A Reinforcement Learning Framework for Relevance Feedback

Ali Montazeralghaem
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
montazer@cs.umass.edu

Hamed Zamani
Microsoft
hazamani@microsoft.com

James Allan
Center for Intelligent Information Retrieval
University of Massachusetts Amherst
allan@cs.umass.edu

ABSTRACT

We present RML, the first known general reinforcement learning framework for relevance feedback that directly optimizes any desired retrieval metric, including precision-oriented, recall-oriented, and even diversity metrics: RML can be easily extended to directly optimize any arbitrary user satisfaction signal. Using the RML framework, we can select effective feedback terms and weight them appropriately, improving on past methods that fit parameters to feedback algorithms using heuristic approaches or methods that do not directly optimize for retrieval performance. Learning an effective relevance feedback model is not trivial since the true feedback distribution is unknown. Experiments on standard TREC collections compare RML to existng feedback algorithms, demonstrate the effectiveness of RML at optimizing for MAP and α-nDCG, and show the impact on related measures.

1 INTRODUCTION

Search queries are often too short and cannot precisely express the users' information needs. Sometimes a query is well specified but does not match the vocabulary of the collection being searched. Relevance feedback is one of the effective techniques to address these issues by utilizing feedback information for expanding queries. That is, given information from a handful of relevant documents, relevance feedback algorithms add terms to a query that better describe the information need within this collection and often provide weights indicating the relative importance of the new terms and the query terms. Relevance feedback information can be obtained through any of explicit feedback such as document relevance judgments [37], implicit feedback such as clickthrough data [18], or pseudo-relevance feedback when the system assumes that the top retrieved documents in response to a query are relevant [21, 23, 37, 51, 54].

Relevance feedback methods compute the feedback weight for each term using the same feedback information. However, ground truth feedback weights are unknown: an evaluation benchmark typically indicates which documents are relevant, but does not indicate which terms shoudl be in a query or what weights they shoudl have. Therefore, existing relevance feedback models make strong assumptions to estimate the feedback weight for each term. For instance, the divergence minimization model of Zhai and Lafferty [54] assumes that a term with high frequency in the feedback documents and low frequency in the collection should be assigned a high feedback weight. However, these assumptions are not necessarily inline with the ultimate goal of relevance feedback, i.e., improving retrieval performance.

Finding feedback weights that optimize ranking metrics is an important and challenging problem, partly because ranking metrics are *non-differentiable*, and because of the unknown optimal feedback weights. Recently, reinforcement learning (RL) has proven to be effective in various problems with non-differentiable metrics through policy gradient. For instance, successful results have been achieved in summarization [31, 34], machine translation [5, 34], and image captioning [34, 35] using RL methods.

In this study, we propose a reinforcement learning framework, namely RML, that learns a relevance feedback function as a stochastic policy network by selecting and weighting feedback terms in order to directly optimize the retrieval metrics as a reward function. RML is built on the REINFORCE algorithm [46] to achieve this objective. In our model, the policy network takes the query and the corresponding feedback documents and computes a probability distribution over the feedback terms.

It has been shown that relevance feedback models can benefit from various features, such as term proximity [24] and semantic similarity [48, 49]. Therefore, we model our policy network such that it can be easily extendable to new features. We propose a component-based model for the policy network whose components learn low-dimensional dense representations for any arbitrary query-dependent and query-independent feature. The output of the policy network is used by an agent to expand the query with respect to the feedback distribution. The agent's action is to sample K terms from the predicted distribution. This sampling helps the policy network to explore the search space and find effective feedback terms and their weights. Any arbitrary retrieval model (e.g., BM25 or query likelihood) can play the role of the environment in our RL setting. The retrieval performance is then used as a reward function to guide the policy network. RML is trained using a cross-entropy loss via a gradient-based optimization model that is guided by the reward function.

An important advantage of the reinforcement learning framework compared to the most existing methods is that it can be optimized for different evaluation metrics, such as average precision or normalized discounted cumulative gain (nDCG) [17]. It can even go beyond relevance-only metrics and, for example, optimize novelty and diversity metrics such as α-nDCG [8]. In the ideal case, the reward function can be easily modeled by various user satisfaction signals.

SIGIR '20, July 25–30, 2020, Xi'an, China

ACM ISBN 978-1-4503-5656-5/18/09…$15.00
https://doi.org/10.1145/3234944.3234973

We conduct experiments on standard newswire and web collections. The experimental results show that RML outperforms competitive baselines. Furthermore, we show that the model improves the diversity of the result list, especially when trained by optimizing α-nDCG. We perform extensive experiments to analyze each component of the model.

