Active Learning to Maximize Accuracy vs. Effort in Interactive Information Retrieval

Aibo Tian
Department of Computer Science
The University of Texas at Austin
atian@cs.utexas.edu

Matthew Lease
School of Information
The University of Texas at Austin
ml@ischool.utexas.edu

ABSTRACT

We consider an interactive information retrieval task in which the user is interested in finding several to many relevant documents with minimal effort. Given an initial document ranking, user interaction with the system produces relevance feedback (RF) which the system then uses to revise the ranking. This interactive process repeats until the user terminates the search. To maximize accuracy relative to user effort, we propose an active learning strategy. At each iteration, the document whose relevance is maximally uncertain to the system is slotted high into the ranking in order to obtain user feedback for it. Simulated feedback on the Robust04 TREC collection shows our active learning approach dominates several standard RF baselines relative to the amount of feedback provided by the user. Evaluation on Robust04 under noisy feedback and on LETOR collections further demonstrate the effectiveness of active learning, as well as value of negative feedback in this task scenario.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Relevance Feedback; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation, Performance

Keywords

Relevance Feedback, Active learning, Personalized Search

1. INTRODUCTION

This paper presents an interactive information retrieval (IR) task [14] based on iterative relevance feedback (RF) [20, 4]. Given an initial document ranking, the user interacts with the system to explicitly or implicitly provide the system with labeled feedback documents. In standard RF fashion, the system then utilizes this feedback to generate an improved ranking. However, our interaction model involves an iterative back-and-forth between user and system, providing the user a natural way to provide feedback and the system with an opportunity to request targeted feedback to improve ranking. User behavior is simulated via a simple model of top-to-bottom search result browsing (§2). Active learning [24, 13] is used to maximize system accuracy while minimizing the amount of feedback required (§4). Figure 1 depicts our overall task scenario and system architecture.

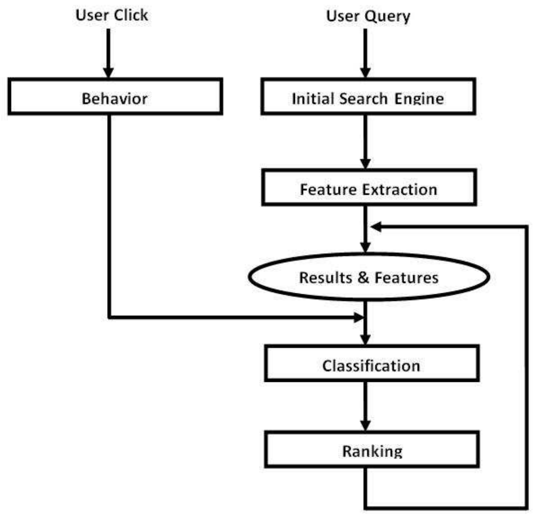


Figure 1: Task scenario and system architecture.

Brandt et al. [3] recently described an interactive IR creating a dynamic ranking tree for user interaction. Ranking is based on recommendations from the history of other users. Radlinski and Joachims [17] also explore use of active learning to quickly learn the document ranking function, focusing on estimating the general relevance measure (and ranking) of documents based on clickthrough data from query logs. Our work is more user-specific in personalizing the ranking based for the current user's interactions.

We investigate two active learning methods for selecting the next document to slot high for feedback. The *Simple Margin* method [24] picks the document lying closest to the decision surface whose relevance is maximally uncertain. We also propose an enhancement, *Local Structure*, which captures the idea that useful examples to label should also be far from already labeled examples and near to other unlabeled examples. We use Laplacian Score feature selection [10] to prune the vocabulary for more tractable learning (§5).

Our iterative RF task scenario presents several challenges to standard evaluation methodology typically used for ad hoc retrieval and single-iteration RF (§6). To address these challenges, we propose a paired approach of *KeepAll* and *TakeOut* evaluation which differ in how feedback documents are treated with regard to evaluation. Both strategies have advantages and disadvantges. In tandem, they provide a useful way to compare relative effectiveness of methods while not having to exclude the full union of feedback documents as typical with *residual* evaluation, something which can be difficult to apply when relevant documents are limited.

Empirical evaluation is conducted on the Robust04 TREC collection, as well as the LETOR 3.0 TD2003 and TD2004 collections [15] (§7). While we primarily assume relevance feedback provided by the user is correct, as in traditional RF settings, we also consider a noisy feedback setting in which the user sometimes provides erroneous feedback. This is simulated through simple false positive and negative rate Bernoulli parameters. Active learning methods are compared to standard RF baselines of Rocchio [20] and model-based feedback [28] (§3). We also evaluate Rocchio using positive-only feedback, and results show negative feedback is quite valuable in this task scenario, something rarely seen in prior work [26]. Overall, learning curves and average performance show active learning methods dominate baseline techniques, maximizing ranking accuracy for users relative to the amount of user feedback provided.

A simple prototype version of our system is available online[1] which reranks search results from Bing[2] using active learning. The interface presents left and right views of search results. The original ranking remains fixed in the left view, while the right view is continually re-ordered based on user feedback. Clicking search results in either views will open the clicked page in a new tab, as well as re-order search results in the view on the right.

Our paper is organized as follows. Our task scenario and user behavior model is introduced in §2. §3 defines the baseline RF methods we employ. Active learning methods are presented in §4. §5 describes feature selection for dimensionality reduction. Evaluation methodology for iterative RF is discussed in §6, and our evaluation of methods is presented in §7. We conclude with discussion of future work in §8.

2. TASK SCENARIO AND USER MODEL

Imagine a user who is interested in finding several to many relevant documents (with minimal effort). Given an initial document ranking, user interaction with the system produces relevance feedback (explicitly or implicitly) which the system then utilizes to improve the ranking. This interactive process repeats iteratively until the user terminates the search. In practice, we expect different information needs, task situations, and users will drive different types of thresholds for stopping. These might include a limited amount of effort the user is willing to invest, a target number of relevant documents to be found, or perhaps a target system accuracy to achieve (e.g. for later use searching the collection as new documents are added to it). Iteration also stops whenever the set of relevant documents available for feedback is exhausted. In all cases, the system goal is to maximize ranking accuracy for any amount of relevance feedback provided. In other words, the system should maximize the *learning curve*.

We generate relevance feedback from a simple model of user behavior. Given a document ranking, the user browses the document ranking from top-to-bottom and clicks on the first relevant document encountered. This interaction generates relevance feedback consisting of both one relevant document (the one clicked) and a set of zero or more non-relevant documents (those ranked above the relevant document). We do not explicitly model the user's stopping criterion for terminating the search (evaluation in §7 will measure system accuracy across varying amounts of user feedback).

