ReQ-ReC: High Recall Retrieval with Query Pooling and Interactive Classification

Cheng Li[*,1], Yue Wang[*,2], Paul Resnick[1], Qiaozhu Mei[1,2]
[1]School of Information, University of Michigan, Ann Arbor, MI, USA
[2]Department of EECS, University of Michigan, Ann Arbor, MI, USA
{lichengz, raywang, presnick, qmei}@umich.edu

ABSTRACT

We consider a scenario where a searcher requires both high precision and high recall from an interactive retrieval process. Such scenarios are very common in real life, exemplified by medical search, legal search, market research, and literature review. When access to the entire data set is available, an active learning loop could be used to ask for additional relevance feedback labels in order to refine a classifier. When data is accessed via search services, however, only limited subsets of the corpus can be considered—subsets defined by queries. In that setting, relevance feedback [17] has been used in a query enhancement loop that updates a query.

We describe and demonstrate the effectiveness of ReQ-ReC (ReQuery-ReClassify), a double-loop retrieval system that combines iterative expansion of a query set with iterative refinements of a classifier. This permits a separation of concerns: the query selector's job is to enhance recall, while the classifier's job is to maximize precision on the items that have been retrieved by any of the queries so far. The overall process alternates between the query enhancement loop (to increase recall) and the classifier refinement loop (to increase precision). The separation allows the query enhancement process to explore larger parts of the query space. Our experiments show that this distribution of work significantly outperforms previous relevance feedback methods that rely on a single ranking function to balance precision and recall.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Relevance Feedback

General Terms

Algorithms, Experimentation

Keywords

Relevance Feedback; Query Expansion; Active Learning

*Cheng Li and Yue Wang contributed equally to this work.

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

http://dx.doi.org/10.1145/2600428.2609618.

1. INTRODUCTION

We are witnessing an explosive growth of text data in many fields, including millions of scientific papers, billions of electronic health records, hundreds of billions of microblog posts, and trillions of Web pages. Such a large scale has created an unprecedented challenge for practitioners to collect information relevant to their daily tasks. Instead of keeping local collections of data related to these tasks, many users rely on centralized search services to retrieve relevant information. These services, such as Web search engines (e.g., Google), literature retrieval systems (e.g., PubMed), or microblog search services (e.g., Twitter search API, Topsy) typically return a limited number of documents that are the most relevant to a user-issued query. These existing retrieval systems are designed to maximize the precision of top-ranked documents; they are good at finding "something relevant," but not necessarily everything that is relevant.

We focus on scenarios where a user requires a high recall of relevant results in addition to high precision. Such scenarios are not uncommon in real life, exemplified by social search, medical search, legal search, market research, and literature review. For example: a social analyst needs to identify all the different posts in which a rumor spreads in order to reconstruct the diffusion process and measure the influence of the rumor; a physician needs to review all the patients that satisfy certain conditions to select cohorts for clinical trials; an attorney needs to find every piece of evidence related to her case from documents that are under legal hold; a scientist does not want to miss any piece of prior work that is related to his ongoing research. We denote all these tasks generically as "high-recall" retrieval tasks.

Finding a needle in a haystack is hard; finding all the needles in a haystack is much harder. Existing retrieval systems do not naturally meet this type of information need. To conduct a comprehensive literature review using a search engine, we have to submit many alternative queries and examine all the results returned by each query. Such a process requires tremendous effort of the user to both construct variations of queries and examine the documents returned.

This high-precision and high-recall task becomes substantially harder as the collection grows large, making it impossible for the user to examine and label all the documents in the collection, and impractical even to label all the documents retrieved by many alternative queries. In some contexts such as e-discovery, a computer-assisted review process has been used that utilizes machine learning techniques to help the user examine the documents. Such a process typically casts high-recall retrieval as a binary classification task. At the

beginning, the user is required to label a small sample of documents. A classifier trained using these labeled documents then takes over and predicts labels for other documents in the collection. An active learning loop can be used to ask for additional relevance labels in order to refine the classifier. These methods, however, require that the user has access to the full collection of documents and that it is feasible to execute her classifier on all the documents.

In other scenarios, the users either do not own the collection or it is too large, so they can only access documents in the collection through an external search service. This makes it unrealistic to either examine or classify the entire collection of documents. Instead, only limited subsets of the document corpus can be considered, subsets defined by queries.

Existing retrieval systems are not tuned for high-recall retrieval on the condition of limited access to the data via search services. In most cases, a system only aims to maximize the precision in the documents that are retrieved by the current query. Relevance feedback has been used in a query enhancement loop that updates a query. Many search engines provide services to collect explicit and/or implicit feedback from the users or to suggest alternative queries to the users. These practices typically generate a new query that replaces the old one, which is expected to improve both precision and recall. Once a new query is issued, the results retrieved by the old queries are forgotten, unless they are manually harvested by the user.

We study a novel framework of retrieval techniques that is particularly useful for high-recall retrieval. The new framework features a ReQ-ReC (ReQuery-ReClassify) process, a double-loop retrieval system that combines iterative expansion of a query set with iterative refinements of a classifier. This permits a separation of concerns, where the query generator's job is to enhance recall while the classifier's job is to maximize precision on the items that have been retrieved by *any* of the queries so far. The overall process alternates between the query expansion loop (to increase recall) and the classifier refinement loop (to increase precision). The separation of the two roles allows the query enhancement process to be more aggressive in exploring new parts of the document space: it can explore a non-overlapping portion of the corpus without worrying about losing the veins of good documents it had found with previous queries; it can also use queries that have lower precision because the classifier will weed out the misses in a later stage. Our experiments show that this distribution of work significantly outperforms previous relevance feedback methods that rely on a single ranking function to balance precision and recall. The new framework also introduces many opportunties to investigate more effective classifiers, query generators, and human-computer interactive algorithms for labeling subsets, and especially to investigate what combinations work best together.

Unlike Web search engines that target users who have real-time, ad hoc information needs, the ReQ-ReC process targets users who care about the completeness of results and who are willing to spend effort to interact with the system iteratively and judge many (but not all) retrieved documents. The process has considerable potential in applications like social media analysis, scientific literature review, e-discovery, patent search, medical record search, and market investigation, where such users can be commonly found.

The rest of the paper is organized as follows. We discuss related work in Section 2. Section 3 gives an overview of the ReQ-ReC double-loop framework and its key components. Section 4 describes several instantiations of the framework. Section 5 provides a systematic evaluation of the proposed methods. Finally, we conclude the paper in Section 6.