The primary contributions of this work can be summarized as follows:

- Designing a policy network as a relevance feedback function to directly optimize the retrieval metrics. Our policy network can be easily extended to new relevance matching features.
- Evaluating our model with two different reward functions to select and weight feedback terms in order to improve MAP and diversity of the top-rank documents.
- We show that our model outperforms competitive baselines using three TREC collections.

2 RELATED WORK

In this section, we first introduce methods used to improve the query language model in previous studies. We also review studies conducted to apply reinforcement learning in information retrieval.

2.1 Estimating Query Language Models

Previous studies try to improve a query language model by adding semantically related terms to the query. There are two groups of resources to select the related terms 1) global and 2) local. The methods that are in the first category try to use some global resources (e.g., Wikipedia and Dbpedia [2]) to expand the query. But methods in the second category are usually more popular and they commonly assume top retrieved documents in the first retrieval are relevant to the query and expand the query by important terms in these documents [21, 23, 36, 37, 40, 54]. These methods are also known as pseudo relevance feedback.

The Rocchio algorithm [37] is one relevance feedback method proposed for the vector space model which also used for PRF. This approach improves the query vector by maximizing its similarity to the top retrieved documents. Several PRF methods have been developed for language model so far. The relevance model [21] and mixture model [53] are two PRF methods which have been shown to be very effective. The mixture model is a model-based feedback strategy with the idea that terms in the feedback documents are drawn from two models: 1) topic model and 2) background model. It tries to separate the background model and topic model by using the EM algorithm and then use the topic model as the feedback model. Similar to the mixture model, the relevance model also tries to estimate $p(w|Q)$ which is the probability of w given the query as evidence. The relevance model estimated this probability by using information from feedback documents. Another model-based approach proposed for the language model framework is the Divergence Minimization Model (DMM) [54] which is based on minimization of the KL-divergence over feedback documents. Lv and Zhai [25] showed that although DMM satisfies most of the retrieval constraints, it performs poorly in experiments. They showed that DMM generates highly skewed feedback model. They introduced a maximum-entropy divergence minimization model (MEDMM) by proposing an entropy term to regularize DMM. They showed that MEDMM can improve the DMM's performance significantly. recently, Li et al. [22] proposed an end-to-end neural pseudo relevance feedback framework i.e., NPRF. However, their model do not directly optimize the retrieval performance.

The aforementioned methods usually use basic features of the feedback terms i.e., term frequency and document frequency. Recently some proposed methods try to use additional evidence i.e., semantic and proximity to PRF methods [15, 24, 27, 28, 50] that has been shown to be effective to improve the query language model.

2.2 Reinforcement Learning

Reinforcement learning (RL) is a framework of machine learning to optimize the behavior of an agent with respect to a desired reward [43]. Due to the progress of the deep learning, RL algorithms have demonstrated an impressive potential for tackling a wide range of complex tasks, from game playing [42] to robotic manipulation [33].

The main roles in RL are the agent and the environment. The environment is the world that is visible to the agent and that the agent can interact with. At every step of interaction, the agent sees observations in the environment and takes some actions based on these observations. The agent receives a reward according to its actions. The reward is a measure to show how good and bad are the actions taken by the agent. The agent's final goal is to maximize the cumulative reward.

RL algorithms have two branches: Model-free and Model-based RL. In the model-based RL approaches, the agent has access to a model of the environment. The main advantage of having a model is that it allows the agent to plan by thinking ahead. In most cases though, a ground-truth model of the environment is not available to the agent. For this reason, model-free RL approaches are more popular. There are two approaches to model-free RL algorithms: 1) Policy optimization and 2) Q-learning. Policy optimization can be used when a large and continuous action space is provided by the environment. On the other hand, when action space is small Q-learning is more promising.

Reinforcement learning in information retrieval also has recently attracted much attention. Zeng et al. [52] recently proposed a reinforcement learning approach for multi-page ranking in the learning to the rank framework. Wei et al. [45] proposed a novel learning to rank model on the basis of Markov decision process (MDP) MDPRank, which can directly optimize the evaluation measures and also leverage the evaluation measure at all ranking positions. Rosset et al. [38] proposed a novel approach based on reinforcement learning to reduce the accesses to the index blocks for pruning the results before showing to the user in the search engine.

