Inf Retrieval (2013) 16:307–330
DOI 10.1007/s10791-012-9198-8

A learning approach to optimizing exploration–exploitation tradeoff in relevance feedback

Maryam Karimzadehgan · ChengXiang Zhai

Received: 7 October 2011 / Accepted: 9 March 2012 / Published online: 30 March 2012

Abstract Relevance feedback is an effective technique for improving search accuracy in interactive information retrieval. In this paper, we study an interesting optimization problem in interactive feedback that aims at optimizing the tradeoff between presenting search results with the highest immediate utility to a user (but not necessarily most useful for collecting feedback information) and presenting search results with the best potential for collecting useful feedback information (but not necessarily the most useful documents from a user's perspective). Optimizing such an exploration–exploitation tradeoff is key to the optimization of the overall utility of relevance feedback to a user in the entire session of relevance feedback. We formally frame this tradeoff as a problem of optimizing the diversification of search results since relevance judgments on more diversified results have been shown to be more useful for relevance feedback. We propose a machine learning approach to adaptively optimizing the diversification of search results for each query so as to optimize the overall utility in an entire session. Experiment results on three representative retrieval test collections show that the proposed learning approach can effectively optimize the exploration–exploitation tradeoff and outperforms the traditional relevance feedback approach which only does exploitation without exploration.

Keywords Interactive retrieval models · Feedback · Diversification · User modeling

1 Introduction

Relevance feedback has proven to be very effective for improving retrieval performance [25, 42, 55, 56, 58, 79]. The idea is to have a user to provide relevance judgments on some initial search results shown to the user, a retrieval system can then use the collected

M. Karimzadehgan (✉) · C. Zhai
Department of Computer Science, University of Illinois at Urbana-Champaign,
Urbana, IL 61801, USA
e-mail: mkarimz2@illinois.edu

C. Zhai
e-mail: czhai@cs.illinois.edu

relevance judgments as feedback information to better infer the user's information need and improve the ranking of the unseen results for the user. For example, a user may view the first page of search results and make judgments, and the system can then leverage the feedback information to improve the ranking of the results on the next few pages for the user.

A lot of research has been done on relevance feedback. However, most existing work is focused on developing and improving relevance feedback methods (e.g., [56, 58, 65, 79]), where the goal is to optimize the ranking of results based on a given set of relevance judgments. Typically, the relevance judgments are assumed to be collected on some top-ranked documents that would be shown to a user as initial results. Although this strategy of relevance feedback is very natural as a user can judge documents while viewing search results in a normal way, the judgments collected in this way (i.e., judgments on the top-ranked documents) may not necessarily be the most useful judgments for relevance feedback. As an extreme case, if the top documents all have very similar contents, the judgments on all these documents would be clearly redundant, and would not be so useful for relevance feedback as if we collect the same number of judgments on a more diversified set of documents. Clearly, in such an extreme case, the benefit brought by relevance feedback to the user in the next iteration of the interactive retrieval process would be relatively small, at least not as much as the user would have if the judgments were collected on a more diverse set of documents. This shows a deficiency of the traditional (standard) relevance feedback strategy: *it does not attempt to obtain the most useful judgments*.

This observation has motivated some researchers to study active feedback methods [61, 73, 74], where the goal is to choose documents for relevance feedback so that the system can learn most from the feedback information. The idea of these methods is usually to choose a set of diverse documents for judgments since judgments on diverse documents can be expected to be more informative and thus helpful for the system to learn the user's information need. For example, in [61], the top-ranked documents are first clustered and only one document is selected from each cluster to form a diverse set of documents to present to the user. However, presenting diverse documents to users means that we generally do not present the search results in the order of relevance. Thus, the utility of the presented documents to the user is generally lower than that of presenting the top-ranked documents. This shows that active feedback has a different deficiency: *it does not attempt to optimize the utility of the presented documents to the user*.

These observations illustrate an interesting dilemma in interactive relevance feedback: On the one hand, we want to present the top-ranked documents to a user so that the presented documents would be most useful for the user in this interaction cycle as in standard relevance feedback. However, such a strategy does not generate the most useful judgments for feedback. On the other hand, if we diversify the results in order to obtain more useful judgments, which would bring more utility to the user in the next interaction cycle, we would risk on decreasing the utility to the user in the current interaction cycle. We refer to this dilemma as *exploration–exploitation tradeoff* [40, 66].

Intuitively, if we present top ranked results to the user, it is very useful for the user since there is no discount compromise of the utility, which can be regarded as emphasizing *exploitation* of all the existing information about the user's information need, but it does not help the system to *explore* the space of all the potentially relevant documents to the user. On the other hand, if we show diversified results (i.e., emphasizing *exploration*), there is a concern of decreasing the utility from the user perspective, thus possibly compromising *exploitation*.

Clearly, in an interactive retrieval system, what matters to a user is the *overall* utility of relevance feedback, which means that we need to optimize this exploration–exploration tradeoff so as to optimize the overall utility over an interaction session[1] which includes both the current interaction cycle and the next interaction cycle after feedback. To the best of our knowledge, no existing work has addressed the problem of optimizing the exploration–exploitation tradeoff for relevance feedback. Indeed, the standard relevance feedback work is focused on exploitation without considering exploration, while the active feedback work does the opposite—focused on exploration without considering exploitation; each can be regarded as optimizing feedback for just one interaction cycle. This tradeoff problem touches a more general issue of how to optimize the utility of a retrieval system over an interaction session for a user [21].

In this paper, we study how to optimize the exploration–exploitation tradeoff in relevance feedback based on the following interactive process: A user is assumed to have a high-recall information need. After issuing a query, the user is assumed to sequentially view results up to N top-ranked results. We further assume that in this process, the user would need to fetch unseen results at least twice due to the limited number of results that can be displayed on one screen, resulting in two natural time points when a system can learn from the feedback information collected so far to optimize the unseen results. One instance of this scenario is that a user views three pages of search results; in such a case, the system can learn from the feedback information collected from the first page of results to optimize the results shown on the second page right before the user navigates into the second page, and can also do the same for the third page when the user reaches it. While most users of a Web search engine do not go this far when viewing results, we would expect them to do so when the search results are displayed on a much smaller screen such as on a smartphone. Also, a user often needs to scroll down on the first page to view all the results; in such case, we may also view the "scrolling" as to fetch more unseen results, thus a point when the system can re-rank the unseen results. For convenience of discussion, we will refer to the very first batch of results seen by the user as the results on the first "page"(or segment), the results seen by the user through either scrolling or clicking on a next-page button as the results on the second "page" (or segment) and so on so forth.

Since more diversified results would mean more exploration and thus less exploitation, while less diversified would mean more exploitation and less exploration, we essentially convert this tradeoff problem to one of optimizing this diversity parameter. We propose and study two methods to optimize this diversity parameter. The first one assumes a fixed optimal value for all the queries (fixed-coefficient diversification), and the second learns a query-specific optimal value for each query (i.e., adaptive diversification). We evaluate our proposed methods on three representative TREC data sets. The experimental results demonstrate that both of our proposed methods can effectively optimize exploration–exploitation tradeoff and achieve higher utility for the session than both traditional relevance feedback which ignores *exploration* and pure active feedback which ignores *exploitation*. Moreover, the *adaptive diversification* method is better than the fixed-coefficient diversification method due to its optimization of the tradeoff at a per-query basis.

The rest of the paper is organized as follows: We discuss related work in Sect. 2. The problem formulation is given in Sect. 3. We then present our work on learning to optimize diversification in Sect. 5. We discuss experiment design and experiment results in Sects. 6 and 7, respectively. Finally, we conclude in Sect. 8.

[1] We define **session** as viewing multiple search results for the same user query through interacting with the system (e.g., clicking on buttons).

2 Related work

Relevance feedback has been shown to be effective with different kinds of retrieval models (e.g., [42, 56, 58, 55, 79]). In vector space model, feedback is done with Rocchio algorithm. In language modeling, relevance feedback can be implemented through estimating a query language model [65, 79] or relevance model [42]. These methods only consider "exploitation", and our study is orthogonal to optimization of these relevance feedback algorithms. We used a mixture model feedback method [79], but our idea is general and can potentially work for other feedback methods too.

Active feedback is essentially an application of active learning which has been extensively studied in machine learning (e.g. [15, 57]). Active learning has been applied to text categorization (e.g. [44, 46, 67]) and information filtering [82] where the authors considered the value of longer-term exploration along with the immediate reward of delivering a document when setting decision boundaries. Our work is related to this work as we maximize the tradeoff between exploration and exploitation, however their application scenario is different than ours where the decision to be optimized is whether to deliver a document to a user whereas in our problem, the decision to be optimized is how much diversification we should impose. Recently, it has also been applied to ad hoc retrieval [33] and relevance feedback [61, 73, 74]. All these active feedback methods emphasize on "exploration" only, and are also orthogonal to our study. We adopted the diversification strategy for active feedback, but the proposed methods are also potentially applicable to other methods for active feedback.