2. RELATED WORK

The ReQuery-ReClassify framework integrates and extends two well-established "human-in-the-loop" mechanisms: relevance feedback in information retrieval, and active learning in text classification.

Relevance feedback was shown long ago to be effective for improving retrieval performance [17]. In a feedback procedure, the retrieval system presents the top-ranked documents to the user and collects back either explicit judgments of these documents or implicit feedback implied by certain actions of the user [9, 19]. The system then learns from the collected feedback and updates the query. The new query reflects a refined understanding of the user's information need [15, 28], which improves both precision and recall in the next round of retrieval. Even without real user judgments, retrieval performance may still benefit from simply treating the top-ranked documents as relevant, which is known as a process of pseudo relevance-feedback [1].

In a search session, relevance feedback can be executed for multiple rounds. Harman [8] studied multiple iterations of relevance feedback, and found that retrieval performance is greatly improved by the first two to three iterations, after which the improvements became marginal. Multiple iterations of relevance feedback have received more attention in content-based image retrieval [3, 16, 30].

In complicated search tasks, the user is often involved in a search session consisting of a series of queries, clickthroughs, and navigation actions. *Session*-based retrieval aims at learning from these signals in order to better understand the user's information need, thus improving the relevance of results when the user issues the next query [19, 14]. Instead of improving the performance of the next query, ReQ-ReC aims to maximize the recall of the results collectively retrieved by all the queries in the search session.

Like traditional iterative relevance feedback, the ReQ-ReC process also adopts multiple iterations of user interaction. Indeed, as shown in Section 3, iterative relevance feedback is a special case instantiation of the ReQ-ReC framework. Instead of replacing the old query with a new query, however, ReQ-ReC can accumulate documents retrieved by any of the queries issued so far. By doing this, rather than optimizing both precision and recall through the choice of a single query, we place the burden of maximizing precision on a classifier, and new queries can be dedicated to improving only recall.

When it is feasible to process the entire collection of documents, the problem of high-recall retrieval can be cast as a binary classification problem where the positive class captures documents that are relevant to the information need and the negative class captures the rest. The practice of relevance feedback essentially becomes an active learning process, in which the system iteratively accumulates training examples by selecting documents and asking the user for labels [18]. This strategy is commonly used in computer-assisted reviews for e-discovery, often referred to as the process of 'predictive coding' [13]. Different active learning algorithms

use specific strategies for selecting the documents to label, many of which attempt to maximize the learning rate of a 'base' classifier with limited supervision [18]. For text classification, a popular choice of such a 'base' classifier is the support vector machine (SVM) [2]. Using SVM, a variety of document selection strategies have been explored. Tong and Koller [23] proposed to select documents closest to the decision hyperplane in order to rapidly shrink the version space and reduce model uncertainty. In contrast, Drucker et al. [4] selected documents with highest decision function values to avoid probing the user with too many non-relevant documents. Xu et al. [27] mixed these two strategies and achieved better retrieval performance.

Like active learning, the ReQ-ReC process also trains a binary classifier. The major difference is that ReQ-ReC does not require knowledge about the entire document collection and thus does not classify *all* documents. Instead, it starts from a limited subset defined by the original query and actively expands the space. This is a huge gain, as text classification and active learning are usually computationally prohibitive for modern IR collections containing a large number of documents. Indeed, previous studies that apply active learning to retrieval can only evaluate their approaches using moderate-scale collections (such as the 11,000-documents Reuters collections used in [4] and [27]), or only focus on the documents retrieved by one query (top 100 documents in [26] and top 200 in [22]). Given its big advantage in efficiency, the ReQ-ReC process could potentially provide a new treatment for active learning, especially when the data collection is large and the positive class is very rare.

The idea of active learning has also been applied to relevance feedback for retrieval. Shen and Zhai [20] studied active feedback, where the system *actively* selects documents and probes the user for feedback instead of passively presenting the top ranked documents. It is shown that selecting diverse top-ranked documents for labeling is desirable, since it avoids asking for labels on similar documents and thus accelerates learning. Xu et al. [26] improved this heuristic by jointly considering relevance, diversity, and density in selected documents. Both techniques exploit density information among top-ranked documents, and select representative ones for feedback. Recently, Tian and Lease [22] combined uncertainty sampling (*Simple Margin*) and density-based sampling (*Local Structure*) in iterative relevance feedback to minimize user effort in seeking several to many relevant documents. The difference between our work and theirs is articulated by the difference between the ReQ-ReC process and relevance feedback described above: the addition of a classifier and use of results from all queries allows more aggressive exploration of alternative queries.

3. THE REQ-REC FRAMEWORK

In this section, we introduce the general ReQuery-ReClassify (ReQ-ReC) framework, including its key components. Specific instantiations of the framework will be discussed in the next section. The basic idea of the framework is to distribute the burden of maximizing both the precision and recall to a *set* of queries and a classifier, where the queries are responsible for increasing the recall of relevant documents retrieved and the classifier is responsible for maximizing the precision of documents retrieved collectively by all of the queries in the set. The framework features a double-loop mechanism: the inner-loop classifies the retrieved documents, actively collects user feedback, and improves the classifier (ReClassify); the outer-loop generates new queries (ReQuery), issues API calls, and iteratively adds newly retrieved documents into the workset. In the rest of the paper, we refer to the framework as "ReQ-ReC" or "double-loop" interchangeably.