Note that the user model just described assumes user feedback is always correct, similar to a traditional RF setting with known relevant and non-relevant documents [4]. We also consider a second, noisy model of user feedback (§7.3). Such noise might arise in practice with either explicit or implicit feedback due to the user's misperceptions of relevance prior to clicking or by simple clicking mistakes. This second user model introduces two Bernoulli parameters, f_p and f_n, which control the user's false positive and negative rates, respectively. As the user browses each non-relevant document, with probability f_p he will mistakenly click on it to indicate it as relevant. Similarly, whenever the user browses a relevant document, with probability f_n he will fail to click on it and instead continue scanning down the results list, providing incorrect negative feedback.

3. RELEVANCE FEEDBACK BASELINES

We evaluate three baseline RF methods for this iterative feedback task: (1) Rocchio feedback [20], (2) Rocchio with positive feedback only, and (3) model-based feedback (language modeling paradigm) [28]. Since prior work has rarely observed benefits from negative feedback [26], we were particularly interested in comparing the relative performance of (1) and (2) for this task scenario.

Let D denote the document collection, D_U the set of unlabeled documents, D_F documents with feedback, D_{F+} positive feedback documents, and D_{F-} the set of negative feedback documents. Let $\mathbf{q}$ and $\mathbf{d}$ denote the query and document vectors. Superscript k represents the kth iteration. $S(\mathbf{d})$ represents $\mathbf{d}$'s final score.

Rocchio [20] performs query expansion based on both positive and negative user feedback. Let $\mathbf{q^k} = \alpha\mathbf{q^0} + (1-\alpha)\mathbf{q_f^{k-1}}$ denote linear interpolation between original query $\mathbf{q^0}$ and feedback vector $\mathbf{q_f^k}$. Using term frequency representation for $\mathbf{q}$ and $\mathbf{d}$ vectors, we compute $\mathbf{q_f^k}$ by:

$$\mathbf{q_f^k} = \frac{\beta}{|D_{F+}^k|} \sum_{\mathbf{d} \in D_{F+}^k} \frac{\mathbf{d}}{|\mathbf{d}|} - \frac{(1-\beta)}{|D_{F-}^k|} \sum_{\mathbf{d_i} \in D_{F-}^k} \frac{\mathbf{d}}{|\mathbf{d}|} \qquad (1)$$

The score of $\mathbf{d}$ in iteration k is then computed by cosine similarity between the expanded query and document $\mathbf{d}$'s vector: $S^k(\mathbf{d}) = cos(\mathbf{q^k}, \mathbf{d}) = \mathbf{q^k} \cdot \mathbf{d}$.

Model-based feedback [28] incorporates only positive feedback into the language model. Let Θ_Q, Θ_D, Θ_F, and Θ_d represent the model for query Q, collection D, positive feedback documents F, and given document $\mathbf{d}$. Similar to Rocchio, let $\Theta_Q^k = \alpha\Theta_Q^0 + (1-\alpha)\Theta_F^k$ denote the updated query model computed by linear interpolation of the original query model and the feedback model. To limit the noise of the feedback documents, feedback documents are assumed to be generated based on linear combination of feedback model and the

[1]http://www.cs.utexas.edu/~atian/search
[2]http://www.bing.com

collection model (2). Let $c(w, \mathbf{d})$ denote the frequency of word w in document $\mathbf{d}$. The feedback model is estimated by maximum likelihood (ML) using the EM algorithm where $\log p(D^k_{F+}, \Theta^k_F)$ is computed by:

$$\sum_{\mathbf{d} \in D^k_{F+}} \sum_w c(w, \mathbf{d}) \log \left(\lambda p(w|\Theta^k_F) + (1-\lambda) p(w|\Theta_D) \right) \quad (2)$$

We estimate the document model by $\Theta_d = (1-\gamma)\frac{\mathbf{d}}{|\mathbf{d}|} + \gamma\, \Theta_D$, where smoothing parameter γ mixes $\mathbf{d}$'s ML estimate with the collection model Θ_D. Document relevance to the query is then measured by KL divergence: $S^k(\mathbf{d}) = -\mathcal{D}(\Theta^k_Q || \Theta_\mathbf{d})$.

4. ACTIVE LEARNING

To maximize ranking accuracy with respect to the amount of user effort (i.e. relevance feedback) invested in training the system, we propose an active learning strategy for utilizing feedback. At the core of this active learning is supervised classification: learning to distinguish between relevant and non-relevant classes given training data. Documents are ranked in order of decreasing estimated relevance [19]. We describe two active learning approaches, **Simple Margin** and **Local Structure**, which we later evaluate (§7).

We adopt a support vector machine (SVM) [24] approach to active learning. Training a linear SVM consists of solving a quadratic optimization subject to linear inequality constraints. Optimization maximizes the margin between positive and negative samples so the classifier generalizes well to unseen data. The version (i.e. parameter) space is the dual space of feature space. A point in version space corresponds to a hyperplane in the feature space and vice-versa. SVM optimization in the feature space corresponds to finding the largest sphere in the version space, subject to constraints of the hyperplanes for the current feature points.

An active learning algorithm selects which unlabeled example to be labeled next in order to maximize learning relative to user effort required.The key problem of active learning is estimating the expected benefits of labeling different data points. Tong and Koller [24] find that the best example to label next halves the version space. Unfortunately, explicitly computing the size of version space is computationally infeasible. Consequently, Tong and Koller identify useful approximation criteria for example selection. Their **Simple Margin** criterion requires minimal computational costs so can be practically applied in real-time. It predicts utility of labeling all unlabeled data based on current SVM model, then chooses the example lying nearest to the classification hyperplane via $M(\mathbf{d}, \Theta)$, where d is the document vector in TF-IDF (normalized $\frac{tf}{logdf}$), Θ parameterizes the SVM model trained with all labeled data, and $M(\mathbf{d}, \Theta)$ is the positive distance of $\mathbf{d}$ from the model's hyperplane (capturing uncertainty of $\mathbf{d}$'s classification under the model).

Because this criterion does not require additional training, its computational cost is low. Its effectiveness, however, depends on the shape of the version space. If the version space is far from spherical, this criterion may not perform well. The middle figure in Figure 2 shows such an example failure case of this criterion. Tong and Koller's approximation criteria select which example to label next entirely based upon the learned SVM model [24]. In other words, the structure information of the whole data set is missed in those criteria.

However, often it is the case (as it is here) that we actually know about the entire dataset, including the labeled and

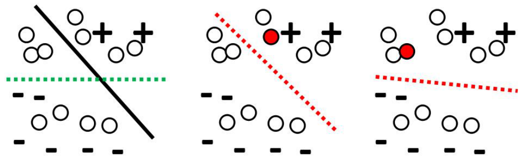


Figure 2: An example illustrating the difference between Simple Margin and Local Structure criteria. "+", "-", and circle represent positive, negative and unlabeled examples. The left figure shows the predicted classification prior to active learning. The black solid line denotes the current hyperplane, while the green dashed line shows the optimal hyperplane. The middle and right figures demonstrate the behavior of Simple Margin and Local Structure criteria, respectively. Red points represent examples selected by active learning for labeling, and the red dashed line shows the updated hyperplane after this labeling. While Simple Margin selects the example closest to the current hyperplane, Local Structure selects an example both close to the hyperplane and away from already labeled examples.

unlabeled. Such additional information can be usefully exploited by the learner. For example, work in semi-supervised learning often exploits the imbalance between plentiful unlabeled data and scarce labeled data by utilizing more accurate estimates of variance on the unlabeled data to improve classifier accuracy. In this case, we propose another selection criterion which considers both the performance of SVM *and* the structure information of the whole data set in tandem.