Nogueira et al. [30] proposed an approach based on reinforcement learning for query reformulation which is most related to ours. They trained an agent to reformulate an initial query to maximize the expected return. However, their model concern semantic matching between query and a candidate term and cannot capture relevance matching signals such as term importance, document frequency of candidate terms in the feedback set and document length [9, 16]. Guo et al. [16] show that semantic matching based models cannot perform well in the ad-hoc retrieval tasks, especially when there is no access to large-scale training data. In contrast,

our model can learn the interaction between relevance matching signals and train even without access to large-scale training data.

3 LEARNING QUERY LANGUAGE MODEL

In this section, we first formulate the problem and define our motivation. Then, we propose a reinforcement learning framework for relevance feedback. Finally, we describe the architecture of the policy network.

3.1 Problem Statement and Motivation

Let $Q = \{q_1, q_2, ..., q_m\}$ be a query with m terms issued by a user u, and $F = \{D_1, D_2, ..., D_k\}$ be a set of k feedback documents. The feedback set F can be obtained from either explicit or implicit feedback from the user. In case of the absence of user feedback, we can assume that the top retrieved documents retrieved in response to the query are relevant. This assumption has been firstly made by Attar and Fraenkel [4] and by Croft and Harper [12] in the 1970s. It is commonly known as blind feedback or pseudo-relevance feedback (PRF) and has been shown to be effective in a wide variety of information retrieval settings [21, 51, 54].

Following previous work that has utilized relevance feedback information for query expansion and query term re-weighting [21, 51, 54], we also focus on estimating an accurate query model θ_Q^* for the query Q. Therefore, we assume that the retrieval model $\mathcal{M}$ is given. Without loss of generality, we focus on the language modeling framework [32] and use the KL-divergence retrieval model [20]. The retrieval score is computed as:

$$\text{score}_{\mathcal{M}}(\theta_Q^*, \theta_D) = -D(\theta_Q^* || \theta_D) = -\sum_{w \in V} p(w|\theta_Q^*) \log \frac{p(w|\theta_Q^*)}{p(w|\theta_D)}$$

where θ_D denotes the unigram language model of the document D over the vocabulary set V, estimated using maximum likelihood estimation smoothed with the Dirichlet prior smoothing method [55]. By fixing the retrieval model and the document language model estimation, the relevance feedback problem is now reduced to estimating an accurate query language model θ_Q^*.

The *ultimate goal* of relevance feedback[1] is to improve the retrieval performance. Therefore, the objective of relevance feedback is to optimize the following evaluation function:

$$\text{eval}(u, Q, \mathcal{M}(\theta_Q^*; C)) \tag{1}$$

where $\mathcal{M}$ and C denote the retrieval model and the document collection, respectively. The query language model θ_Q^* is estimated as:

$$\theta_Q^* = \mathcal{R}(Q, F; C, \Omega) \tag{2}$$

where $\mathcal{R}$ denotes the relevance feedback model with the parameter set Ω. The 'eval' function measures the performance of the given retrieval list (i.e., $\mathcal{M}(\theta_Q^*; C)$) with respect to the user's information need. Theoretically, this function can be defined based on different levels of relevance, ranging from algorithmic and topical relevance to motivational relevance. In its simplest form, the 'eval' function may measure topical relevance based on the ranking and set-based metrics such as precision of the top retrieved documents, average precision (AP), and normalized discounted cumulative gain (nDCG) [17]. It may go beyond relevance-only metrics, and for instance measure novelty and diversity, such as α-nDCG [8]. In the ideal case, it may measure user satisfaction and accomplishment based on multiple signals captured from user interaction with the system or questionnaires. In all of aforementioned cases, the function 'eval' is *non-differentiable*. Recently, reinforcement learning (RL) has proven to be effective in various problems with non-differentiable metrics through policy gradient.

Since the optimal query language model is unknown, typical supervised learning approaches cannot be employed to optimize the query language model estimation. Existing pseudo-relevance feedback methods, such as relevance models [21], mixture model [53], and divergence minimization model [54], do not directly optimize the ultimate objective, i.e., improving relevance. Their objectives are mostly equivalent to minimizing the distance between the query language model and the language model estimated for the feedback set, and sometimes try to distinguish the terms in the feedback set from those in the collection. In the rest of this section, we introduce a reinforcement learning framework for relevance feedback that directly optimizes the ultimate objective function, no matter how it is computed.