3.1 The Double Loop Process

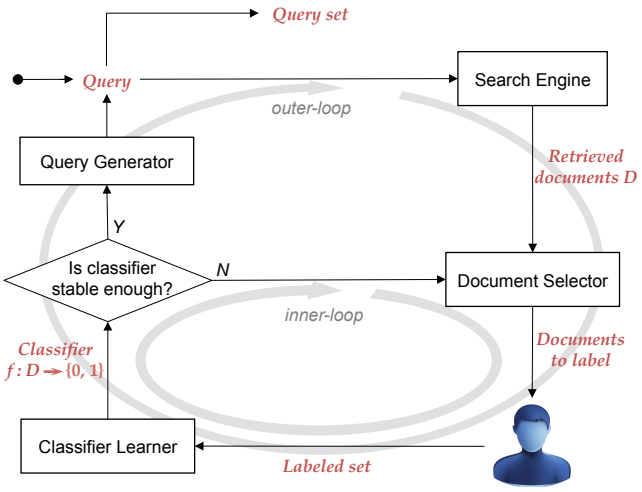
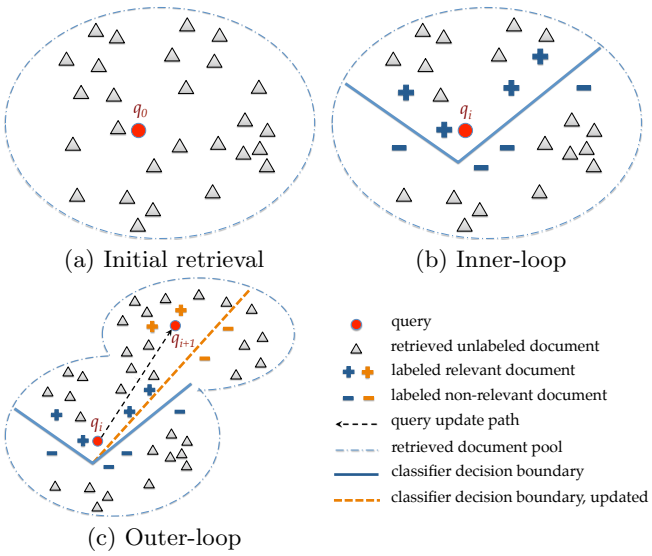


Figure 1: ReQ-ReC framework

The ReQ-ReC framework can be viewed as a double-loop review process, as illustrated in Figure 1. The process maintains a set of queries, a pool of retrieved documents, and a binary classifier. With an initial query composed by the user, the system retrieves an initial set of documents using a search service. An inner-loop starts from there, in which the system iteratively presents a small number of documents (e.g., 10) selected from the current pool of retrieved documents to the user and asks her to label them as either relevant or not. The classifier is consequently updated based on the accumulated judgments of the user, which is then used to *reclassify* the pool of documents. After a few iterations of the inner-loop, the the classifier's predictions stabilize. At this point, the inner-loop will suspend. The system then proposes to add a new query to the query set, aiming to retrieve more relevant documents from the collection. Upon the approval—and possible edits—of the user, the system will retrieve a new set of documents using the new query, and *merge* them into the pool of retrieved documents. The *requery* process makes up one iteration of the outer-loop of the framework. After new documents are retrieved and added into the pool, the system starts a new inner-loop and continues to update the classifier left from the last iteration. The whole review process will end when no more relevant documents can be retrieved by a new query or when the user is satisfied.

Another way to look at the framework is to imagine a search process in the information space (e.g. a vector space of documents and queries), as illustrated in Figure 2. The system interacts with the user as it navigates through the information space, aiming to delineate a manifold that contains as many relevant documents and as few non-relevant documents as possible. Each query can only reveal a small region of the information space that surrounds it. The "first guess" on such a manifold is, of course, the region surround-



(a) Initial retrieval (b) Inner-loop

(c) Outer-loop

Figure 2: A double-loop process of search in the information space. (a) Each query only retrieves its surrounding region under inspection. (b) The inner-loop updates a classifier that refines the boundary between relevant and non-relevant documents. (c) The outer-loop expands the subspace which includes more relevant documents.

ing the initial query. A classifier clarifies the boundary of the manifold (to maximize precision), which is iteratively refined with newly labeled data points selected from the revealed regions. To explore other regions in the space so as to expand the relevant manifold (to maximize recall), the system will estimate a promising direction and will make a new query to move in that direction into the uncharted space. This new region and all previously unveiled regions are combined as the current search space, in which the system continues to refine the boundary of the relevant manifold. The search process will end if the relevant manifold stops expanding, or if the user decides to terminate early.

From this perspective, each query contributes a new region to the search space without giving up any already discovered regions. Such a pure "expansion" of the search space will include many non-relevant documents, but the classifier is able to filter the non-relevant documents at the end and recover the true boundary of the relevant manifold. By contrast, in a relevance feedback procedure, every new query will "redefine" the search space as the region surrounding the new query. Given a good query, this region indeed contains fewer non-relevant documents than our "expanded" search space (i.e., achieves a higher precision), but it is also likely to contain fewer new relevant documents. In relevance feedback, the challenge is to find a new query that both retrieves the relevant documents from the old query and also retrieves new ones. In ReQ-ReC, the challenge is simply to find a query that retrieves new relevant documents.

3.2 Anatomy of the ReQ-ReC Framework

Given the high-level intuitions of the ReQ-ReC framework, we now discuss the key components in the double-loop. To facilitate the discussion, we introduce the notations in Table 1 and summarize the framework in Algorithm 1.

Table 1: Notations of the double-loop process

$\mathcal{D}$	index of the document collection
q_i	the i-th query submitted
$\mathcal{D}_q$	the union of all unjudged documents retrieved by the set of queries $\{q_i\}$
$\mathcal{D}_s$	documents selected for user judgments
$\mathcal{D}_l$	set of documents labeled already
$retrieve(\mathcal{D}, q_i)$	a retrieval function that returns a subset of documents from index $\mathcal{D}$ by query q_i
Θ_A	model for document selection
Θ_R	model for relevant/non-rel classification
$train_A(\mathcal{D}_q, \mathcal{D}_l)$	function to train/update Θ_A using labeled and unlabeled documents
$train_R(\mathcal{D}_q, \mathcal{D}_l)$	function to train/update Θ_R using labeled and unlabeled documents
$selectK(\mathcal{D}_q, \Theta_A)$	function to select K documents using the document selection model
$label(\mathcal{D}_s)$	function to obtain relevance labels of $\mathcal{D}_s$
$predict(\mathcal{D}_q, \Theta_R)$	function to predict the relevance labels and rank unlabeled documents
$query(\{q_i\}, \cdot)$	function to generate a new query

3.2.1 Search

The ReQ-ReC framework assumes neither ownership nor full access to the document collection, but instead relies on a standard search service to retrieve documents from the index. The retrieval service's ranking function can use any reasonable retrieval model that takes the input of a query q_i and outputs a certain number of ranked documents from the index (e.g., using a vector space model, a language modeling approach, or a boolean retrieval model). In most cases, the user has no knowledge about the algorithm that is employed by the external search service. In that case, the retrieval function is treated as a black box in the framework.

After each search process the retrieved documents will be merged into the pool of unlabeled documents $\mathcal{D}_q$, which expands the workset for document selection and classification.

Algorithm 1 The double-loop process

Input: Initial query q_0, index of document collection $\mathcal{D}$
Output: A set of labeled documents $\mathcal{D}_l$ and a set of unjudged documents in $\mathcal{D}_q$ with system predicted labels.