Intuitively, a useful unlabeled example to label should embody several characteristics. First, the classifier should be uncertain of its label prediction for the example given current model parameters; this is captured by the Simple Margin criterion. A second property, however, is that example should not lie close to already labeled data since close examples are likely to have the same label (e.g. standard nearest neighbor approaches to classification or utilizing unlabeled data in semi-supervised learning). Such close examples are unlikely to improve the performance of the classifier much more than a single, more useful example (e.g. see Figure 2). As a third characteristic, the data should have nearby unlabeled neighbors so knowing the example's label is likely to help us better label other nearby examples.

To capture the three properties above, we propose a new active learning selection criterion we refer to as **Local Structure**. Let $SL(\mathbf{d}) = \max_{\mathbf{d_f} \in D_F} cos(\mathbf{d}, \mathbf{d_f})$ find the maximum cosine similarity between $\mathbf{d}$ and all labeled $\mathbf{d_f} \in D_F$, i.e. how close $\mathbf{d}$ is to another already labeled document. Let $\mathbf{d_m}$ denote $\mathbf{d}$'s mth closest neighboring document, where m is a parameter. $SN(\mathbf{d}) = cos(\mathbf{d}, \mathbf{d_m})$ measures cosine similarity of $\mathbf{d}$ and $\mathbf{d_m}$ to model $\mathbf{d}$'s local neighborhood. The intuition is that smaller values of SN indicate a greater number of nearby documents that might benefit by labeling $\mathbf{d}$. We minimize a linear combination of SL, SN and $M(\mathbf{d}, \Theta)$ to select an example $\mathbf{d}'$ to label capturing the three properties:

$$\mathbf{d}' = \arg\min_{d \in D_U} \alpha M(\mathbf{d}, \Theta) + (1-\alpha)\left(SL(\mathbf{d}) - SN(\mathbf{d})\right) \quad (3)$$

Figure 2 visually compares the differing behaviors of active learning based on Simple Margin and Local Structure criteria. In this figure, Simple Margin criterion is seen to select the point nearest to the hyperplane without considering structure information. The chosen example lies near to an already labeled example, resulting in minimal change to the classification hyperplane and thereby minimal benefit to learning. In contrast, Local Structure criterion chooses the point close to the hyperplane and away from labeled data. In doing so, it thereby avoids the failure case of Simple Margin.

In our iterative RF scenario, the document selected for feedback is "slotted" into the top position in the ranking to guarantee the user provides feedback for it. While this ensures the system obtains the most useful feedback, this will often hurt ranking accuracy (since the maximally uncertain document will be non-relevant roughly half of the time). We discuss this issue further in regard to evaluation methodology (§6.2) and our future work (§8).

5. FEATURE SELECTION

High dimensionality represents a general obstacle to effective machine learning. Natural language learning tasks exhibit an especially visible example of this due to fast vocabulary growth from many words used infrequently. Various prior work in IR has investigated dimensionality reduction methods to mitigate issues of term mismatch between semantically related terms (e.g. LSI [8], pLSI [12], and LDA [27]). Learning to rank methods are also impacted if we use term-level features rather than higher-level aggregate features as used in LETOR [15, 1]. As such, we employ dimensionality reduction methods for more tractable learning.

In particular, we apply a dimensionality reduction technique known as Laplacian score (LS) [10]. This unsupervised technique is attractive due to the sparsity of labeled data available in our task scenario (i.e. relatively few feedback documents in comparison to the size of the feature space). LS is a filter-based method for feature selection, where we refer to the different dimensions of a feature vector as the feature set. Given m input features, LS scores each feature independently and selects the top N features, where N is a parameter. LS is based on Laplacian Eigenmaps and Locality Preserving Projection [2, 11]. The basic assumption of LS is that if two data points are close, they are likely to have the same class label. An effective feature set, therefore, should preserve such proximity for examples belonging to the same class. This is the same property exploited by our Local Structure method for example selection described earlier (§4). LS for the rth feature (LS_r) is computed according to $LS_r = \frac{1}{\sigma_{f_r}} \sum_{i,j} (f_{ri} - f_{rj})^2 S_{ij}$, where $\mathbf{f_r}$ is the rth feature for all examples, f_{ri} is the rth feature for example i, σ is the variance, and S is a sparse, inverse distance matrix over all examples in which only examples in close proximity have non-zero values. S thereby stores the local structure of the whole example space based on all features.

Let $\mathbf{x_i^{1:k}}$ denote the k nearest neighbors to feature vector $\mathbf{x_i}$, where k is a parameter. Non-zero entries in S are given by $(\forall \mathbf{x_i}, \forall \mathbf{x_j} \in \mathbf{x_i^{1:k}})\ S_{ij} = e^{-|\mathbf{x_i} - \mathbf{x_j}|}$, where $\mathbf{x_i}$ and $\mathbf{x_j}$ denote feature vectors for examples i and j along all dimensions.

6. EVALUATION METHODOLOGY

Evaluating iterative RF presents several different challenges from typical Cranfield-style evaluation of ad hoc and single-iteration RF. Acknowledging these challenges directly helps to ensure we establish a solid evaluation framework for measuring accuracy in this setting, comparing methods, and understanding limitations of evaluation methodology used.

6.1 Challenges

Feedback document selection vs. use. Relevance feedback methods can differ in how documents are selected for feedback as well as how those documents are utilized. Recent work in the TREC RF track has sought to isolate this confounding effect to better understand what makes one RF method more effective than another [4]. Our iterative feedback scenario makes it difficult to avoid this confound since each method produces a unique ranking at each iteration which the user then browses to provide feedback (§2).

Scarcity of relevant documents. Multiple RF methods are typically compared via *residual* evaluation in which feedback documents used by any of the methods being compared are excluded from evaluation (i.e. removed from both "gold standard" judgments and system rankings). This yields a single, common document collection for fair comparison of methods. With iterative feedback, ideally the residual collection would be determined by removing all feedback documents used by any method at any iteration considered. Once the residual collection was known, we could "go back in time" to evaluate accuracy of RF methods at earlier iterations.

The problem with this ideal strategy is scarcity of relevant documents in typical test collections available: using assessed documents for feedback often leaves relatively few for evaluation (especially relevant documents). Consequently, we sought an evaluation methodology that would allow us to meaningfully evaluate relevance feedback without the full exclusion required by traditional residual evaluation.