Multi-arm bandit models an agent that simultaneously attempts to acquire new knowledge and to optimize its decisions based on existing knowledge. Multi-arm bandit methods that seek to find the optimal tradeoff between exploration and exploitation have been studied extensively (e.g., [49, 54, 77, 48]). Existing solutions to the standard bandit problem assume a fixed set of arms with no delayed feedback. However, recently, the work in [2] has considered such a tradeoff to maximize total clicks on a web content. Also, reinforcement learning has been used for solving sequential decision making problems, which assumes there exists an agent interacting with the environment with the goal of maximizing the total reward. There have been extensive applications adopting reinforcement learning (e.g. [1, 43, 62, 36, 81]). Our work is similar to these works in that we also maximize the total utility by optimizing the tradeoff between exploration and exploitation, but we explore a new application in optimizing interactive relevance feedback.

Our work is also related to previous work in diversifying search results. Goffman [24] recognized that the relevance of a document must be determined w.r.t documents appearing before it. Several researchers have been working on methods to eliminate the redundancy in the result sets (e.g. [4, 8, 12, 78, 84]) by sequentially selecting documents that are relevant to the query but dissimilar to documents ranked above them or by introducing an evaluation framework that rewards novelty and diversity and penalizes redundancy [13]. Some other approaches [10, 76] maximize diversity (topic coverage) among a set of documents without regard for redundancy [10] or by learning the importance of individual words and then selecting the optimal set of results that cover the largest number of words [76]. Some other work on diversification make use of taxonomy for classifying queries and documents and create a diverse set of results according to this taxonomy [3, 68, 85]. Generating *related* queries and taking the results from each of them for re-ranking purposes [52] is another way for diversification. In addition, some other work, consider diversity in the area of spatial information retrieval [64] or in image retrieval [50, 59] or for query diversification [14]. All these studies consider diversity in

terms of providing a complete picture of different aspects of the query. However, we consider diversity in terms of providing more information for optimizing the utility of relevance feedback over an interactive retrieval session. We used the method in [78] however, our approach can be applied to any diversification method where there is a novelty parameter.

Learning to rank has recently attracted much attention (e.g. [6, 7, 31, 35, 53]). Most of the work in this line has not considered diversity. In [54] an online learning algorithm that directly learns a diverse ranking of documents based on user's clicking behavior to minimize abandonment (maximizing clickthrough) for a single query is proposed. While abandonment is minimized, their approach cannot generalize to new queries. The difference between our work and theirs is that they optimize the tradeoff for multiple users and a single query, while we optimize such a tradeoff for a single user over multiple interaction cycles (i.e., over multiple pages). Logistic regression [27] is widely used to learn a retrieval function to rank documents directly [11, 17, 22, 74, 83, 23]. Our work uses logistic regression to learn the diversity parameter.

The study of difficult queries has attracted much attention recently especially with the launch of ROBUST track in TREC conference which aims at studying the robustness of retrieval models [69, 70]. However, the most effective retrieval models that are developed by ROBUST participants relied on external resources (mostly Web) to perform query expansion which has bypassed the difficulty of the problem in reality. Because there are often no such external resources to exploit and indeed the web resources would not help improve the search accuracy for difficult queries on the Web itself. There has been some work on understanding why a query is difficult [5, 9, 26] or identifying difficult queries [70] or on predicating query performance [18, 28, 30, 34, 60, 75]. The work in [38, 71] are the first studies of negative relevance feedback that exploit the top non-relevant documents to improve the ranking of documents for difficult queries. Our work can also considered as a work to predict query difficult but we go beyond just predicting query difficulty by optimizing the way a system responds to a difficult query, i.e., optimizing the diversification so as to maximize the utility over the entire session.

Interactive search is another related line to our work [21, 43]. The author in [21] has proposed the holistic model to determine what is the best next action the system should perform when considering past interactions. However, none of these works has provided a technique to optimize the exploration–exploitation tradeoff.

The basic idea of our work has been published in a short paper of CIKM [37]. The extensions to this short paper made in the current paper include: detailed experiment results with more findings, review of all the technical methods used in the paper and comprehensive study of related work.

3 Problem formulation

Given a query Q and a document collection $\mathcal{C}$, we consider an interactive retrieval system where we assume a user would "progressively" view three segments of results through either scrolling or clicking on next-page button. The number of results in each segment is not necessary equal. For example, S_1 can be the top 5 results seen by a user in the beginning (the screen size does not allow the user to see all the 10 results), S_2 can be the bottom 5 results on the first page, and S_3 can be the next 10 results on the second page. The system first uses the query to retrieve k documents $S_1 = (d_{11}, \ldots, d_{1k})$ and present them on the first segment S_1. The user is then assumed to click on any relevant documents on this

segment, leading to a set of relevance judgments on the first segment, J_1. We assume that the user would continue viewing the second segment of search results, and at least some results on the third segment. We will focus on studying relevance feedback that happens after the user finishes viewing all the results on the second segment.

Our goal is to study how to use all the information collected after the user views the first segment (i.e., judgments J_1) to *optimally interact with a user on the second segment* so as to optimize the overall utility over the session of the user interacting with all the three segments. Specifically, we would like to optimize the diversification of the results on the second segment $S_2 = (d_{21}, \ldots, d_{2m})$ so that the relevance feedback that happens just before the user views the results on the third segment would deliver the maximum overall utility on both the second segment and the third segment. (The utility on the first segment has already been fixed in this setup, thus it is irrelevant for the optimization.) As discussed before, more diversification leads to more exploration and less exploitation, thus optimizing the diversification is a way to optimize exploration–exploitation tradeoff.

A common way to generate diversified results is a greedy algorithm called MMR [8] which models both topical relevance and redundancy of documents. With this strategy, after we have already selected documents $d_{21}, \ldots, d_{2,i-1}$ for the second segment, document d_{2i} would be selected to maximize the following score:

$$s(d_{2i}; d_{21}, \ldots, d_{2,i-1}) = (1 - \lambda) S_R(d_{2i}, Q) + \lambda S_{novelty}(d_{2i}; d_{21}, \ldots, d_{2,i-1})$$

where Q is the query, $S_R(d_{2i},Q)$ is a relevance-based scoring function (i.e., regular retrieval function). $S_{novelty}$ is the novelty value of d_{2i} w.r.t. $\{d_{21}, \ldots, d_{2,i-1}\}$, and $\lambda \in [0, 1]$ is the diversification parameter that controls the degree of diversification, or equivalently, the amount of exploration.

The results on the third segment $S_3 = (d_{31}, \ldots, d_{3p})$ would be generated using relevance feedback on the judgments collected on the second segment. Clearly, the utility of this third segment would be affected by the usefulness of the collected judgments J_2 which further depend on the diversification on the second segment. Our goal is to optimize the diversification coefficient, i.e., λ, to maximize the *overall utility* of the results on the second segment and the third segment, i.e., to optimize the exploration–exploitation tradeoff. Formally, we assume that there is a function A that can map a query Q and its corresponding relevance judgments J_1 on the first segment to the optimal diversification coefficient, i.e., $\lambda = A(Q, J_1)$. Later, we will propose two different ways to compute $A(Q, J_1)$. One way is simply to search for an optimal λ over a set of training queries and take this fixed coefficient as the optimal value for all the unseen test queries. The other way is to use a machine learning method to learn a potentially different optimal λ for each query.

Since our main goal is to study different methods for optimizing the diversification coefficient, in our experiments, we assume $|S_1| = |S_2| = |S_3| = 10$ to simulate a scenario when the user views the first three pages of results and we refer to S_1 as first-page result, to S_2 as second-page result and S_3 as third-page result. The proposed method does not depend on this configuration, though, and can be applied to other scenarios as well.

In the next section, we describe the methods used throughout the paper in more details.

4 Background

We now present all the components, i.e., Basic Retrieval Model, Feedback Methods and Novelty Method in this problem setup (Sect. 3) in more detail.

4.1 Basic retrieval model

We use the Kullback-Leibler divergence retrieval model [41] as our basic retrieval model for ranking documents on all the three pages (on the second page, it is used together with a novelty measure function to diversify results). This model is a generalization of the query likelihood retrieval model [51] and would score a document D w.r.t query Q based on the negative Kullback-Leibler divergence between the query language model θ_Q and the document language model θ_D:

$$S_R(D, Q) = -D(\theta_Q||\theta_D) = -\sum_{w \in V} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)}$$

where V is the words in the vocabulary. The document language model is usually smoothed using Dirichlet prior smoothing which is an effective smoothing method [80]. The query language model, is often estimated (in case of no feedback) based on $p(w|\theta_Q) = \frac{c(w,Q)}{|Q|}$, where $c(w, Q)$ is the count of word w in query Q and $|Q|$ is the total number of words in the query. When there is feedback information, the information would be used to improve our estimate of query language model, θ_Q.

