP89. S-: stemmed version; U-: un-stemmed version. Cluster size $k = 40$. Significant differences are reported with respect to the corresponding baseline.

	tf.idf version				LM version			
	Baseline	basis-select	set-select	bag-select	Baseline	basis-select	set-select	bag-select
Avg. Prec.	16.43%	*16.67%*	**17.18%***	*16.68%**	22.16%	21.92%	**22.52%**	22%
Prec. at 0	46.66%	*46.94%*	**47.29%**	*46.92%*	57.37%	*57.91%*	*57.89%*	**58.16%**
Recall	47.45%	*55.07%*	**57.58%**	*51.98%*	48.31%	*55.64%*	**58.09%**	*53.34%*

Table 7: Simple similarity metric based on tf.idf vs. LM-based similarity on LA+FR. Cluster size $k = 10$.

also like to study the role that overlapping plays in our framework: is most of the performance gain due to the (high) degree of overlap in our clusters or to the way structure and individual-document information are integrated? Another interesting direction is to examine whether other algorithms, such as the LM-based pseudo-feedback methods we used for reference comparisons [11, 13], can benefit if we replace the basic LM retrieval algorithm they employ with one of ours.

Most importantly, we would like to develop a principled probabilistic interpretation of the framework we have proposed. We have done some preliminary work based on considering the factorization $p(q|d) = \sum_c p(q|d,c)p(c|d)$; some of the components of our scoring functions can be considered to be (very rough) approximations of the terms in this factorization. Creating a rigorous probabilistic foundation for the work described here is one of our main future goals.

Acknowledgments We thank Eric Breck, Claire Cardie, Shimon Edelman, Thorsten Joachims, Art Munson, Bo Pang, Ves Stoyanov, and the anonymous reviewers for valuable comments. Thanks to ChengXiang Zhai and Victor Lavrenko for answering questions about their work, and Andrés Corrada-Emmanuel for responding to queries about Lemur. This paper is based upon work supported in part by the National Science Foundation under grants ITR/IM IIS-0081334 and IIS-0329064 and by an Alfred P. Sloan Research Fellowship. Any opinions, findings, and conclusions or recommendations expressed above are those of the authors and do not necessarily reflect the views of the National Science Foundation or Sloan Foundation.

8. REFERENCES

[1] James Allan, M.E. Connel, W. Bruce Croft, Fang-Fang Feng, D. Fisher, and X. Li. Inquery and trec-9. In *Proceedings of the Ninth Text Retrieval Conference (TREC-9)*, pages 551–562, 2001.

[2] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[3] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

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

[5] W. Bruce Croft and John Lafferty, editors. *Language Modeling for Information Retrieval*. Kluwer, 2003.

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

[7] Djoerd Hiemstra. Term-specific smoothing for the language modeling approach to information retrieval: The importance of a query term. In *Proceedings of SIGIR*, 2002.

[8] Thomas Hofmann. Unsupervised learning by probabilistic latent semantic analysis. *Machine Learning*, 42(1-2):177–196, 2001.

[9] Thomas Hofmann and Jan Puzicha. Unsupervised learning from dyadic data. Technical Report TR-98-042, International Computer Science Institute (ICSI), 1998.

[10] Rukmini Iyer and Mari Ostendorf. Modeling long distance dependence in language: Topic mixtures vs. dynamic cache models. *IEEE Transactions on Speech and Audio Processing*, 7(1):30–39, 1999.

[11] 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.

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

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

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

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

[16] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, 2nd edition, 1979.

[17] Hugo Zaragoza, Djoerd Hiemstra, and Michael Tipping. Bayesian extension to the language model for ad hoc information retrieval. In *Proceedings of SIGIR*, pages 4–9, 2003.

[18] ChengXiang Zhai and John Lafferty. Two-stage language models for information retrieval. In *Proceedings of SIGIR*, pages 49–56, 2002.

[19] 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.