$\mathcal{D}_q \leftarrow \emptyset$
$\mathcal{D}_l \leftarrow \emptyset$
repeat // outer loop
  $\mathcal{D}_q \leftarrow retrieve(\mathcal{D}, q_i) \cup \mathcal{D}_q$
  repeat // inner loop
    if $\mathcal{D}_l == \emptyset$ **then**
      $\mathcal{D}_s \leftarrow selectK(\mathcal{D}_q)$
    else
      $\Theta_A \leftarrow train_A(\mathcal{D}_q, \mathcal{D}_l)$
      $\mathcal{D}_s \leftarrow selectK(\Theta_A, \mathcal{D}_q)$
    end if
    $\mathcal{D}_l \leftarrow \mathcal{D}_l \cup label(\mathcal{D}_s)$
    $\mathcal{D}_q \leftarrow \mathcal{D}_q - \mathcal{D}_s$
    $\Theta_R \leftarrow train_R(\mathcal{D}_q, \mathcal{D}_l)$
    $predict(\Theta_R, \mathcal{D}_q)$
  until meet stopping criteria for inner loop
  $q_{i+1} \leftarrow query(\{q_i\}, \mathcal{D}_q, \mathcal{D}_l, \Theta_A, \Theta_R)$
until meet stop criteria for outer loop

3.2.2 Document Selection

In every iteration of the inner-loop, during steps 6-10 of the algorithm the system selects K (e.g., 10) documents $\mathcal{D}_s$ from the pool of retrieved documents that are yet unlabeled, $\mathcal{D}_q$, and asks the user for judgments. At the beginning of the double-loop process, where there are no judged documents, this process can simply return the top documents ranked by the retrieval function, select a more diverse set of documents through an unsupervised approach, or even randomly sample from $\mathcal{D}_q$. Once labeled documents have been accumulated, the process is able to select documents based on an active learning strategy. Such a process aims to maximize the learning rate of the classifier and thus reduce the user's effort on labeling documents.

3.2.3 Classification

Given an accumulated set of labeled documents, the classification component learns or updates a binary classifier (i.e., Θ_R) at step 14 and reclassifies documents from $\mathcal{D}_q$ at step 15. Any reasonable classifier can be applied here.

In many high-recall retrieval tasks such as medical record search, it is important to find all patients that "match" certain conditions, but it is not necessary to rank the records identified as relevant [7]. In those cases, the labels of documents in $\mathcal{D}_q$ can be directly predicted by the classifier. In cases where ranking is desired, documents in $\mathcal{D}_q$ and $\mathcal{D}_l$ can be ranked/reranked using either the confidence values or the posterior probabilities output by the classifier, or by using an alternative machine learning method such as a regression or learning-to-rank model.

3.2.4 Query Expansion

When the classifier appears to be achieving a stable precision on the current workset of documents $\mathcal{D}_q$, the system proceeds to expand $\mathcal{D}_q$ in order to increase the recall. This is done through constructing a new query (step 17) and retrieving another set of documents through the search service. Any reasonable query expansion method can be applied here, including the classical relevance feedback methods such as Rocchio's [15] or model-based feedback [28]. Other query reformulation methods can also be applied, such as synonym expansion [24] and semantic term matching [5].

3.2.5 Stop Criteria

Stop criteria of the inner-loop: new labels stop being requested when either of the following conditions is met:

- The performance of the classifier converges. The system correctly predicts the user's labels of a new batch of documents $\mathcal{D}_s$ and, after adding those labels, there is no evident change in the classifier's predictions.
- The user runs out of energy or patience.

Stop criteria of the outer-loop: new queries stop being submitted when either of the following conditions is met:

- New queries no longer pick up new relevant documents. This can be assessed heuristically by running the existing classifier on a new result set, or can be verified by running the inner loop again to check whether any new positive documents are identified.
- The user runs out of energy or patience.

4. INSTANTIATIONS OF REC-REQ

The key components of the general ReQ-ReC framework, document selection, classification, and query expansion can be instantiated in many ways. To illustrate the power of the framework, we describe five instantiations, beginning with iterative relevance feedback as a degenerate form and progressively substituting elements that take greater advantage of the broader framework. Section 5 will provide performance comparisons of these instantiations.

4.1 Iterative Relevance Feedback

Interestingly, an iterative relevance feedback process can be interpreted as a special case of the ReQ-ReC framework, if both the classification component and the document selection component simply adopt a ranking function that is based on the current query, q_i. More specifically, define Θ_R to classify a document as relevant if it is in $retrieve(\mathcal{D}, q_i)$, and define Θ_A to always select the next highest ranked unlabeled item from $retrieve(\mathcal{D}, q_i)$. There is no difference in whether the results retrieved by the previous queries are kept in the document pool $\mathcal{D}_q$ or not, if the results are eventually ranked based on the last query, q_i.

Note that many query updating methods (in the context of relevance feedback) can be applied to generate the new query at each iteration. To establish a baseline for performance comparison, we choose Rocchio's method [15], by which the next query is selected according to Equation 1:

$$\vec{q}_i = \alpha\vec{q}_0 + \beta\frac{1}{|D_r|}\sum_{\vec{d}_j \in D_r}\vec{d}_j - \gamma\frac{1}{|D_{nr}|}\sum_{\vec{d}_k \in D_{nr}}\vec{d}_k, \quad (1)$$

where $\vec{q}_0$ is the original query vector, D_r and D_{nr} are the set of known relevant and nonrelevant documents, and α, β, and γ are parameters. The basic idea of Rocchio's method is to learn a new query vector from documents labeled as positive or negative, and then interpolate it with the original query vector. When the parameters are well tuned, this achieves performance comparable to alternatives such as model-based feedback [28] and negative feedback [25].

4.2 Passive

The next two instantiations modify the relevance feedback process by introducing a separate classifier, Θ_R, rather than using the retrieval function as a degenerate classifier. This classifier is involved to maximize the precision of labels for $\mathcal{D}_q$. Here, keeping the documents retrieved by previous queries does make a difference, because Θ_R will operate at the end to rank all of the results from all of the queries.

Any machine learning-based classifier, as well as any reasonable selection of features, can be used to identify relevant documents in $\mathcal{D}_q$. We adopt the support vector machine (SVM) [2] with unigram features and linear kernel. In cases where a ranked list of documents is desired, documents in $\mathcal{D}_q$ are ranked by the score of the decision function $w^T x + b$ output by linear SVM.