4.2 Feedback model

Since a query model described above is usually short, the simple estimation method explained before is not discriminative. Several methods have been proposed to improve the estimation of θ_Q by exploiting documents terms, i.e., documents that are used for either relevance or pseudo-relevance feedback [41, 42, 79]. In [79], authors have defined a two-component mixture model (i.e., a fixed background language model, $p(w|\mathcal{C})$, estimated using the whole collection and unknown topic language model to be estimated) and assumed that the feedback documents are generated using such a mixture model. Formally, let θ_T be the unknown topic model and $\mathcal{F}$ be a set of feedback documents. The log-likelihood[2] function of the mixture model is:

$$\mathcal{L}(\mathcal{F}|\theta_T) = \sum_{D \in \mathcal{F}} \sum_{w \in V} c(w, D) \log[(1 - \alpha)p(w|\theta_T) + \alpha p(w|\mathcal{C})]$$

where $\alpha \in [0, 1)$ is a mixture noise parameter (which is set to 0.9 in our experiments) which controls the weight of the background model. Expectation-Maximization (EM) algorithm [19] can be used to estimate $p(w|\theta_T)$ which is then interpolated with the original query model $p(w|Q)$ to obtain an improved estimation of the query model:

$$p(w|\theta_Q) = (1 - \gamma)p(w|Q) + \gamma p(w|\theta_T)$$

where γ is the feedback coefficient to be set manually (it is set to 0.8 in our experiments).

The basic idea in relevance feedback is to extract useful terms from positive documents and use them to expand the original query. When a query is difficult, it is often impossible to obtain positive (or relevant) documents for feedback. Therefore, the best way would be to exploit the negative documents to perform negative feedback [71]. The idea of negative feedback is to identify distracting non-relevant documents and penalize unseen documents containing such information.

[2] Please note that all the logs are taken to base-e (natural logarithm), unless we explicitly mention the base that the logarithm is taken.

Formally, let θ_Q be the estimated query model, θ_D be the estimated document model, and θ_N be a negative topic model estimated based on the negative feedback documents $N = \{L_1, \ldots, L_f\}$. In MultiNeg strategy [71] which is the best performing method, the original relevance score is adjusted with multiple negative models as:

$$\begin{aligned} S(Q,D) &= S_R(Q,D) - \beta \times S(Q_{neg}, D) \\ &= S_R(Q,D) - \beta \times \max\left(\bigcup_{i=1}^{f}\left\{S\left(Q_{neg}^{i}, D\right)\right\}\right) \\ &= -D(\theta_Q||\theta_D) + \beta \times \min\left(\bigcup_{i=1}^{f}\{D(\theta_i||\theta_D)\}\right). \end{aligned}$$

Where Q_{neg} means negative query representation, i.e., the query which is formed by using the language model of negative (non-relevant) documents, Q_{neg}^{i} is the negative query representation formed from the i-th non-relevant document and Q means the original query. θ_i is a negative model for each individual negative document L_i in N. Expectation-Maximization algorithm is used to estimate a negative model θ_i. In particular, all non-relevant documents are assumed to be generated from a mixture model of a unigram language model θ_N and a background language model. Thus, the log-likelihood of the sample N is:

$$\mathcal{L}(N|\theta_N) = \sum_{D \in N}\sum_{w \in D} \log[(1-\delta)p(w|\theta_N) + \delta p(w|\mathcal{C})]$$

where δ is a mixture parameter which controls the weight of the background model and non-relevant model (it is set to 0.9 in our experiments). The result of the Expectation-Maximization algorithm gives a discriminative negative model θ_N which eliminates background noise. β is set to 0.1 in our experiments.

To perform relevance feedback after the user views the second page, we perform the two-component mixture model proposed in [79] (and described above) for positive feedback and and the MultiNeg Strategy proposed in [71] (and described above) for negative feedback; both have been shown to be effective for the respective feedback task.

4.3 Novelty method

To compute the novelty score $S_{novelty}$ described in this section, we use a method proposed in [78]. The best performing novelty measure they reported is MixAvg. The novelty measure is based on a two-component generative mixture model in which one component is the old reference topic θ_0 estimated based on $\{d_{21}, \ldots, d_{2,i-1}\}$ described in Sect. 3 and the other is the background language model (e.g., general English model). Specifically, let θ_B be a background language model with a mixing weight of μ, the log-likelihood of a new document $d = \{w_1, \ldots, w_n\}$ is:

$$\mathcal{L}(\mu|d, \theta_0) = \sum_{i=1}^{n} \log[(1-\mu)p(w_i|\theta_0) + \mu p(w_i|\theta_B)]$$

And the estimated novelty score is obtained by:

$$\mu^* = arg\max_{\mu} \mathcal{L}(\mu|d, \theta_0)$$

The EM algorithm can be used to find the unique μ^* that maximizes the score. We call this method MixAvg-MMR in our experiments.

One should note that our focus is studying the parameter λ (novelty parameter) described in Sect. 3 and the parameters that disscussed in this section, will be fixed to their optimal or default values as suggested in the literature.

5 Learning to optimize diversification

We consider two ways to optimize the diversification parameter λ. The first is simply to vary this parameter on a training data set and select the best performing value, and set this parameter to such a fixed optimal value across all the test queries. We call this approach *fixed-coefficient diversification*.

Intuitively, the right amount of diversification may depend on queries (e.g., a query with more subtopics may benefit from more diversification). Thus for our second method, we propose a learning method to "learn when to diversify the results" by adaptively learning this coefficient for each query, which we present it in details in the next section.

5.1 Features for diversification

In this section, we describe a learning approach that adaptively learns the λ parameter for each query.

We first identify some features that are correlated to the diversification parameter, i.e. λ. These features are computed based on *only* the first-page results and the judgments on it, J_1. Because at the time of optimizing the diversification on the second page, we only have this much information. In the next subsection, we will discuss how we combine these features to learn the function $A(Q, J_1)$ (which can be used to compute an optimal value for λ for each query) using the past queries as training data. The learned *function* $A(Q, J_1)$ can then be used for future queries to predict new λ. Note that since we do not learn a fixed λ value, but a function for computing an optimal value of λ, this allows us to obtain a potentially different optimal value of λ for each query.

The following notations will be used in the definition of features. F_{rel} and F_{nonRel} are the set of relevant and non-relevant documents in J_1, respectively. $c(w, Q)$ is the count of term w in query Q. $|Q| = \sum_{w \in Q} c(w, Q)$ is the total number of words in query Q. $p(w|\theta_Q) = \frac{c(w,Q)}{|Q|}$ is the query language model. $p(w|\mathcal{C})$ is the collection language model. $p(w|\theta_{F_{rel}})$ and $p(w|\theta_{F_{nonRel}})$ are the language models of relevant documents and non-relevant documents on the first-page results, respectively.

The followings are the features extracted from the first-page results. Please note that the reason for showing the first-page results to the user is to get some limited information about relevant documents and as a result extract the following features which are needed for our learning algorithm.

5.1.1 Query length

As mentioned in [29], query length affects retrieval performance and is defined as:

$$|Q| = \sum_{w \in Q} c(w, Q).$$

5.1.2 Query distribution

Each term is associated with an inverse document frequency which describes the informative amount that a term in the query carries. It is defined as follows [29]:

$$QDist = \frac{idf_{max}}{idf_{min}}$$

where idf_{max} and idf_{min} are the maximum and minimum *idf*s among the terms w in query Q, respectively.

5.1.3 Query clarity

Query clarity has been shown to predict query difficulty [18]. When a query is difficult, it might mean that it has different interpretations, so in order to complete the picture of all aspects of the query, we need to provide a diverse set of documents to make sure that all aspects of the query are covered.

1. According to definition [18], the clarity of a query is the Kullback-Leibler divergence of the query model from the collection model. For our case, the query model is estimated from the relevant documents in F_{rel} which is defined as follows:

$$QClar_1 = \sum_{w \in F_{rel}} p(w|\theta_{F_{rel}}) log_2 \frac{p(w|\theta_{F_{rel}})}{p(w|\mathcal{C})}$$

where $p(w|\theta_{F_{rel}})$ is estimated as $\frac{c(w,F_{rel})}{\sum_{w' \in F_{rel}} c(w',F_{rel})}$, where $c(w, F_{rel})$ is the number of word w in F_{rel} (i.e., relevant documents).