Another challenge due to scarcity of relevant documents lies in computing average system accuracy across topics. Recall our task scenario involves the user selecting a relevant document for feedback at each iteration (§2). Once the set of relevant documents is exhausted for a given topic, iterative RF terminates. However, if RF for a given topic terminates at iteration 8, how do we compute average system accuracy across topics at iteration 9? Do we (a) omit the query from the average or (b) include its accuracy as of iteration 8? Choice (a) would yield increasing error bars across iterations as progressively fewer topics contribute to the computed average, while choice (b) would dampen the learning effects (positive or negative) we are trying to observe. We adopt (a) with awareness of needing to monitor result instability at late iterations of feedback.

6.2 KeepAll and TakeOut Evaluation

To address the challenge of scarce relevant documents, we evaluate accuracy of methods in two distinct ways, which we denote as **KeepAll** and **TakeOut**. Both have strengths and weaknesses, and our general thought is that evaluating methods under both strategies in tandem provides a balanced view for comparing relative accuracy of methods.

KeepAll evaluation compares methods without removal of feedback documents. This means that even after a given document's relevance is made known to the system via user feedback, we continue to include that document in the evaluation set. However, we also follow prior work's strategy [9] of imposing the restriction that systems cannot memorize the list feedback documents, but instead must limit learning to parameter updates. Nonetheless ranking documents that were previously viewed still results in artificially inflated ranking accuracies observed across systems. On the positive side, *KeepAll* evaluation does preserve the expected trend of accuracies rising across systems with increasing feedback.

A possible problem with *KeepAll* in our task scenario is that it could lead to the same document being selected repeatedly for RF across iterations. While such repeated selection is actually a useful property in some learning models such as Boosting [21], it would be odd to the user to keep seeing the same feedback documents over-and-over again in the results list, let alone having to tell the system multiple times that a given document is relevant.

To address this, we distinguish two separate rankings: a *feedback ranking* and an *evaluation ranking*. The evaluation ranking is generated over all documents and is used to evaluate system accuracy. The feedback ranking is presented to the user for feedback and filters out all previous feedback documents from the evaluation ranking. This is similar to a search interface that lets users "bookmark" known relevant webpages in a sidebar and thereafter excludes them from the ranking. Since the user is never presented the same relevant document twice for feedback, RF iteration terminates for each topic once all of its relevant documents are exhausted.

TakeOut evaluation, unlike *KeepAll*, removes feedback documents from evaluation. We do not perform full *residual evaluation*, but rather evaluate each method on a unique residual document subset at each iteration based on feedback documents used up to the current iteration. This means different methods will typically be evaluated on different document subsets in the same iteration, though they will still have the same number of relative documents under the correct user feedback. Moreover, since progressively fewer relevant documents remain to be found as iteration progresses, topics become increasingly difficult. Relevant documents that are easier to find are taken out in earlier iterations, so remaining relevant documents in the pool tend to become progressively more difficult to find individually as well as fewer in number overall. Consequently, we observe a counter-intuitive effect whereby the general trend across systems is for ranking accuracy to decrease rather than increase as iteration proceeds. Nonetheless, we would expect in comparative evaluation of methods to see a stronger RF method to decrease in accuracy more slowly than other methods.

A final challenge with both *KeepAll* and *TakeOut* evaluation is specific to active learning (§4). Recall active learning selects a document whose relevance is maximally uncertain and slots it at the top rank. A practical consequence of evaluating ranking accuracy is that early precision metrics like MRR become entirely dominated by this single document, and so fail to provide any meaningful measure of the actual ranking. To address this artifact, we focus evaluation of active learning on the point at which the user's effort threshold is exceeded. The user then terminates the training phrase and only wants to use the system thereafter. This disables active learning and so eliminates these slotting effects. To model this, we evaluate system accuracy on the *evaluation ranking* (without slotting) rather than the *feedback ranking* (with slotting). User effort is still measured off the feedback ranking, ensuring that slotting any non-relevant documents increases measured user effort since the user must browse more results before finding a relevant document.

7. EVALUATION

Recall our interactive search task involves iterative back-and-forth between system ranking and user feedback (§2). The system goal in this scenario is to maximize ranking accuracy for any amount of relevance feedback provided. Consequently, our evaluation measures ranking accuracy achieved under varying amounts of user feedback in order to compute the *learning curve* of each relevance feedback method. Measuring this curve enables us to compare the relative strength of each method across a range of possible stopping points which might be encountered in practice and suggest different methods being better suited to different use cases. Our main results assume user feedback to the system is always correct, similar to a traditional RF setting with known relevant and non-relevant documents (§7.2). We also present results for a noisy feedback scenario (§7.3) and an alternative learning scenario on LETOR TD2003 and TD2004 collections (§7.4).

7.1 Experimental Setup

Experiments in §7.2 and §7.3 are reported on the TREC[3] Robust04 document collection of newswire articles. Topics 301-450 were used for evaluation during system development and tuning, with topics 601-700 held out for final testing. System queries are taken from title field of each topic. An initial ranking of 200 documents for each query is generated using Indri [23]. This initial ranking provides both a baseline measure of system accuracy as well as the document pool subsequently re-ranked by each RF method. To reduce vocabulary growth from rare words, we prune out any term which does not occur at least twice in some document and at least four times in the entire collection. This reduces the vocabulary size of the query-specific document pools from about 10,000-20,000 terms to about 3,000-4,000 terms.

We compare the relative performance of several RF baseline methods in this task setting of iterative feedback (§3): (1) Rocchio feedback [20] (denoted by "Rocchio"), (2) Rocchio with positive feedback only (denoted by "RocchioPos"), and (3) model-based feedback [28] (denoted by "Language"). Since prior work has only observed modest benefits from negative feedback, we were particularly interested in comparing the relative performance of (1) and (2) in our task setting. Additional details of baseline methods are discussed in §3. For all baselines, parameters are tuned on the development set to maximize the accuracy of each method: Rocchio ($\alpha = 0.05$, $\beta = 0.5$), RocchioPos ($\alpha = 0.05$, $\beta = 1$), and model-based (Language) ($\alpha = 0.05$, $\lambda = 0.8$, and $\gamma = 0.3$).

For active learning, we evaluate both **Simple Margin** and **Local Structure** criteria (§4). Simple Margin has no parameters, while with Local Structure we use $m = 10$ neighbors and $\alpha = 0.5$. LS feature selection (§5) further reduces vocabulary size for active learning using $k = 15$ neighbors and $N = 2500$ features (a modest further reduction from the already reduced vocabulary size). Feature selection is not used with baseline methods. While baselines benefit from interpolating the original query with feedback (an anchoring effect to prevent query drift), active learning methods perform ranking based on feedback documents only. Exploring anchoring between the query model and active learning via rank fusion [6] remains for future work.