We call this second instantiation of ReQ-ReC *Passive*. It is passive in the sense that the classifier is not used to control the interactive process with the user; we still choose the top-ranked documents for labeling and use Rocchio's method of query expansion, as in our iterative RF instantiation. By comparing the performance of *passive* and the Iterative RF baseline, we can determine the effect of the classifier acting solely as a post-hoc reranking function.

4.3 Unanchored Passive

Note that in Rocchio updating, the parameter that interpolates the new query vector with the original query is quite sensitive. This is because when one relies on the query to maximize both precision and recall, the expansion has to be conservative so that the new query does not drift too far from the original query. When the burden of maximizing precision is transferred from the query to the classifier, we anticipate that this interpolation should become less critical. To test this, we introduce another simple instantiation by removing the original query vector (i.e., the $\vec{q}_0$ component in Equation 1) from Rocchio, by setting $\alpha = 0$. Note that this is a rather extreme case for test purposes. In reality, keeping closer to the original query may still be important even for the purpose of increasing recall. We call this instantiation *Unanchored Passive*, because the updated queries are no longer anchored to the initial query.

4.4 Active

Next, we consider an instantiation of RecQ-ReC that makes use of the classifier to select documents for labeling in the inner loop. As before, we train the classifier using SVM. We select documents for labeling using uncertainty sampling [23], a simple active learning algorithm that selects examples closest to the decision hyperplane learned by the classifier. In each inner-loop iteration, we present to the user ten documents that are the most uncertain by the current classifier. Specifically, five are chosen from each side of the hyperplane. We call this instantiation *Active* because the classifier is active in choosing which documents to label.

Note that after the very first search process, the system has *no* labeled documents in the pool. A classifier cannot be trained and thus the uncertainty sampling cannot be applied. At this *cold start*, we simply select the top 10 documents returned by the search service as the first batch of documents to request user judgments.

As uncertainty-based active learning gradually refines the decision boundary of the classifier, every new query to the search service may affect its performance. This is because a new query expands the pool of documents $\mathcal{D}_q$ with newly retrieved documents, which might dramatically change the distribution and the manifold of data in the search space. At this point, instead of gradually refining the old decision boundary, the classifier may need a bigger push to quickly adapt to the new distribution of data and approach the new decision boundary. In other words, it is important for the classifier to quickly explore the newly retrieved documents. Therefore, in the first inner-loop iteration after each new query brings back new documents, we select top ranked documents for labeling instead of the most uncertain ones. Uncertain ones are picked in the following inner-loop iterations.

4.5 Diverse Active

The final instantiation we consider modifies the query expansion algorithm used in the Active instantiation. Previously, we considered an unanchored version of Rocchio's method of selecting the next query. Here, we consider a different modification of Rocchio's method.

To maximize recall, we naturally want a new query to retrieve as many relevant documents as possible. Even more importantly, these relevant documents should overlap as little as possible with the documents retrieved by previous queries. In other words, a new query should retrieve as many *new relevant* documents as possible.

Our idea is inspired by the theory of "weak ties" in sociology [6]. While strong ties trigger social communication, weak ties can bring in novel information. If we think of the top-ranked documents in a retrieved list as "strong ties" to the query, we can think of the lower-ranked documents as "weak ties." We thus exploit documents that are judged as relevant, but ranked lower in the list returned by the search service. These documents are likely to act as bridges to expand the search space into other clusters of relevant documents.

Are there many such documents? In a relevance feedback process, there might be few, as the user always labels the top-ranked documents. In a ReQ-ReC process that actively selects documents, however, documents ranked lower by the retrieval function are more likely to be viewed and judged by the user.

In Equation 1, instead of using all relevant documents D_r, we use its subset D_{rl}, which includes the documents that are judged as relevant but ranked low by the original retrieval function. We employ a simple criterion to determine which documents should be included in D_{rl}. For each document d, we maintain its rank returned by the retrieval function, denoted as r_d. If the document has been retrieved by multiple queries in the past, its highest rank in those retrieved lists is kept. Let r_l be the lowest rank r_d of all the documents in D_r. We include documents that are ranked lower than $r_l/2$ in D_{rl}. This leads to inclusion in the next query of terms from relevant documents that were not highly weighted in previous queries. Since this method aims to diversify new queries, while still using the classifier to actively choose documents for labeling, we refer to this method as *Diverse Active*.

5. EXPERIMENTS

In this section, we present empirical experiments to evaluate the effectiveness of the ReQ-ReC framework and its instantiations. We start with a description of the data sets, metrics, and methods included in the comparisons.

5.1 Data Sets

There are several criteria for selecting the right data sets for evaluating ReQ-ReC. Ideally, the data sets should be large enough and standard search APIs should exist. A representative set of queries should also exist, and each query should have a reasonable number of relevant documents in the data set. To avoid the high variance of real-time user judgments and to facilitate comprehensive and fair comparisons, we use existing judgments for each query to 'automate' the actual user feedback in the process. The same approach is used in most existing work on relevance feedback (e.g., [8, 20, 25]). We therefore require that many relevant judgments exist for each query.

We first select four large scale TREC data sets, the data sets used in TREC-2012 Microblog Track (MB12) [21], TREC-2013 Microblog Track (MB13)[1], the TREC-2005 HARD Track (HARD), and the TREC-2009 Web Track (ClueWeb09[2], category A)[3]. These data sets normally provide 50–60 queries and 500–1,000 relevant judgments for a query.

[1] `https://github.com/lintool/twitter-tools/wiki/`
[2] `http://lemurproject.org/clueweb09/`
[3] `http://trec.nist.gov/data/web09.html`

Note that there is a natural deficiency of using TREC judgments for the evaluation of a high-recall task, simply because not all documents in a TREC data set have been judged. Instead, judgments are provided for only a pool of documents that consist of the top-ranked documents submitted by each participating team. In many cases, only a sample of the pool is judged. Therefore, it is likely that many relevant documents for a query are actually not labeled in the TREC provided judgments. This creates a problem for a 'simulated' feedback process—when the system requests the label of a document, the label may not exist in the TREC judgments. It is risky to label that document either as relevant or as irrelevant, especially because mislabeling a relevant documents as irrelevant may seriously confuse a classifier. In such situations, we ignore that document and fetch the next document available. The same treatment has been used in the literature [20]. When measuring the performance of a retrieved list, however, we follow the norm in the literature and treat a document not judged by TREC as negative.