2. To avoid the expensive computation of query clarity, authors in [29] proposed a simplified clarity score as a comparable pre-retrieval performance predictor. It is calculated as follows:

$$QClar_2 = \sum_{w \in Q} p(w|\theta_Q) log_2 \frac{p(w|\theta_Q)}{p(w|\mathcal{C})}.$$

5.1.4 Query entropy [16, 63]

If query entropy is high, it means that the query covers broad topics, as a result, we do not need to diversify the results. We define two query entropies as follows:

1.

$$QEnt1 = \sum_{w \in Q} -p(w|\theta_Q) log_2 p(w|\theta_Q).$$

2. Since a query is often short, we compute another query entropy score based on the relevant documents in F_{rel} as follows:

$$QEnt2 = \sum_{w \in F_{rel}} -p(w|\theta_{F_{rel}}) log_2 p(w|\theta_{F_{rel}})$$

where $p(w|\theta_{F_{rel}})$is estimated as $\frac{c(w,F_{rel})}{\sum_{w' \in F_{rel}} c(w',F_{rel})}$.

5.1.5 Number of relevant documents

If the query has *high initial precision*, i.e., the fraction of retrieved relevant documents on the first-page results are high, we do not need to diversify the results; as a result we consider the number of relevant documents in the first-page results as a feature:

$$num = |F_{rel}|.$$

5.1.6 Virtual average precision

Another feature to capture *high initial precision* is to calculate *Average precision* for the first-page results as in [45]:

$$VirAP = \sum_{d \in F_{rel}} \frac{prec(r_d)}{10}$$

where r_d is the rank of document d and $prec(r_d)$ is the precision of top r_d documents.

5.1.7 Separation of relevant and non-relevant documents

If the query is *clear* enough (relevant and non-relevant documents are separated) we do not need to diversify the results. Thus we introduce the following two features to measure the separation between relevant documents in slightly different ways.

1. Jensen-Shannon Divergence (JSD)

$$JSD = \frac{1}{2}[D(\theta_{F_{rel}}||\theta_{F_{nonRel}}) + D(\theta_{F_{nonRel}}||\theta_{F_{rel}})]$$

where D is the Kullback-Leibler divergence between the two models.

2. Cosine Similarity: We denote the Term Frequency vectors of F_{rel} and F_{nonRel} as $V_{F_{rel}}$ and $V_{F_{nonRel}}$, respectively. The Cosine Similarity is then defined as:

$$CosSim = \frac{V_{F_{rel}} \cdot V_{F_{nonRel}}}{||V_{F_{rel}}|| \cdot ||V_{F_{nonRel}}||}.$$

5.1.8 Diversification (Div)

Intuitively, if the baseline results are already diversified, we do not need to do more diversification; one measure to capture that is as follows: We cluster the top 30 results from Kullback-Leibler divergence retrieval to 5 clusters using K-Means algorithm [20]. We then consider the ratio of the size of the first largest cluster to the second largest cluster. If this ratio is small, it means that we have already formed multiple clusters and the results are already diversified, so we do not need to diversify more.

5.1.9 Analysis of computational efficiency of the features

The features discussed above can be categorized into query-based features and document-based features. Query-based features are those that are calculated based on the query. Examples of these kinds of features are: query length $|Q|$, query distribution *QDist*, query entropy *QEnt*1 and etc. Document-based features are those that are calculated based on relevant and non-relevant documents on the first-page results. Examples are: query clarity $QClar_1$, query entropy *QEnt*2 and etc. Since we only have 10 documents on the first-page results, the calculation of language models based on both relevant and non-relevant documents can be computed efficiently for online prediction. Also, the query-based features can be computed efficiently for online prediction.

5.2 Learning algorithm

We would like to use a learning technique to combine the features to predict the novelty parameter for diversification. In this paper, we use logistic regression[3] since it can take any input value and outputs a value between zero and one which is what we want. The logistic regression model is of the form: $f(z) = \frac{1}{1+\exp(-z)}$, where z is a set of features and $f(z)$ is the probability of an outcome. $f(z)$ is used to predict the novelty coefficient, i.e., λ. z is of the form $z = \bar{w}\bar{x}$ where $\bar{x}$ is a vector of the features and $\bar{w}$ is the learned weight (based on our training data) associated with each feature. When a weight is positive, it means that the corresponding feature increases the probability of outcome while a negative coefficient means the opposite. Once these weights are learned based on our training data (i.e., past queries and their optimal λ values as determined by some utility function to be defined in the next section), we can use the model (features along with their weights) to predict the diversity coefficient parameter (λ) for test queries. Statistical package R[4] is used to train the model.

5.3 Evaluation metric

Our goal is to optimize the utility to a user over all the three pages P_1, P_2, and P_3. We thus need to define how we measure the utility over all these pages. Such a measure is needed to evaluate the effectiveness of any method for optimizing exploration–exploitation tradeoff and also needed to set a criterion for selecting the optimal λ values for training in learning an adaptive diversification coefficient.

Since the first-page results is the same for all the methods, we only consider the utility on the second and third-page results. The user is assumed to see all the results on the second page, thus we define the utility on the second page as the number of relevant documents on the second page, denoted as $U(P_2) = REL(P_2)$.

For the third page, since the user is only assumed to see some results, the utility is defined as the expected number of relevant documents that the user would see on the third page:

$$U(P_3) = \sum_{j=1}^{10} s_j \sum_{i=1}^{j} \delta(i) \tag{1}$$

where s_j denotes the probability that the user would stop reading after seeing the top j documents and $\delta(i)$ is 1 if the i-th document is relevant and is zero otherwise.

[3] Since logistic regression has a global optimum, the choice of the learning algorithm is of little importance.

[4] http://www.r-project.org/.

If we assume the uniform stopping probability, i.e., s_j is uniform (i.e. $s_j = \frac{1}{10}$ since we have 10 documents per result page), we would model the impatient user [47] and the utility on the third page with some algebraic transformation is reduced to:

$$U(P_3) = \sum_{n=1}^{10} \frac{(11-n)}{10} \cdot \delta(n) \tag{2}$$

where n is the ranking of the document. For example, if a document is ranked first on the third page, the expected utility for that document would be one since $n = 1$, similarly, if the document is ranked at 10-th, the expected utility from that document would be $\frac{1}{10}$.

This measure is reasonable for a user who is "impatient" who might indeed stop at any of the positions equally likely. However, if the user is more "patient" with a high probability of viewing all the results on the third page, we would like to put more weight on the utility from the third page. One way to account for that is to parameterize the weights on the third page so that we can examine this effect as follows:

$$U(P_3) = \sum_{n=1}^{10} [(\frac{(k-1)}{9}) \cdot (n-1) \cdot \delta(n) + 1] \tag{3}$$

where $k \in [0, 1]$. When $k = 0.1$, we gain Eq. (2).

The total utility on both the second page and the third page is thus $U(P_2) + U(P_3)$.

6 Experiment design

We used three standard TREC data sets in our study[5]: TREC2004 robust track, WT2G and AP88-90 which represent different genre of text collections. The robust track is a standard *ad hoc* retrieval with an emphasis on the overall reliability of IR systems which contains difficult queries and is a heterogeneous data set. There were 249 queries[6] in this data set. WT2G is a web data set with 50 queries while AP88-90 is a homogeneous data set with 149 queries[7]. We used Porter stemming and did stop word removal on all these collections.

Since we do not have a real system for interactive search, we simulated user-clicked documents (user feedback) as follows: We used Lemur toolkit[8] to index document collections and retrieved a document list for each query using Kullback-Leibler divergence retrieval model [41] to return 10 result documents (this is to simulate the first-page results). The reason for showing this page to the user is to get some limited information about relevant documents and extract features that we need for our learning algorithm. Also, the reason for returning only 10 documents for each page is to resemble the Web Search Engine that shows 10 results per page, so we also simulate to have 10 results per page. We then extract features from the first-page results to decide if we need to diversify the results on the second page and then feedback is used to improve the ranking of documents on the third page. (We assume that the user would click on a relevant document.)

We compare our proposed methods (i.e., FixedDivFB and AdaptDivFB) with baseline methods, i.e., NoFeedback and RegularRelFB by measuring the total utility over all three

[5] All experiments measured according to Eq. (2) unless otherwise stated.

[6] One query was dropped because the evaluators did not provide any relevant documents for it.

[7] One query was dropped because the evaluators did not provide any relevant documents for it.

[8] http://www.lemurproject.org/.

pages. Since the first page is the same for all the methods, we actually only computed the utility over the second and third pages. While the first-page results are similar for all these four methods, the second-page results for different methods are different. For NoFeedback and RegularRelFB, we return 10 results using Kullback–Leibler divergence retrieval model [41]. For FixedDivFB, we use MixAvg-MMR [78] with a learned fixed novelty parameter (using training queries) for all test queries and for AdaptDivFB, we use adaptive novelty coefficient (by learning it with training data) for each individual query. Then for the third page, we use language model feedback [71, 79] to return 10 documents as the third-page results. Figure 1 shows these methods clearly.