Ranking accuracy is measured by three metrics: mean-average precision (MAP), top-10 precision (P@10), and mean

[3]http://trec.nist.gov

	MAP	P10	MRR	MAP_All	P@10_All
RocchioPos	0.3296 ($p < 0.001$)	0.2554 ($p < 0.001$)	0.5437 ($p < 0.001$)	0.6268 ($p < 0.001$)	0.6423 ($p < 0.001$)
Rocchio	0.3449 ($p < 0.001$)	0.2755 ($p < 0.001$)	0.5848 ($p < 0.001$)	0.6846 ($p < 0.001$)	0.6987 ($p < 0.001$)
Language	0.3923 ($p \approx 0.24$)	0.2918 ($p \approx 0.019$)	0.6028 ($p \approx 0.016$)	0.6730 ($p < 0.001$)	0.6754 ($p < 0.001$)
ActiveMargin	0.3952†	0.3116‡	**0.6251‡**	0.7427‡	0.7399‡
ActiveStructure	**0.3978†**	**0.3135‡**	0.6248‡	**0.7475‡***	**0.7525‡**

Table 1: Average accuracy of each method across feedback iterations. Statistical significance is computed by two-sided paired t-test [22]. Significance levels p for ActiveMargin vs. each baseline are included in each baseline's cell. † denotes significant improvement ($p < 0.05$) over at least one baseline, ‡ a significant improvement over all baselines, and * a significant improvement of Local Structure over Simple Margin. MAP_All and P@10_All correspond to *KeepAll* evaluation; other measures correspond to *TakeOut* evaluation.

reciprocal rank (MRR). User effort is measured by the total number of clicks, which equals the number of positive documents with correct feedback and upper-bounds the number of positive documents with noisy feedback. We also tracked the total number of documents viewed by the user in browsing search results but saw the number of clicks and views were very correlated and largely yielded the same learning curve plots. Consequently, we omit these results to simplify presentation. Another related metric we did not measure is Cooper's expected search length: the number of non-relevant documents a user needs to skip over to find some number of relevant documents desired [5].

7.2 Correct Relevance Feedback

Figure 3 presents empirical results for both *TakeOut* and *KeepAll* evaluation conditions (see §6.2) assuming relevance feedback provided by the user is always correct. We also compute the average accuracy of each method across iterations and present this in Table 1, along with results of statistical significance tests. Statistical significance of improvements is determined using a two-sided paired t-test [22] at the 5% significance level over all queries. The proposed active learning methods (Simple Margin and Local Structure) are seen to by-and-large dominate baselines across iterations, metrics, and for both *TakeOut* and *KeepAll* conditions. We omit the plot of MRR due to space limitations, though we do report it for *TakeOut* evaluation in Table 1. MRR is not meaningful for *KeepAll* since the labeled documents are essentially always ranked at the top.

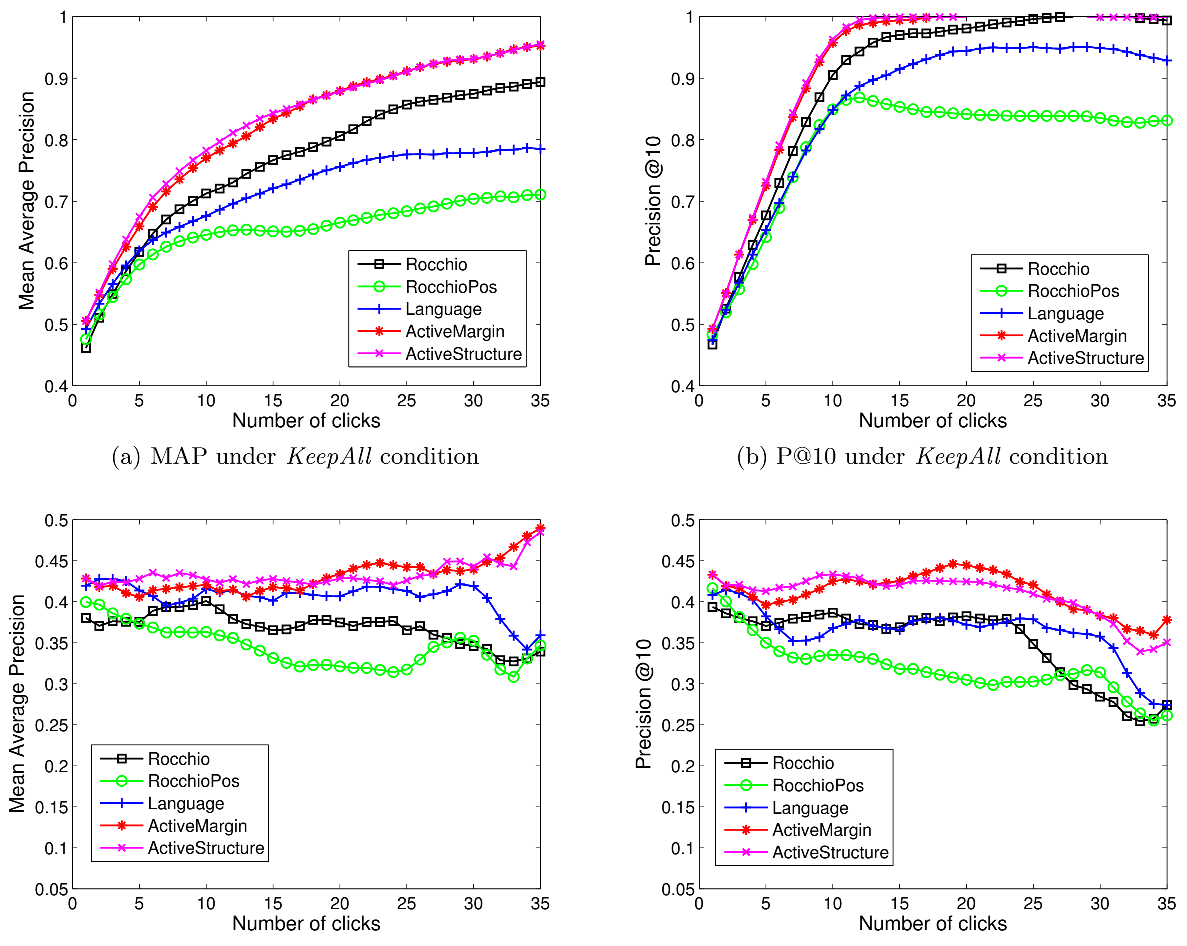
As mentioned earlier (§6), an experimental challenge is that while we expect system accuracy to improve with increasing feedback, we have a confounding factor that tends to work against this: as more relevant documents are "used up" for feedback, fewer remain to be found in subsequent iterations, effectively increasing query difficult. Moreover, documents that are "easier" to find are taken out in earlier iterations, so remaining relevant documents in the pool become progressively more difficult to find individually as well as fewer in number overall. This effect is particularly pronounced in the *TakeOut* condition in which system accuracies tend to decrease due to fewer relevant documents remaining in the residual collection used for evaluation.