Table 2: Basic information of data sets

	#docs	avg dl	#topics(IDs)	#qrels
20NG	18,828	225	20 categories	18,828
HARD	1,033,461	353	50 (303-689)	37,798
MB12	15,012,766	19	59 (51-110)	69,045
MB13	≈243,000,000	14	60 (111-170)	71,279
ClueWeb09	503,903,810	1570	50 (1-50)	23,601

* HARD has non-consecutive topic IDs. Topic 76 of MB12 has no judgment hence is removed.

To better understand the behavior of ReQ-ReC, it is desirable to include a data set that is fully judged, even though a large data set like that is rare. Therefore, we include the 20-newsgroup data set (20NG) [11] for this purpose. As every document belongs to one of the 20 topics, we use the titles of 20 topics as the queries, following the practice in [4]. For words that are abbreviated in the topic titles, we manually expand them into the normal words. For example, "rec" is converted to "recreation," and "autos" to "automobiles." Although it is feasible to apply a classifier to the entire 20NG data set, we only access the data using rate-limited retrieval functions. The statistics of all five data sets in our experiments are presented in Table 2.

Both the 2013 Microblog Track[4] and the ClueWeb09[5] provide official search APIs, which are implemented using the Dirichlet prior retrieval function (Dirichlet) [29]. For other data sets, we maintain a similar search service using Lucene,[6] which also implements the Dirichlet prior function. Documents are tokenized with Lucene's StandardAnalyzer and stemmed by the Krovetz stemmer [10]. *No* stopwords are removed.

5.2 Metrics

Many popular metrics for retrieval performance, such as *precision@K* and NDCG, are not suitable for high-recall tasks. We use two standard retrieval metrics that depend more on recall, namely the mean average precision (MAP) [12] and the R-precision (R-Prec) [12]. R-precision measures the precision at the R-th position for a query with R relevant judgments. The R-th position is where precision equals recall. To increase R-precision, a system has to simultaneously increase precision and recall. For each query, we use the top 1,000 relevant documents (either labeled or predicted) to compute the measures.

When measuring performance, we include documents that the user labeled during the process. This is because a high-recall retrieval task is successful when more relevant documents can be found, whether they are actually judged by the user or predicted by the system. If an interactive process does a good job of presenting more relevant documents to the user, it should not be punished by having those documents excluded from the evaluation. In all methods included in comparative evaluation, we put the documents judged as relevant at the top of the ranked list, followed by those predicted to be relevant using Θ_R.

5.3 Methods

We summarize all baseline methods and ReQ-ReC instantiations included in our evaluation in Table 3. The most important baseline we are comparing with is the iterative relevance feedback as described in Section 4.1, in which a new query is expected to maximize both precision and recall. We then include four instantiations of the ReQ-ReC framework, as described in Section 4.

In *Passive* and *Unanchored Passive*, we employed a negative form of pseudo-relevance feedback: the lowest ranked 1,000 documents retrieved by the final query are treated as negative examples to train the classifier. The positive examples for training came from the actual judgments.

5.4 Parameters

For the MP13 and ClueWeb09 datasets, we used the official search APIs, which returned, respectively, 10,000 and 1,000 documents per query. For the three data sets without official search APIs, the parameter of the Dirichlet prior μ for the base retrieval function was tuned to maximize the mean average precision and each query returned the top 2,000 matching documents.

To obtain the strongest baseline, we set the parameters of Rocchio to those that maximize the mean average precision of a relevance feedback process using 10 judgments. We fix α to be 1 and conduct a grid search on the other two. For ClueWeb09, we set the parameters according to the recommendation in [12] as the rate limits of the API prevent us from tuning the parameters. We do not further tune the parameters in the ReQ-ReC methods even though the optimal parameters for the baseline may be suboptimal for ReQ-ReC. The values of all the parameters used are shown in Table 4. In all our experiments, we also use the default parameter of SVM ($c = 1$). We stop the inner-loops when SVM confidence value produces stable ranking of $\mathcal{D}_q$, i.e., Spearman's rank correlation coefficient of previous and current rankings of $\mathcal{D}_q$ is above 0.8 for two consecutive inner-loops.

5.5 Overall Performance

Table 5 summarizes the performance of all included methods, with one additional criterion to stop the process when the "user" has judged 300 documents for a topic. Statistical significance of the results are provided by comparing to the baseline, iterative relevance feedback, and by comparing to another ReQ-ReC method. In general, methods developed

[4] https://github.com/lintool/twitter-tools/wiki/TREC-2013-API-Specifications
[5] http://boston.lti.cs.cmu.edu/Services
[6] http://lucene.apache.org/

Table 3: Baselines and methods included in comparison.

Method	Doc. Selection	Classification	Query Expansion	# outer loops	# inner loops
Relevance Feedback (RF)	top	-	Rocchio	1	1
Iterative RF	top	-	Rocchio	M	1
Passive	top	SVM at end	Rocchio	M	1
Unanchored Passive (Unanchored)	top	SVM at end	Rocchio - $\vec{q}_0$	M	1
Active	uncertainty	SVM	Rocchio	M	M
Diverse Active (Diverse)	uncertainty	SVM	divRoc	M	M

* **M**: multiple iterations; **top**: select 10 top-ranked documents; **uncertainty**: uncertainty-based active document selection; **divRoc**: diverse Rocchio; **Rocchio -** $\vec{q}_0$: Rocchio without interpolation of the original query.

Table 4: Parameter settings: μ in Dirichlet prior; β and γ in Rocchio (α fixed as 1); Results per query: number of documents returned by a search API call.

	MB12	MB13	ClueWeb09	HARD	20NG
μ	2100	-	-	1100	3200
β	0.95	0.85	0.75	0.6	0.5
γ	0.4	0.15	0.15	0.05	0.4
Results/query	2,000	10,000	1,000	2,000	2,000

under the ReQ-ReC framework significantly outperform iterative relevance feedback. *Diverse Active*, which uses an active document selection strategy and a diverse query expansion, achieves the best performance. For most data sets, the improvement over iterative relevance feedback is as large as 20% – 30% of MAP and R-Precision. This is promising given the difficulty of improvements based on those two metrics. On the largest data set, ClueWeb09, the best ReQ-ReC algorithm achieves more than 120% improvement over iterative relevance feedback.

We make the following remarks:

- (Compare *Relevance Feedback* with *Iterative RF*) Multiple iterations of relevance feedback indeed outperforms a single iteration of feedback, even if the same number of judgments (i.e., 300) are used in this single iteration. The only exception is the ClueWeb09 data, for which the collection is too large and the relevance judgments are very sparse. In this case, an iterative relevance feedback method may stop earlier if none of the top 10 results brought back by a new query are relevant. In that situation, presenting more documents to the user at once may be less risky.