To train our proposed methods, we need to obtain training data first. In our study, we used fivefold cross validation to split our queries for testing and training purposes for each data set and get the average across all folds. Since we are restricted to vary the parameters in testing stage, we set the parameters to their optimal values gained from training data in the testing stage, i.e., we set the Dirichlet prior smoothing to its optimal value for each training data using NoFeedback method, and set the feedback coefficient to its optimal value using RegularRelFB. We fixed feedback term count to 100 and mixture noise parameter [79] to 0.9 and β in negative feedback [71] to 0.1. We did not vary these parameters because this is not our purpose of study. For FixedDivFB, we choose an optimal fixed coefficient novelty, i.e., λ for all queries learned based on training data. We then use the optimal parameters gained from training data to measure the performance of test queries. The needed training data for AdaptDivFB is of the form of a set of feature vectors computed based on different queries and retrieval results along with the corresponding optimal novelty parameter. The learning task is to learn from such a training data set to predict the optimal novelty parameter for a new test query based on its corresponding feature values. To get the optimal novelty parameter for training queries for AdaptDivFB, we try different novelty coefficient $\lambda \in \{0, 0.1, 0.2, \ldots, 1\}$ using MixAvg-MMR [78] on training data sets and we choose the best λ for each training query to form our training data set. As a result, the main difference between the FixedDivFB and AdaptDivFB methods lies in what they can learn from the training data: the FixedDivFB learns a fixed novelty coefficient, λ, that leads to the best utility (described in Sect. 5.3) on the training data set, while the AdaptDivFB method learns a *prediction model* that best fits the training data set.

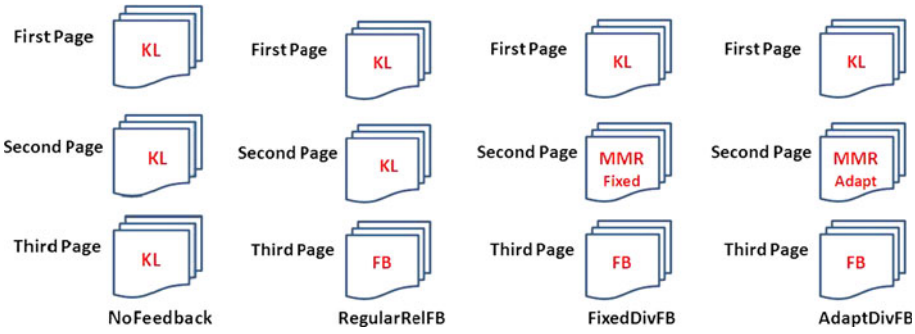

Fig. 1 Visualization of different methods. KL means Kullback-Leibler divergence retrieval model [41], MMR Fixed means MixAvg-MMR using fixed novelty coefficient for all queries, MMR Adapt means MixAvg-MMR using adaptive novelty coefficient for each query and FB means language model feedback [71, 79]

7 Experiment results

Our main hypothesis is that optimizing the exploration–exploitation tradeoff over a session is effective and we need to optimize such a tradeoff for each individual query. In order to test these hypotheses, we conduct a set of experiments. In Sect. 7.1, we empirically examine the exploration–exploitation tradeoff. Next, in Sect. 7.2, we examine the effectiveness of optimizing the total user utility over a session. In Sect. 7.3, we go further to see the effect of optimizing either exploration or exploitation and compare the results when we optimize based on the combination. Then, in Sect. 7.4, we examine what kind of queries would benefit from optimizing the exploration–exploitation tradeoff. Next, we consider the effect of optimizing the exploration–exploitation and evaluate this effect on user patience. Since there are so many methods for diversification, in Sect. 7.6 we consider a different diversification method and evaluate the effect of the diversification method on our main hypothesis. Finally, we analyze our features for optimizing the novelty coefficient.

7.1 Is optimal exploration–exploitation tradeoff query dependent?

Diversification methods are usually controlled by an interpolation novelty coefficient λ to control the balance between relevance and redundancy. According to the description in Sect. 3, when $\lambda = 1$, the emphasize is on *novelty* whereas for $\lambda = 0$, the emphasize is on *relevancy*. To show the sensitivity of λ to the utility function described on Sect. 5.3, we plot the utility of several randomly selected topics from each category, i.e., easy[9], difficult[10] and others (351, 421, 425) by varying λ from 0 to 1. The results are shown in Fig. 2. These patterns show that the optimal exploration–exploitation tradeoff is query dependent and it is important to dynamically optimize the novelty coefficient in a per-query basis.

7.2 Effectiveness of optimizing the total user utility over a session

Our hypothesis is that in order to achieve optimal retrieval performance and outperform relevance feedback, we need to optimize based on the *total user utility*. In order to test our hypothesis, we compare our proposed methods (i.e., FixedDivFB and AdaptDivFB) with baselines (i.e., NoFeedback and RegularRelFB) which are optimized based on the whole utility, i.e., utilities on the second and third pages. Table 1 shows the comparison of our methods with baselines across different TREC data sets (we show the utilities based on the second, third and total (second + third) pages.). The results indicate that method RegularRelFB is better than NoFeedback as we expected since feedback outperforms the basic retrieval model using Kullback-Leibler divergence. Method FixedDivFB outperforms RegularRelFB and AdaptDivFB outperforms both FixedDivFB and RegularRelFB methods since it uses different novelty coefficient for each query. Statistical Significant tests using Wilcoxon singed-rank test [72] indeed indicate that our proposed methods are statistically significant over method RegularRelFB[11]. The table also shows percentage

[9] We define a query as easy when its Precision@10 is 0.9 or 1, given a retrieval model.

[10] We define a query as difficult when its Precision@10 is no larger than 0.1, given a retrieval model.

[11] The reason why AdaptDivFB is not statistically significant over FixedDivFB for WT2G data set is because we only have 50 queries and since we do fivefold cross validation, we have only 40 training queries in each fold which presumably is not sufficient for learning.

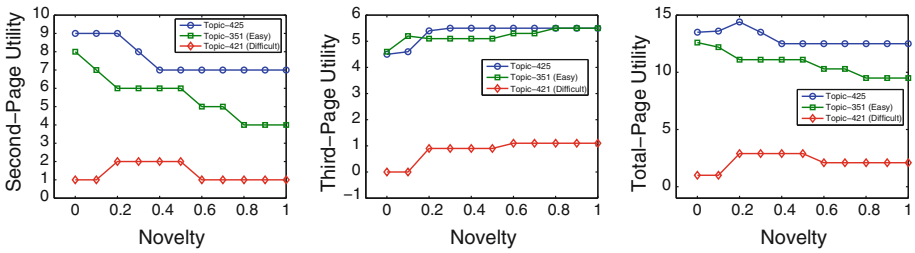

Fig. 2 Exploration–exploitation tradeoff patterns. Optimal exploration–exploitation tradeoff is query dependent

improvement, e.g., FixedDivFB/RegularRelFB means the percentage improvement for FixedDivFB over RegularRelFB. Thus, by optimizing exploration–exploitation tradeoff, both FixedDivFB and AdaptDivFB outperform the regular relevance feedback method, i.e., RegularRelFB which ignores exploration. And indeed AdaptDivFB outperforms FixedDivFB method which indicates that the optimal exploration–exploitation tradeoff is query dependent.

Please note that while it appears that AdaptDivFB is simply more effective than RegularRelFB even for the second page (where we would expect RegularRelFB to have some advantage), a decomposed analysis in Table 3 shows that the improvement on the second page comes from difficult queries, and AdaptDivFB does not perform as well as RegularRelFB for easy queries on the second page.

7.3 Detailed analysis of exploration–exploitation tradeoff

Our hypothesis is that to achieve optimal overall utility on a session, we should train with a "similar" objective function (i.e., a similar utility measure on the training data). Thus, we expect training to optimize utility on both second and third page should lead to better performance than training to optimize either one alone. In particular, optimizing the second-page **only** would lead to over-exploitation (ending up with lower third-page utility), while optimizing third-page **only** would be the opposite.

In order to see if this hypothesis is true, we conduct two experiments: (1) optimizing the second-page utility **only** (exploitation), (2) optimizing the third-page utility **only** (exploration). The results are shown in Table 2 (we only show the results for AdaptDivFB method, similar patterns can also be seen for FixedDivFB method.). From the table, we see that across all data sets, the second-page utility once optimized on the second-page only is higher than when it is trained based on both pages. And indeed the third-page utility is lowered. The opposite trend could be explained when optimizing on the third-page only. The table also shows the percentage improvements, e.g., AdaptDivFB (2nd + 3rd/2nd) indicates the percentage improvement when optimized on both pages versus optimized on the second page only.

The other interesting observation is that the overall utility is degraded and this indeed indicates that optimizing the total utility, i.e. both second and third-page utility, is necessary to lead to the *optimal* retrieval performance.