With the *KeepAll* condition, *all* documents are reranked and included in evaluation at each iteration. As noted earlier (see §6.2), this system does not "remember" earlier feedback documents and must rerank them like any other document based on learned parameters. However, it is still the case that the system has learned from having seen some of the relevant documents that are now being evaluated on, which tends to introduce the opposite evaluation effect that system accuracy tends to be inflated across methods with increasing feedback. Allowing for this effect, Figure 3(a) shows that under *KeepAll* evaluation, all feedback methods are seen to exhibit the expected (and more intuitive) behavior of achieving increased accuracy with greater feedback. While *TakeOut* and *KeepAll* evaluation conditions each have their own respective limitations, the key point to observe with regard to our evaluation of active learning is that it dominates the other baselines across both methods of evaluation.

Another interesting point to note is the observed value of negative feedback in our results vs. what has traditionally been observed in ad hoc search or single-iteration relevance feedback. The traditional wisdom has been that negative documents have relatively little in common, so learning what one looks like does not help the system avoid ranking another one highly. In our experiments, three methods use both positive and negative feedback: Rocchio, Simple Margin, and Local Structure. RocchioPos and Language only use positive feedback. Of particular note, Rocchio outperforms RocchioPos on nearly all metrics in all iterations. Both active learning methods dominate all baselines, including all methods using only positive feedback, in MAP and P@10. One possible explanation for our findings is that our user model and method for obtaining relevance feedback tend to produce many more non-relevant documents than relevant documents for feedback, thus *en masse*, many examples of non-relevant documents do help to characterize the overall space of non-relevance. Another possible effect we are observing is due the entire document pool coming from the initial Indri ranking, thus the set of non-relevant documents used for feedback might otherwise be highly ranked by the system. The TREC Relevance Feedback track [4] has investigated issues of feedback document selection in recent years, though the question remains open as to what constitutes a useful non-relevant document for negative feedback [26].

7.3 Noisy Relevance Feedback

This section reports relative accuracy achieved by different RF methods under noisy feedback, when the user clicks on non-relevant documents or fails to click on relevant ones. As discussed earlier (§2), noisy feedback is modeled using two Bernoulli parameters, f_p and f_n, to control the false positive and negative rates, respectively. Setting both parameters to zero yields the same correct feedback setting evaluated above. We evaluate methods for three noise conditions: (1) $f_p = 0$ and $f_n = 0.1$, (2) $f_p = 0.1$ and $f_n = 0$, and (3) $f_p = 0.1$ and $f_n = 0.1$. Results of *KeepAll* evaluation are shown in Figure 5.



(a) MAP under *KeepAll* condition (b) P@10 under *KeepAll* condition

(c) MAP under *TakeOut* condition (d) P@10 under *TakeOut* condition

Figure 3: Learning curve plots depict ranking accuracy on Robust04 of the different feedback methods given a varying number of correct relevance feedback documents provided by the user. The horizontal axis indicates the number of feedback iterations (and by this same token, the number of positive feedback documents used).

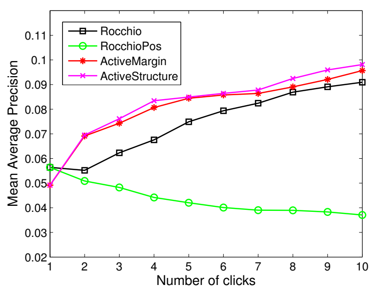
While all methods are seen to decrease in accuracy in comparison to the noiseless condition, active learning methods continue to dominate the other baselines. However, results with noise rates of 20% and higher have shown Rocchio to dominate active learning, suggesting a threshold for method selection based upon the level of noise. Our hypothesis for this behavior is that active learning, which updates the decision surface based on feedback, is inherently more sensitive than Rocchio, which simply updates its linear combination of feedback documents. Thus the same virtue which enables active learning to learn more efficiently from correct feedback appears to render it less robust to noisy feedback.

7.4 Relevance Feedback on LETOR

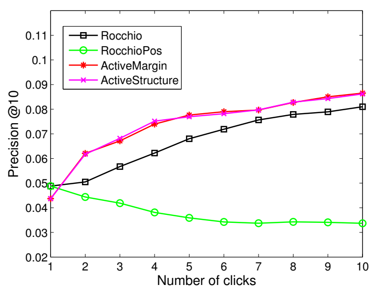
This section presents experimental results conducted on LETOR 3.0 [15]. LETOR defines a standard feature representation of documents for supporting machine learning studies, enabling easier comparison of alternative learning strategies for the same features. In comparison to the term-based features used in our earlier experiments on Robust04, LETOR features are quite different and potentially more powerful, providing another useful benchmark for assessing the relative effectiveness of our active learning methods.

While LETOR includes several document collections, not all are suitable for our study. In particular, because we are interested in RF (which requires many relevant documents), we exclude OHSUMED, as well as Homepage or Named Page Finding collections. Instead, we focus on Topic Distillation: TD2003 and TD2004. These collections are derived from the 2003 and 2004 TREC Web track [7] tasks on the .GOV collection (a 1.25 million web crawl of the .gov domain performed in 2002). No dimensionality reduction is performed.

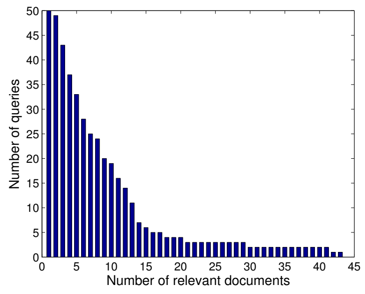
For LETOR, we do not have the query, so model-based feedback [28] cannot be evaluated. For other methods, we use two positive and two negative documents to form the query for Rocchio, and to initialize the active learning models. Active learning parameters are identical to those used in our earlier experiments on Robust04 with one notable exception: we restrict the ratio of non-relevant feedback to relevant feedback at a heuristic 7:1 ratio. That is, we ignore additional non-relevant documents provided as feedback whenever it would exceed this threshold. While TD2003 and



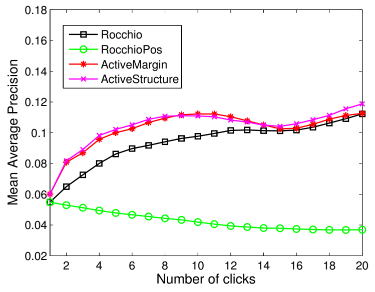
(a) MAP (TD2003)



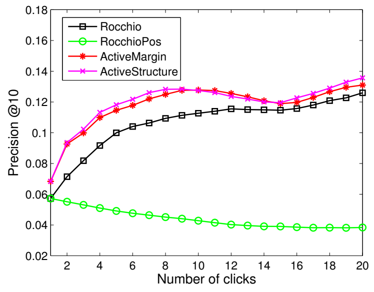
(b) P10 (TD2003)