- (Compare *Iterative RF* with *Passive* and *Unanchored-Passive*) Distributing the burden of maximizing precision to a classifier is effective, even if the classifier is only involved at the end of the process. Iterative relevance feedback relies on the new query to maximize both precision and recall. By simply keeping the results retrieved by all previous queries and classifying them at the end (by an SVM trained on accumulated judgments), the retrieval performance increases significantly on *all* the data sets (*Passive*). Since the involvement of the classifier releases the burden of the queries to maximize precision, we anticipate that the queries no longer have to be tied closely to the original one. Indeed, even if we strip the effect of the original query from every expanded query (*Unanchored-Passive*), the ReQ-ReC process still yields results comparable to—and sometimes even better than—anchored query expansion (*Passive*). The performance is further improved when the classifier is involved in all the iterations instead of being applied at the end (*Active*).

- (*Active*) A straightforward active document selection approach (which picks the documents that the classifier is the least certain about) outperforms picking documents from the top of the ranked list. This is consistent with the observations in literature [22]. By actively selecting documents to present to the user, her effort of labeling documents is significantly reduced.

- (*Diverse Active*) The diverse query expansion method inspired by the weak-tie theory is clearly the winner on all five data sets. By moving the burden of precision to a classifier, the objective of a new query is purely to bring new relevant documents into the pool of retrieved documents. This gives freedom to the queries to expand the search space aggressively, and provides a great opportunity to investigate new algorithms that are particularly suitable for this goal.

5.6 Learning Behavior Analysis

The previous section summarizes the performance of ReQ-ReC methods when the stop criteria are met. To better understand the behavior of ReQ-ReC, we provide the following analysis and plot the intermediate performance of three methods (*Iterative RF*, *Active*, and *Diverse Active*) throughout the user-interaction process. Note that each topic may accumulate judgments at a different pace and meet stop criteria earlier or later. We interpolate a per-topic curve by a piecewise linear function, and extrapolate it by extending the end-point constantly to the right. These per-topic curves are then averaged to generate the aggregated curve.

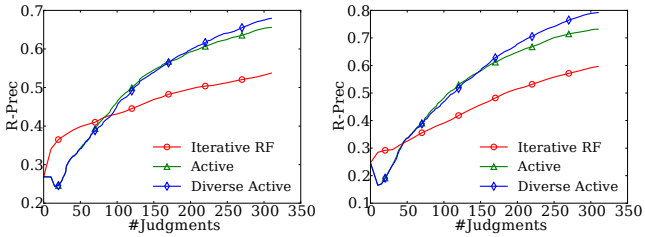
Figure 3 plots the performance of each method against the number of documents the "user" has judged so far throughout the ReQ-ReC process, measured using R-precision. All three curves start at the same point where there is no user judgment. At that point the ranking is essentially based on the original retrieval function (i.e., Dirichlet prior). When user judgments are beginning to be collected, there is a significant gain by iterative relevance feedback. Performance increases rapidly at the first 2 runs (20 judgments), and the growth becomes much slower after that. This is consistent with the findings in literature.

Methods developed under the ReQ-ReC framework (*Active* and *Diverse Active*) do not really take off until we obtain a reasonable number of judgments (50 on the HARD data set and 90 on the microblog data set). This is ascribed to

Table 5: Retrieval performance of competing methods. At most 300 judgments per topic. ReQ-ReC methods significantly outperform iterative relevance feedback.

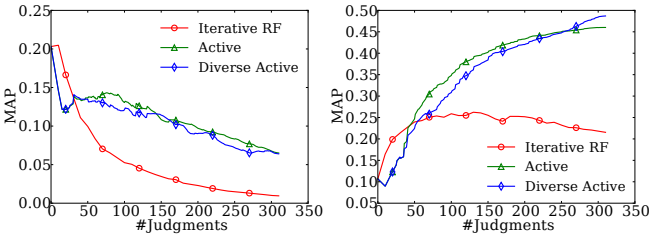
	MB13		MB12		HARD		20NG		ClueWeb09	
	R-prec	MAP	R-prec	MAP	R-prec	MAP	R-prec	MAP	R-prec	MAP
Dirichlet	0.268	0.203	0.233	0.183	0.247	0.174	0.327	0.107	0.101	0.058
RF	0.417	0.415	0.466	0.479	0.440	0.447	0.451	0.356	0.256	0.229
Iterative RF	0.532	0.552	0.633	0.649	0.592	0.597	0.474	0.421	0.237	0.216
Passive	0.568**	0.585**	0.646**	0.661**	0.615**	0.637**	0.548**	0.490**	0.275**	0.247**
Unanchored	0.603**▽	0.618**▽	0.667**▽	0.673**▽	0.609	0.624**	0.527*	0.464*	0.268**	0.236**
Active	0.653**▽	0.661**▽	0.727**▽	0.740**▽	0.729**▽	0.737**▽	0.595**▽	0.562**▽	0.493**▽	0.493**▽
Diverse	**0.675****△ (+27%)	**0.692****△ (+25%)	**0.760****△ (+20%)	**0.771****△ (+19%)	**0.789****△ (+33%)	**0.799****△ (+34%)	**0.620****▽ (+31%)	**0.580****▽ (+38%)	**0.533****△ (+125%)	**0.533****△ (+147%)

** and * indicate the improvement over *Iterative Relevance Feedback* is statistically significant according to Wilcoxon signed rank test at the significance level of 0.01 and 0.05; ▽: the improvement over *Passive* is significant at the level of 0.05; △: the improvement over *Active* is significant at the level of 0.05; ($+x\%$) indicates the percentage of improvement over the baseline *Iterative RF*.



(a) MB13 (b) HARD

Figure 3: R-Prec vs. labeling effort



(a) MB13 (b) 20NG

Figure 4: Residual Analysis

the "cold start" problem of supervised classification. When few labeled documents are available, the performance of a classifier does not outperform a simple ranking function.

As stated before, a ReQ-ReC process targets users who truly seek a high recall of relevant documents and are therefore willing to spend more effort on interacting with the system and labeling more results. Indeed, after the first few iterations, the two methods developed under ReQ-ReC framework improve dramatically and become significantly better than iterative relevance feedback. For the users who are reluctant to label more than 50 documents, conventional relevance feedback may still be a better choice.

The cold start implies that there is considerable room for improving the performance of the ReQ-ReC. For example, a semi-supervised classifier may be used early on to achieve better precision with few training examples.