These observations indeed confirm our hypothesis that in order to have the optimal retrieval performance, the exploration–exploitation tradeoff needs to be optimized.

Table 1 Comparison of different methods on different TREC data sets

Methods	Second	Third	Total
Robust 2004			
NoFeedback	2.8593	1.2171	4.0764
RegularRelFB	2.8593	2.2749	5.1342
FixedDivFB	2.8513	2.3207	5.172*
FixedDivFB/RegularRelFB	−0.27 %	2.01 %	0.74 %
AdaptDivFB	2.919	2.3511	**5.2701***,§
AdaptDivFB/RegularRelFB	2.08 %	3.34 %	2.65 %
AdaptDivFB/FixedDivFB	2.37 %	1.31 %	1.9 %
WT2G			
NoFeedback	2.7	1.276	3.976
RegularRelFB	2.7	1.632	4.332
FixedDivFB	2.8	1.722	4.522*
FixedDivFB/RegularRelFB	3.7 %	5.51 %	4.385 %
AdaptDivFB	2.84	1.7	**4.58***
AdaptDivFB/RegularRelFB	5.18 %	6.61 %	5.72 %
AdaptDivFB/FixedDivFB	1.43 %	1.045 %	1.28 %
AP88-90			
NoFeedback	2.5154	1.3149	3.8303
RegularRelFB	2.5154	2.4153	4.9307
FixedDivFB	2.558	2.4267	4.9847*
FixedDivFB/RegularRelFB	1.69 %	0.47 %	1.09 %
AdaptDivFB	2.6633	2.4446	**5.1079***,§
AdaptDivFB/RegularRelFB	5.88 %	1.21 %	3.59 %
AdaptDivFB/FixedDivFB	4.12 %	0.73 %	2.47 %

* and § mean significant over RegularRelFB and FixedDivFB, respectively

7.4 What kind of queries most benefit from optimizing exploration–exploitation tradeoff?

Our hypothesis is that if a query is more *difficult*, it would benefit more from optimizing the exploration–exploitation tradeoff. Indeed, the results in Table 3 (these results are based on their counterpart results in Table 1 for Robust 2004 data set.) confirm this hypothesis. We separate the results in Table 1 into easy and difficult queries and measure their performance.

As we see from these results, it is clear that optimizing the exploration–exploitation tradeoff helps difficult queries more than easy queries, i.e., both methods FixedDivFB and AdaptDivFB outperform baseline results for difficult queries but the improvement for easy queries is negligible.

Another observation from this table is that increasing diversity helps difficult queries due to the implied (desirable) negative feedback however, hurts easy queries because of the implied (incorrect) negative feedback.

7.5 Exploration–exploitation tradeoff and user patience

Since the standard relevance feedback does not do exploration, in "certain situations", i.e., when a user is more *patient*, exploration would have more benefit. When a user is more

Table 2 Comparison of optimizing based on second-page utility, third-page utility and second + third utility for AdaptDivFB

AdaptDivFB			
Methods	Second	Third	Total
Robust 2004			
AdaptDivFB (2nd)	2.976	2.1905	5.1665
AdaptDivFB (3rd)	2.5249	2.4427	4.9676
AdaptDivFB (2nd + 3rd)	2.919	2.3511	**5.2701***,§
AdaptDivFB (2nd + 3rd/2nd)	−1.91 %	7.33 %	2.005 %
AdaptDivFB (2nd + 3rd/3rd)	15.60 %	−3.75 %	6.09 %
WT2G			
AdaptDivFB (2nd)	2.9	1.57	4.47
AdaptDivFB (3rd)	2.38	1.962	4.342
AdaptDivFB (2nd + 3rd)	2.84	1.74	**4.58***,§
AdaptDivFB (2nd + 3rd/2nd)	−2.07 %	10.82 %	2.46 %
AdaptDivFB (2nd + 3rd/3rd)	19.32 %	−11.31 %	5.48 %
AP88-90			
AdaptDivFB (2nd)	2.728	2.226	4.954
AdaptDivFB (3rd)	2.482	2.516	4.998
AdaptDivFB (2nd + 3rd)	2.6633	2.4446	**5.1079***,§
AdaptDivFB (2nd + 3rd/2nd)	−2.37 %	9.82 %	3.11 %
AdaptDivFB (2nd + 3rd/3rd)	7.3 %	−2.83 %	2.19 %

* and § mean significant over AdaptDivFB (2nd) and AdaptDivFB (3rd), respectively

Table 3 Comparison of methods for difficult and easy queries on Robust 2004 data set, 51 difficult queries and 21 easy queries

Methods	Second	Third	Total
Difficult queries			
NoFeedback	0.7129	0.5208	1.2337
RegularRelFB	0.7129	1.4738	2.1867
FixedDivFB	0.8256	1.497	2.3226*
AdaptDivFB	0.9197	1.5091	**2.4288***,§
Easy queries			
NoFeedback	6.7	2.6223	9.3223
RegularFB	6.7	3.901	10.601
FixedDivFB	6.45	4.02	10.47
AdaptDivFB	6.5333	4.068	**10.6013**§

* and § mean significant over RegularRelFB and FixedDivFB, respectively

patient, there is a high probability of viewing all the results on the third page, as a result, putting more weight on the utility from the third page would be more beneficial for such a user. As we discussed in Sect. 5.3, we model user patience with k (Eq. 2), so we now have a different utility measure which is parameterized with k. Our hypothesis is that for patient users, *exploration* is more useful.

In Fig. 3, we vary k and measure the performance for RegularRelFB and FixedDivFB[12]. Figure 3 (left) shows the difference between these two methods; as k gets larger, the difference (i.e., FixedDivFB-RegularRelFB) between these methods is larger which

[12] Similar trends can be seen for AdaptDivFB.

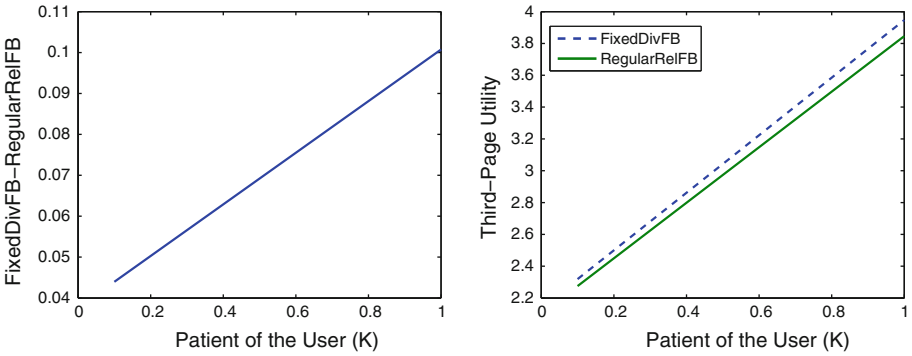

Fig. 3 Modeling patience of the user (based on Robust 2004)

indicates that the benefit from exploration is more amplified as k increases. Also, Fig. 3 (right) shows their performance on the third page (to show the exploration benefit) for both methods, it also indicates that as k gets larger, the difference between these two methods is more amplified. (The linear trend curves are expected given the form of our utility function.)

7.6 Sensitivity to diversification methods

We also want to know how our findings chang if we use a different diversification method, i.e., MMR-PLSA proposed in [39]. In this method, both the documents and the query would be mapped to a low-dimensional space representation through Probabilistic Latent Semantic Indexing (PLSA) model [32], and then a similar greedy algorithm to MMR [8] is used to select a diverse set of documents.

The results for this method are shown in Table 4. These results also support our *main finding*, i.e., optimizing *exploration–exploitation tradeoff* outperforms traditional relevance feedback, and adaptive optimization (AdaptDivFB) is better than fixed-coefficient optimization (FixedDivFB).

7.7 Analysis of features for optimizing novelty coefficient

In this section, we discuss about the feature distributions and how they are linked to the novelty coefficient. As discussed, we do fivefold cross validation, and for each fold we have a different model according to our training data. So, in total, we have 15 models for method AdaptDivFB in Table 1 (3 data sets and fivefold). For each model, we use AIC (Akaike Information Criterion) [27] to include only statistical significant features. In order to understand the correlations of features to the novelty parameter, in Table 5, we show the distributions across all 15 models. Negative means there is a negative correlation between that feature and novelty parameter whereas Positive means the opposite. As shown in the table, features *QEnt*1, *QClar*2, *VirAP* and *Div* are the most frequently used features by these 15 models.