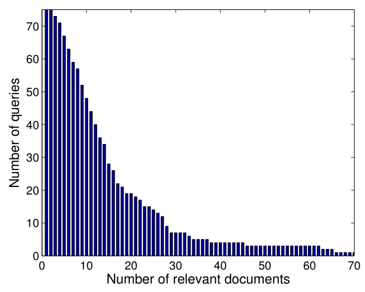
(c) Histogram of number of relevant documents per topic (TD2003)



(d) MAP (TD2004)



(e) P10 (TD2004)



(f) Histogram of number of relevant documents per topic (TD2004)

Figure 4: Learning curves of RF methods on LETOR's TD2003 (first row) and TD2004 (second row) collections depict ranking accuracy vs. amount of relevance feedback provided. Accuracy levels reflect training on a *single* topic and testing on the remaining topics, repeated round-robin for cross-validation. The horizontal axis indicates the number of feedback iterations (equal to the number of positive feedback documents used). Active learning methods perform as well or better than Rocchio. As the histogram shows, relevant documents per topic are quite sparse, restricting the number of feedback iterations possible with stable evaluation.

TD2004 have more relevant documents per topic than other LETOR collections, relevant documents remain relatively sparse. We found that an excess of non-relevant documents hurt active learning accuracy. Parameters are tuned on TD2003, and tested on TD2004.

A significant difference between our evaluation on LETOR vs. Robust04 stems from the differing nature of features in the two collections. With term-based features (Robust04), RF let us learn term-based weights to improve ranking accuracy for the current query. For example, while the word "dog" might be very important to the current query, RF on this query cannot tell us anything about how important this word should be for other queries. With LETOR, in contrast, features capture general properties of query-document compatibility. This means, for example, that if the PageRank feature is important for the current query, there is a good chance it will be important for other queries as well.

Due to this difference in features, we can simplify our evaluation methodology enormously by adopting a slightly altered task scenario. In this new scenario, training and testing phases are completely distinct: the user trains the system via iterative feedback on *one* topic, then uses the system to rank documents for all other topics. Because the user cannot provide iterative feedback on multiple queries in parallel, we abandon LETOR's standard 5-fold partition of topics. Instead, for each topic we train on it alone and evaluate on all other topics, then we average this over all topics. Since we are only training on a single topic, resulting accuracy is far lower than what typically published LETOR results. As with our first batch of results presented for Robust04, we again assume the user provides correct feedback.

This revised task means that the user no longer cares about finding relevant documents for the current topic. However, he stills terminates iteration whenever either (a) the trained system is "good enough", (b) his effort threshold in training the system is exceeded, or (c) no additional relevant documents remain for feedback. As in earlier experiments on Robust04, once (c) occurs, we exclude the given topic from average accuracy calculations in subsequent iterations. This means that learning curve plots reflect an average over progressively fewer topics as the number of relevant documents is exhausted. As the average is computed over progressively fewer topics, stability decreases, especially at late iterations.

Consequently, we only present results for 10 iterations with TD2003 and 20 iterations with TD2004. These termination points stem from the corresponding histograms for TD2003 and TD2004 indicating number of relevant documents per query (Figures 4(c) and Figure 4(f), respectively). We stop iteration when the number of topics remaining would fall below 20 (a loose threshold for result stability from observation). While the histogram for the Robust04 collection is not shown, the choice of terminating after 35 iterations there is the same: there would be less than 20 topics with any relevant documents left to be found.

Results in Figure 4 show that active learning methods perform as well or better than Rocchio.

8. CONCLUSION AND FUTURE WORK

This paper considers an interactive IR scenario in which the user is interested in finding several to many relevant documents with minimal effort. Given an initial document ranking, the user interacts with the system to provide iterative relevance feedback. Evaluation on the TREC Robust04 collection and LETOR TD2003 and TD2004 collections shows our active learning approach dominates several standard baselines in terms of effectiveness achieved relative to the amount of user effort required.

Future work will continue to develop our active learning methods, evaluate on additional document collections, compare stability of parameters across collections, and evaluate noisy feedback using real user data. We are also interested in modeling and evaluating active learning for diversity ranking [16, 18, 25] with interactive retrieval.

Investigation of the exploration vs. exploitation learning tradeoff represents another interesting direction for future work: varying the rank at which to slot-in an uncertain document for feedback, as well as potentially varying the number of uncertain documents used. While a lower ranking of uncertain documents will increase ranking accuracy at the current iteration, doing so may reduce accuracy at future iterations by increasing the probability of a different relevant document occurring at a higher rank. In that event, the user would never see the uncertain document (under assumptions of the current user model), thereby failing to label the example selected by active learning.

Acknowledgments

We would like to thank the anonymous reviewers for their valuable feedback. This work was partially supported by a John P. Commons Fellowship for the second author.

9. REFERENCES

[1] B. Bai, J. Weston, D. Grangier, R. Collobert, K. Sadamasa, Y. Qi, O. Chapelle, and K. Weinberger. Learning to rank with (a lot of) word features. *Information Retrieval*, 13(3):291–314, 2010.

[2] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *NIPS*, 1:585–592, 2002.

[3] C. Brandt, T. Joachims, Y. Yue, and J. Bank. Dynamic ranked retrieval. In *Proc. WSDM*, pages 247–256. ACM, 2011.

[4] C. Buckley and S. Robertson. Relevance feedback track overview. In *17th TREC Notebook*, 2008.

[5] W. Cooper. Expected search length: A single measure of retrieval effectiveness based on the weak ordering action of retrieval systems. *American Documentation*, 19(1):30–41, 1968.

[6] G. Cormack, C. Clarke, and S. Buttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *SIGIR*, pages 758–759, 2009.

[7] N. Craswell and D. Hawking. Overview of the TREC-2004 Web track. In *Proc. TREC*, 2005.

[8] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

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

[10] X. He, D. Cai, and P. Niyogi. Laplacian score for feature selection. *NIPS*, 18:507, 2006.

[11] X. He and P. Niyogi. Locality preserving projections. *NIPS*, 16:153, 2004.

[12] T. Hofmann. Probabilistic latent semantic indexing. In *Proc. SIGIR*, pages 50–57. ACM, 1999.

[13] A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: guiding supervised learning with decision-theoretic active learning. In *IJCAI*, pages 877–882, 2007.

[14] P. Over. The TREC interactive track: an annotated bibliography. *Information Processing & Management*, 37(3):369–381, 2001.

[15] T. Qin, T. Liu, J. Xu, and H. Li. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, pages 1–29, 2010.

[16] F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *Proc. SIGIR*, pages 691–692. ACM, 2006.

[17] F. Radlinski and T. Joachims. Active exploration for learning rankings from clickthrough data. In *Proc. SIGKDD*, pages 570–579. ACM, 2007.