We also notice that the benefit of *Diverse Active* over *Active* kicks in later in the process, when there are around 150 judgments collected. At that point, getting new relevant documents becomes more challenging, as many documents retrieved by the new query may have already been retrieved by a previous query. At this stage, introducing some diversity to the query expansion brings in considerable benefit. Similar observations are made on the other three data sets.

Another interesting analysis is how well a method works with documents that have not been selected for labeling so far. We are particularly interested in this behavior because we have decided to include all judged documents when measuring the performance of the system (see Section 4).

We plot the residual MAP in Figure 4, which is the mean average precision computed purely based on documents that have not been presented to the user so far in the process. In general, the two ReQ-ReC methods (*Active* and *Diverse Active*) do a much better job in finding the relevant documents and ranking them high, even if they are not judged by the user. On the microblog data set, we see that the residual MAP decreases when more documents are presented to and labeled by the user. This may be simply because there are fewer relevant documents remaining in the collection. However, it is also likely due to the fact that the TREC judgments are not complete. There might be many relevant documents that were not judged by TREC at all. If a method successfully finds those documents, its performance may be significantly undervalued simply because we have to treat these documents as negative in computing the metrics.

We are therefore interested in how ReQ-ReC behaves if the data set is fully judged. Looking at the curves on the 20NG, we observe a contrary pattern, where the two ReQ-ReC methods actually enjoy a continuous growth of residual MAP, while the same metric for iterative feedback is still dropping. This is a promising finding that indicates the performance of ReQ-ReC may be underestimated on data sets with incomplete judgments (i.e., TREC data sets).

6. CONCLUSION

We present ReQ-ReC (ReQuery-ReClassify), a double-loop retrieval framework that is suitable for high-recall retrieval tasks without sacrificing precision. The interactive process combines iterative expansion of a query set with iterative refinements of a classifier. The work of maximizing precision and recall is distributed so that the queries increase recall and the classifier handles precision.

The ReQ-ReC framework is general, which includes classi-

cal feedback methods as special cases, and also leads to many instantiations that use different combinations of document selection, classification, and query expansion methods. The framework is very effective. Some instantiations achieved a 20% – 30% improvement of mean average precision and R-precision on most data sets, with the largest improvement up to 150% over classical iterative relevance feedback.

In order to clearly illustrate the power of the framework, we have intended to keep all the instantiations simple. It is a promising future direction to optimize the choices and combinations of the key components of the ReQ-ReC framework. Findings from our experiments also indicate possibilities for investigating new classification and query expansion algorithms that are particularly suited to this framework.

Acknowledgment. The authors thank Sam Carton, Kevyn Collins-Thompson, ChengXiang Zhai, and reviewers for their useful comments. This work is partially supported by the National Science Foundation under grant numbers IIS-0968489 and IIS-1054199, and partially supported by the DARPA under award number W911NF-12-1-0037.

7. REFERENCES

[1] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using smart: Trec 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48, 1995.

[2] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[3] I. J. Cox, M. L. Miller, S. M. Omohundro, and P. N. Yianilos. Pichunter: Bayesian relevance feedback for image retrieval. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 361–369. IEEE, 1996.

[4] H. Drucker, B. Shahrary, and D. C. Gibbon. Support vector machines: relevance feedback and information retrieval. *Information Processing and Management*, 38(3):305–323, 2002.

[5] H. Fang and C. Zhai. Semantic term matching in axiomatic approaches to information retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 115–122. ACM, 2006.

[6] M. S. Granovetter. The strength of weak ties. *American journal of sociology*, pages 1360–1380, 1973.

[7] D. A. Hanauer. Emerse: the electronic medical record search engine. In *AMIA Annual Symposium Proceedings*, volume 2006, page 941. American Medical Informatics Association, 2006.

[8] D. Harman. Relevance feedback revisited. In *Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '92, pages 1–10, New York, NY, USA, 1992. ACM.

[9] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 154–161. ACM, 2005.

[10] R. Krovetz. Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–202. ACM, 1993.

[11] K. Lang. Newsweeder: Learning to filter netnews. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339, 1995.

[12] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

[13] D. W. Oard and W. Webber. *Information Retrieval for E-Discovery*. Foundations and Trends in Information Retrieval. Now Publishers, 2013.

[14] K. Raman, P. N. Bennett, and K. Collins-Thompson. Toward whole-session relevance: Exploring intrinsic diversity in web search. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '13, pages 463–472. ACM, 2013.

[15] J. Rocchio. Relevance feedback in information retrieval. In *The SMART retrieval system experiments in automatic document processing*, pages 313–323. Prentice Hall, 1971.

[16] Y. Rui, T. S. Huang, M. Ortega, and S. Mehrotra. Relevance feedback: a power tool for interactive content-based image retrieval. *Circ. and Sys. for Video Tech., IEEE Transactions on*, 8(5):644–655, 1998.

[17] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. In *Journal of the American Society for Information Science (1986-1998)*, volume 41, pages 288–297. ACM, 1990.

[18] B. Settles. Active learning literature survey. *University of Wisconsin, Madison*, 2010.

[19] X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '05, pages 43–50. ACM, 2005.

[20] X. Shen and C. Zhai. Active feedback in ad hoc information retrieval. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 59–66. ACM, 2005.

[21] I. Soboroff, I. Ounis, C. Macdonald, and J. Lin. Overview of the trec-2012 microblog track. In *Proceedings of the Twenty-First Text REtrieval Conference*, 2012.

[22] A. Tian and M. Lease. Active learning to maximize accuracy vs. effort in interactive information retrieval. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 145–154. ACM, 2011.

[23] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *J. Mach. Learn. Res.*, 2:45–66, Mar. 2002.

[24] E. M. Voorhees. Query expansion using lexical-semantic relations. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 61–69. ACM, 1994.

[25] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 219–226. ACM, 2008.

[26] Z. Xu, R. Akella, and Y. Zhang. Incorporating diversity and density in active learning for relevance feedback. In *Advances in Information Retrieval*, pages 246–257. Springer, 2007.

[27] Z. Xu, X. Xu, K. Yu, and V. Tresp. A hybrid relevance feedback approach to text retrieval. In *European Conference on Information Retrieval*, Lecture Notes in Computer Science, pages 281–293. Springer, 2003.

[28] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 403–410. ACM, 2001.

[29] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 334–342. ACM, 2001.

[30] X. S. Zhou and T. S. Huang. Relevance feedback in image retrieval: A comprehensive review. *Multimedia systems*, 8(6):536–544, 2003.