The coefficients in this table are consistent with what we discussed in Sect. 5.1, i.e., if a query is long (|Q|), or the similarity between relevant and non-relevant documents is high (CosSim), or when the results are not sufficiently diversified (Div), we need diversification. In other cases, i.e., when a query is clear enough (QClar2) or relevant documents contain broad topics (QEnt1), we do not need diversification. The contradictory effect of negative

Table 4 Comparing different methods and using MMR-PLSA as a diversification method

Methods	Second	Third	Total
Robust 2004			
NoFeedback	2.8593	1.2117	4.0764
RegularRelFB	2.8593	2.2749	5.1342
FixedDivFB	2.8511	2.3597	5.2108*
AdaptDivFB	2.911	2.4007	**5.3117***,§
WT2G			
NoFeedback	2.7	1.276	3.976
RegularRelFB	2.7	1.632	4.332
FixedDivFB	2.86	1.732	4.92*
AdaptDivFB	3	1.776	**4.776***,§
AP88-90			
NoFeedback	2.5154	1.3149	3.8303
RegularRelFB	2.5154	2.4153	4.9307
FixedDivFB	2.604	2.401	5.005*
AdaptDivFB	2.6377	2.4059	**5.0436***

* and § mean significant over RegularRelFB and FixedDivFB, respectively

Table 5 Feature distributions

	Negative (%)	Positive (%)
$\|Q\|$	–	20
QDist	33.3	–
$QEnt_1$	53.3	–
$QEnt_2$	26.6	–
$QClar_1$	20	–
$QClar_2$	40	–
num	33.3	–
JSD	26.6	–
CosSim	–	33.3
Div	–	40
VirAP	53.3	–
$\exp(QClar_1)$	–	26.6

coefficient of *QClar*1 and positive coefficient of *exp*(*QClar*1) could be explained as follows: in most models these two features co-occur with each other and the coefficients are such that when a query is *clear* enough, i.e., *QClar*1 is relatively large, the overall effect would be negative suggesting that we should use a smaller novelty coefficient. However, when a query is not clear enough, the overall effect is positive which suggests we need to use a larger diversity coefficient. The interesting observation of negative influence of VirAP indeed confirms our hypothesis that for difficult queries, we need more diversification.

8 Conclusions

In this paper, we studied how to optimize relevance feedback to maximize the total utility over an entire interaction session. Our work is a first step toward the general goal of

optimizing the utility in the whole interaction session in relevance feedback. In particular, we studied the issue of exploration–exploitation tradeoff in interactive feedback, which is the tradeoff between presenting search results with the highest immediate utility to a user and presenting search results with the best potentials for collecting feedback information. We framed this tradeoff as a problem of optimizing the diversification of search results. We proposed two methods that optimize the exploration–exploitation tradeoff. The first method is to fix a novelty coefficient for diversification and the other one is to adaptively optimizing the diversification of search results for each query. We also defined utility from user perspective and defined how we can model both patient and impatient user. Experiment results on three representative TREC data sets indicate that our proposed methods are effective for optimizing the tradeoff between exploration and exploitation and outperform the traditional relevance feedback which only does exploitation without exploration. In summary, our findings are as follows:

- Optimal exploration–exploitation tradeoff is query dependent and it is important to dynamically optimize the novelty coefficient in a per-query basis.
- In order to achieve optimal retrieval performance and outperform relevance feedback, we need to optimize based on the total user utility. In other words, in order to have the optimal retrieval performance, the exploration–exploitation tradeoff needs to be optimized.
- When a user is more patient, exploration would have more benefit because there is a high probability of viewing all the results on later pages.
- If a query is more difficult, it would benefit more from optimizing the exploration–exploitation tradeoff.

One limitation of our study is that most users of a current Web search engine do not view so many pages, even though in the case of a high-recall search task or when a user uses a small-screen device (e.g., a smartphone), we can expect a user to often view more than two pages of results. Thus it would be interesting to consider more realistic assumptions such as considering fewer results per page to simulate the smartphone scenario. Also, it would be interesting to evaluate the methods using the actual click-through data from query logs. Ideally, building a real-system that involves real users in interaction would be interesting.

Another limitation of our work is that in our current formulation, we assumed the availability of some limited feedback information from the first-page result to make the problem more tractable; an interesting future direction would be to just get information about the query which would also increase the applicability of our method, but the problem would be that the only information we have available is information regarding the query and we do not yet know anything about the relevant/non-relevant documents, so it would also be harder to solve the problem of optimizing the diversity parameter.

In the current problem formulation, we only consider the second-page when measuring the novelty and we only use the feedback information from the second page to re-rank results on the third page, so the diversity on the second page is what matters in terms of optimizing the exploration–exploitation tradeoff. For future, it is interesting to explore optimizing the diversity in the combined set of first and second page.

Acknowledgments We thank the anonymous reviewers for their useful comments. This material is based upon work supported by the National Science Foundation under Grant Numbers IIS-0713581, CNS-0834709, and CNS 1028381, by NIH/NLM grant 1 R01 LM009153-01, and by a Sloan Research Fellowship. Maryam Karimzadehgan was supported by the Google PhD fellowship. Any opinions, findings,

conclusions, or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

References

1. Abbeel, P., Coates, A., Quigley, M., & Ng, A. Y. (2006). An application of reinforcement learning to aerobatic helicopter flight. *NIPS* (pp. 1–8).
2. Agarwal, D., Chen, B. -C., & Elango, P. (2009). Explore/exploit schemes for web content optimization. *ICDM*.
3. Agrawal, R., Gollapudi, S., Halverson, A., & Leong, S. (2009). Diversifying search results. *ACM WSDM* (pp. 5–14).
4. Bookstein, A. (1983). Information retrieval: A sequential learning process. *Journal of American Society (ASIS), 34*(5), 331–342.
5. Buckley, C. (2004). Why current ir engines fail. *ACM SIGIR*, 584–585.
6. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., et al. (2005). Learning to rank using gradient descent. *ICML* (pp. 89–96).
7. Cao, Z., Qin, T., Liu, T. -Y., Tsai, M. -F., & Li, H. (2007). Learning to rank: From pairwise approach to listwise approach. *ICML* (pp. 129–136).
8. Carbonell, J., & Goldstein, J. (1998). The use of mmr, diversity-based reranking for reordering documents and producing summaries. *ACM SIGIR* (pp. 335–336).
9. Carmel, D., Yom-Tov, E., Darlow, A., & Pelleg, D. (2006). What makes a query difficult? *ACM SIGIR* (pp. 390–397).
10. Carterette, B., & Chandar, P. (2009). Probabilistic models of novel document rankings for faceted topic retrieval. *ACM CIKM* (pp. 1287–1296).
11. Carterette, B., & Petkova, D. (2006). Learning a ranking from pairwise preferences. *ACM SIGIR* (pp. 629–630).
12. Chen, H., & Karger, D. R. (2006). Less is more: Probabilistic models for retrieving fewer relevant documents. *ACM SIGIR* (pp. 429–436).
13. Clarke, C. L., Kolla, M., Cormack, G. V., Vechtomova, O., Ashkan, A., Bnttcher, S., et al. (2008). Novelty and diversity in information retrieval evaluation. *ACM SIGIR* (pp. 659–666).
14. Clough, P., Sanderson, M., Abouammoh, M., Navarro, S., & Paramita, M. L. (2009). Multiple approaches to analysing query diversity. *ACM SIGIR* (pp. 734–735).
15. Cohn, D. A., Ghahramani, Z., & Jordan, M. I. (1996). Active learning with statistical models. *AI Journal, 4*, 129–145.
16. Collins-Thompson, K., & Bennett, P. N. (2009). Estimating query performance using class predictions. *ACM SIGIR* (pp. 672–673).
17. Cooper, W. S., Gey, F. C., & Dabney, D. P. (1992). Probabilistic retrieval based on staged logistic regression. *ACM SIGIR* (pp. 198–210).
18. Cronen-Townsend, S., Zhou, Y., & Croft, W. B. (2002). Predicting query performance. *ACM SIGIR* (pp. 299–306).
19. Dempster, A. P., Laird, N. M., & Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of Royal Statist, 39*, 1–38.
20. Faber, V. (1994). Clustering and the continuous k-means algorithm. *Los Alamos Science, 22*, 138–144.
21. Fuhr, N. (2008). A probability ranking principle for interactive information retrieval. *Information Retrieval Journal, 11*(3), 251–265.
22. Fuhr, N., & Buckley, C. (1991). A probabilistic learning approach for document indexing. *TOIS, 9*(3), 223–248.
23. Gey, F. C. (1994). Inferring probability of relevance using the method of logistic regression. *ACM SIGIR* (pp. 222–231).
24. Goffman, W. (1964). A searching procedure for information retrieval. *Information Storage and Retrieval, 2*, 73–78.
25. Harman, D. (1992). Relevance feedback revisited. In *Proceedings of ACM SIGIR 1992* (pp. 1–10).
26. Harman, D., & Buckley, C. (2004). Sigir 2004 workshop: Ria and where can ir go from here. *SIGIR Forum, 38*(2), 45–49.
27. Hastie, T., Tibshirani, R., & Friedman, J. (2001). *The elements of statistical learning*. New York, USA: Springer Series in Statistics.
28. Hauff, C., Murdock, V., & Baeza-Yates, R. (2008). Improved query difficulty prediction for the web. *ACM CIKM* (pp. 439–448).