[18] D. Rafiei, K. Bharat, and A. Shukla. Diversifying web search results. In *WWW*, pages 781–790. ACM, 2010.

[19] S. Robertson. The probability ranking principle in IR. *Journal of Documentation*, 33(4):294–304, 1977.

[20] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American Society for Information Science*, 41(4):288–297, 1990.

[21] R. Schapire. A brief introduction to boosting. In *Proc. IJCAI*, volume 16, pages 1401–1406, 1999.

[22] M. D. Smucker, J. Allan, and B. Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proc. of CIKM*, pages 623–632, 2007.

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

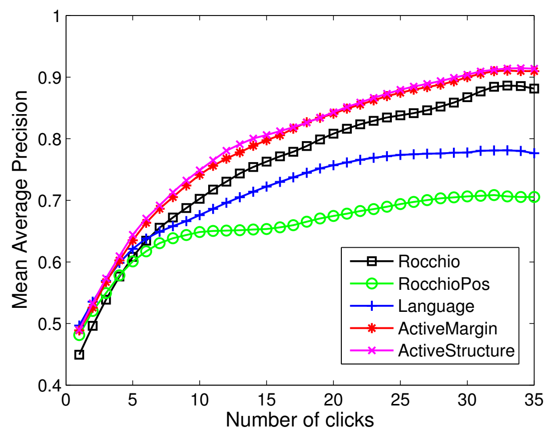
[24] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. *The Journal of Machine Learning Research*, 2:45–66, 2002.

[25] J. Wang and J. Zhu. Portfolio theory of information retrieval. In *Proc. SIGIR*, pages 115–122. ACM, 2009.

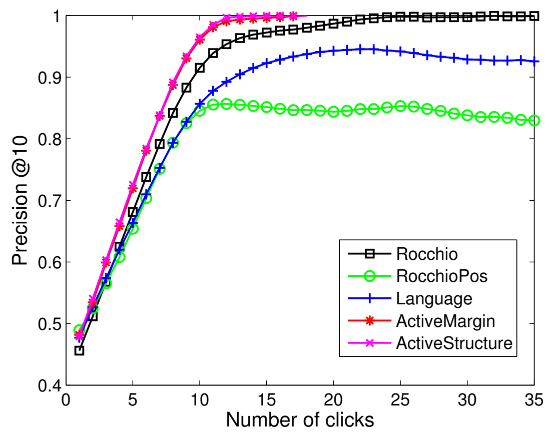
[26] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *Proc. SIGIR*, pages 219–226. ACM, 2008.

[27] X. Wei and W. Croft. LDA-based document models for ad-hoc retrieval. In *SIGIR*, pages 178–185, 2006.

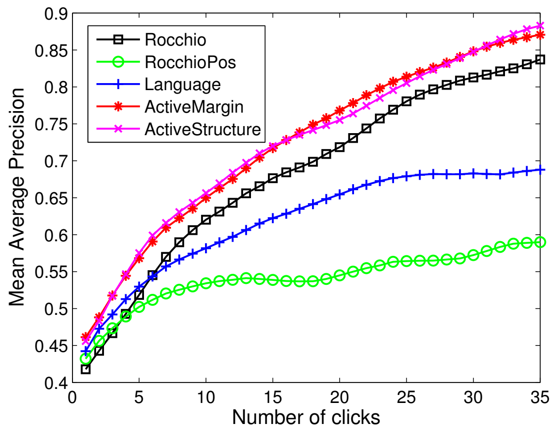
[28] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proc. CIKM*, pages 403–410. ACM, 2001.



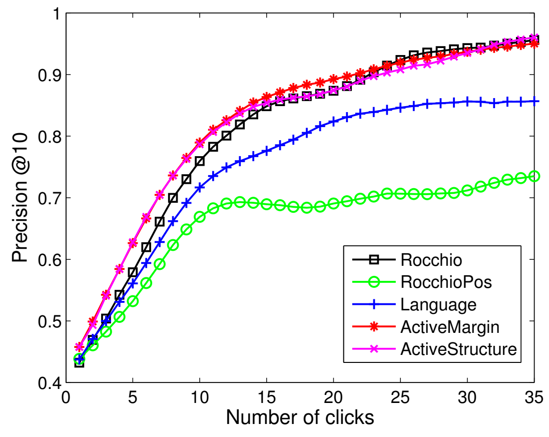
(a) MAP: $f_p = 0$ and $f_n = 0.1$



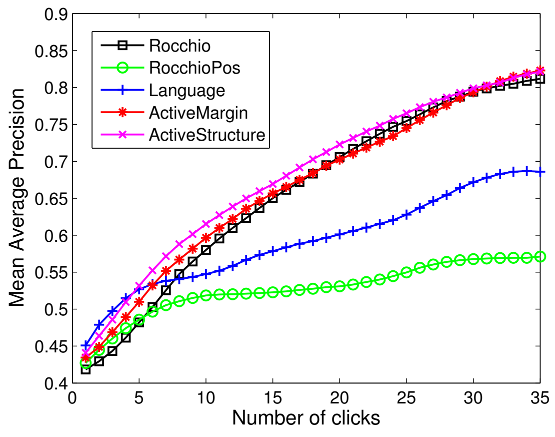
(b) P@10: $f_p = 0$ and $f_n = 0.1$



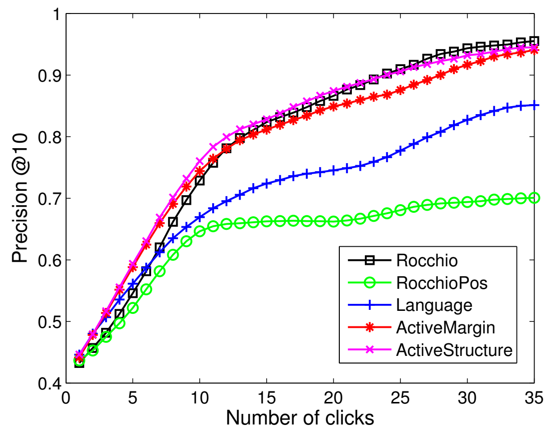
(c) MAP: $f_p = 0.1$ and $f_n = 0$



(d) P@10: $f_p = 0.1$ and $f_n = 0$



(e) MAP: $f_p = 0.1$ and $f_n = 0.1$



(f) P@10: $f_p = 0.1$ and $f_n = 0.1$

Figure 5: Learning curves for the five relevance feedback methods given noisy user feedback. f_p and f_n denote Bernoulli probabilities of false positives (erroneous clicks) and false negatives (missed relevant documents) by the user, respectively. Evaluation is performed under *KeepAll* condition. The horizontal axis indicates the number of feedback iterations (and by this same token, the number of user clicks). In comparison to earlier results with correct feedback, we see the introduction of noise decreases accuracy, as expected. Active learning methods Simple Margin and Local Structure continue to dominate the baseline methods.