29. He, B., & Ounis, I. (2004). Inferring query performance using pre-retrieval predictors. *SPIRE* (pp. 43–54).
30. He, J., Larson, M., & Rijke, M. D. (2008). Using coherence-based measures to predict query difficulty. *ECIR* (pp. 689–694).
31. Herbrich, R., Graepel, T., & Obermayer, K. (2000). Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers, 88*, 115–132.
32. Hofmann, T. (1999). Probabilistic latent sematic indexing. In *Proceedings of ACM SIGIR'99* (pp. 50–57).
33. Jaakkola, T., & Siegelmann, H. (2001). *Active information retrieval. NIPS*.
34. Jensen E. C., Beitzel, S. M., Grossman, D., Frieder, O., & Chowdhury, A. (2005). Predicting query difficulty on the web by learning visual clues. *ACM SIGIR* (pp. 615–616).
35. Joachims, T. (2002). Optimizing search engines using clickthrough data. *ACM KDD* (pp. 133–142).
36. Joachims, T., Freitag, D., & Mitchell, T. (1997). Webwatcher: A tour guide for the world wide web. *IJCAI*.
37. Karimzadehgan, M., & Zhai, C. (2010). Exploration-exploitation tradeoff in interactive relevance feedback. *ACM CIKM* (pp. 1397–1400).
38. Karimzadehgan, M., & Zhai, C. (2011). Improving retrieval accuracy of difficult queries through generalizing negative document language models. *ACM CIKM* (pp. 27–36).
39. Karimzadehgan, M., Zhai, C., & Belford, G. (2008). Multi-aspect expertise matching for review assignment. *ACM CIKM* (pp. 1113–1122).
40. Kumar, P. R., & Varaiya, P. P. (1986). *Stochastic systems: Estimation, identification, and adaptive control*. Englewood Gifts, NJ: Prentice Hall.
41. Laffetry, J., & Zhai, C. (2001). Document language models, query models, and risk minimization for information retrieval. *ACM SIGIR* (pp. 111–119).
42. Lavrenko, V., & Croft, W. B. (2001). Relevance-based language models. *ACM SIGIR* (pp. 120–127).
43. Leuski, A. (2001). *Interactive information organization: Techniques and evaluation*. PhD Thesis, University of Massachusetts.
44. Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. *ACM SIGIR* (pp. 3–12).
45. Lv, Y., & Zhai, C. (2009). Adaptive relevance feedback in information retrieval. *ACM CIKM* (pp. 255–264).
46. McCallum, A., & Nigam, K. (1998). Employing em and pool-based active learning for text classification. *ICML* (pp. 350–358).
47. Moffat, A., & Zobel, J. (2008). Rank-biased precision for measurement of retrieval effectiveness. *ACM Transactions on Information Systems, 27*(1).
48. Pandey, S., Chakrabarti, D., & Agarwal, D. (2007). Multi-armed bandit problems with dependent arms. *ICML* (pp. 721–728).
49. Pandey, S., Roy, S., Olston, C., Cho, J., & Chakrabarti, S. (2005). Shuffling a stacked deck: The case for partially randomized ranking of search engine results. *VLDB* (pp. 781–792).
50. Paramita, M. L., Sanderson, M., & Clough, P. (2009). Diversity in photo retrieval: Overview of the imageclefphoto task 2009. *CLEF workshop*.
51. Ponte, J. M., & Croft, W. B. (1998). A language modeling approach to information retrieval. *ACM SIGIR* (pp. 275–281).
52. Radlinski, F., & Dumais, S. (2006). Improving personalized web search using result diversification. *ACM SIGIR* (pp. 691–692).
53. Radlinski, F., & Joachims, T. (2005). Query chains: Learning to rank from implicit feedback. *ACM KDD* (pp. 239–248).
54. Radlinski, F., Kleinberg, R., & Joachims, T. (2008). Learning diverse rankings with multi-armed bandits. *ICML* (pp. 784–791).
55. Robertson, S. E., & Jones, K. S. (1976). Relevance weighting of search terms. *Journal of the American Society of Information Science, 27*(3), 129–146.
56. Rocchio, J. (1971). Relevance feedback in information retrieval. In *The SMART retrieval system* (pp. 313–323).
57. Roy, N., & McCallum, A. (2001). Toward optimal active learning through sampling estimation of error reduction. *ICML* (pp. 441–448).
58. Salton, G., & Buckley, C. (1990). Improving retrieval performance by relevance feedback. *Journal of Information Science, 41*(4), 288–297.
59. Sanderson, M., Tang, J., Arni, T., & Clough, P. (2009). What else is there? search diversity examined. *ECIR* (pp. 562–569).

60. Scholer, F., & Garcia, S. (2009). A case for improved evaluation of query difficulty prediction. *ACM SIGIR* (pp. 640–641).
61. Shen, X., & Zhai, C. (2005). Active feedback in ad hoc information retrieval. *ACM SIGIR* (pp. 59–66).
62. Singh, S., Litman, D., Kearns, M., & Walker, M. (2002). Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence, 16*, 105–133.
63. Song, R., Luo, Z., Wen, J. -R., Yu, Y., & Hon, H. -W. (2007). Identifying ambiguous queries in web search. *WWW* (pp. 1169–1170).
64. Tang, J., & Sanderson, M. (2010). Evaluation and user preference study on spatial diversity. *ECIR* (pp. 179–190).
65. Tao, T., & Zhai, C. (2006). Regularized estimation of mixture models for robust pseudo-relevance feedback. *ACM SIGIR* (pp. 162–169).
66. Thrun, S. B. (1992). The role of exploration in learning control. In *Handbook for intelligent control: Neural, fuzzy and adaptive approaches*. Florence, Kentucky: Van Nostrand Reinhold.
67. Tong, S., & Koller, D. (2000). Support vector machine active learning with applications to text classification. *ICML* (pp. 999–1006).
68. Vee, E., Srivastava, U., Shanmugasundaram, J., Bhat, P., & Yahia, S. A. (2008). Efficient computation of diverse query results. *ICDE* (pp. 228–236).
69. Voorhees, E. M. (2004). Overview of the trec 2004 robust retrieval track. *TREC*.
70. Voorhees, E. M. (2005). Draft: Overview of the trec 2005 robust retrieval track. *Notebook of TREC 2005*.
71. Wang, X., Fang, H., & Zhai, C. (2008). A study of methods for negative relevance feedback. *ACM SIGIR* (pp. 219–226).
72. Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics, 1*, 80–83.
73. Xu, Z., Akella, R., & Zhang, Y. (2007). Incorporating diversity and density in active learning for relevance feedback. *ECIR* (pp. 246–257).
74. Xu, Z., & Ram (2008). A bayesian logistic regression model for active relevance feedback. *ACM SIGIR* (pp. 227–234).
75. Yom-Tov, E., Fine, S., Carmel, D., & Darlow, A. (2005). Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. *ACM SIGIR* (pp. 512–519).
76. Yue, Y., & Joachims, T. (2008). Predicting diverse subsets using structural svms. *ICML* (pp. 1224–1231).
77. Yue, Y., & Joachims, T. (2009). Interactively optimizing information retrieval systems as a dueling bandits problem. *ICML* (pp. 1201–1208).
78. Zhai, C., Cohen, W., & Lafferty, J. (2003). Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. *ACM SIGIR* (pp. 10–17).
79. Zhai, C., & Lafferty, J. (2001). Model-based feedback in the language modeling approach to information retrieval. *ACM CIKM* (pp. 403–410).
80. Zhai, C., & Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. *ACM SIGIR* (pp. 334–342).
81. Zhang, W., & Dietterich, T. (1995). A reinforcement learning approach to job-shop scheduling. *IJCAI* (pp. 1114–1120).
82. Zhang, Y., Xu, W., & Callan, J. (2003). Exploration and exploitation in adaptive filtering based on bayesian active learning, international conference on machine learning. *ICML* (pp. 896–903).
83. Zheng, Z., Chen, K., Sun, G., & Zha, H. (2007). A regression framework for learning ranking functions using relative relevance judgments. *ACM SIGIR* (pp. 287–294).
84. Zhu, X., Goldberg, A. B., Gael, J. V., & Andrzejewski, D. (2007). Improving diversity in ranking using absorbing random walks. *NAACL HTL* (pp. 97–104).
85. Ziegler, C. -N., McNee, S. M., Konstan, J. A., & Lausen, G. (2005). Improving recommendation lists through topic diversification. *WWW* (pp. 22–32).