3.2 A Reinforcement Learning Framework for Relevance Feedback

This subsection introduces a relevance feedback framework, called RML, that directly optimizes the ultimate objective of relevance feedback, i.e., improving the retrieval performance. In more detail, given a training query set $T = \{Q_1, Q_2, \cdots, Q_n\}$, RML learns a relevance feedback model $\mathcal{R}$ with the parameter set Ω as follows:

$$\arg\max_{\Omega} \sum_{i=1}^{n} \text{eval}(u, Q_i, \mathcal{M}(\mathcal{R}(Q_i, F_i; C, \Omega); C)) \tag{3}$$

Optimizing the above objective using supervised learning is challenging since we do not know what is the ground truth relevance feedback model per query. In other words, the true distribution of the query model learned from the feedback set is unknown. To address this issue, we model relevance feedback as a reinforcement learning task. In reinforcement learning, there is an agent that at each timestamp takes an action and gets a reward signal from the environment, a number that tells the agent how good or bad was the taken action. The agent tries to figure out the best actions to take or the optimal way to behave in the environment in order to carry out his task in the best possible way to gain more reward.

In the following, we describe how we model the agent, the actions, and the reward function, such that the whole training process directly optimizes the above objective.

3.2.1 Training. In the following, we explain the training process of our model in detail.

Agent: At each timestamp in the training phase, the agent is a query modeling component. In our case, the agent expands the query and re-weights the query terms according to learned feedback distribution. The goal is to learn a feedback distribution that leads to an accurate query language model.

Policy Network: A policy is a rule used by an agent to decide what actions to take. At each timestamp t, the policy network

[1] Throughout this article, we often use "relevance feedback" as a reference to both true and pseudo relevance feedback settings.

will take the feedback documents and corresponding query and predict a probability distribution over the feedback terms $p(w|\theta_Q^t)$ where $\theta_Q^t = \mathcal{R}(Q, F; C, \Omega^t)$. We will discuss the policy network architecture in the rest of this section.

Agent's Action: Due to both efficiency and effectiveness reasons, the common practice is to only use a number of feedback terms to expand the query. Because of this reason, the agent's action is to sample K terms $S^t = \{w_1, w_2, \cdots, w_K\}$ from the query model distribution learned by $\mathcal{R}$ without replacement (i.e., $w \sim \theta_Q^t$, where $\theta_Q^t = \mathcal{R}(Q, F; C, \Omega^t)$) to get the actual feedback distribution. This results in the following probability distribution after normalization:

$$p'(w|\theta_Q^t) = \begin{cases} \frac{p(w|\theta_Q^t)}{Z} & \text{if } w \in S^t \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where Z is a normalization factor, i.e., $Z = \sum_{w \in S^t} p(w|\theta_Q^t)$. Note that although a natural decision is to take the top K terms with the highest probability, instead of sampling, we intentionally take samples to let the reinforcement learning agent explore. On the other hand, we take sample from the $p(w|\theta_Q^t)$ distribution, instead of uniform, let the model exploit.

The agent finally takes its *action* by linearly interpolating the sampled query model distribution $p'(w|\theta_Q^t)$ with the original query model estimated using maximum likelihood estimation. In other words, the agent produces the following query language model:

$$p(w|\widehat{\theta}_Q^t) = \alpha p(w|\theta_Q^{MLE}) + (1 - \alpha) p'(w|\theta_Q^t) \quad (5)$$

where $p(w|\theta_Q^{MLE}) = \frac{\text{count}(w,Q)}{|Q|}$ ($|Q|$ denotes the length of the given query), and α is a hyper-parameter controlling the weight of the original query model.

Environment: The environment for our RL agent is defined based on the retrieval collection C as well as the retrieval model $\mathcal{M}$, which is the KL-divergence retrieval model [20] in our case.

Reward Function: We compute the reward function based on the 'eval' function, which is our ultimate objective. To provide a useful signal to the model to learn effective relevance distribution, we define the reward function as follows:

$$\text{Reward}(Q, t) = \text{eval}(u, Q, \mathcal{M}(\widehat{\theta}_Q^t; C)) - \text{eval}(u, Q, \mathcal{M}(\widehat{\theta}_Q^{t-1}; C)) \quad (6)$$

In fact, our reward function provides information about the quality of relevance distribution at timestamp t compared to $t-1$. If the current reward is better than that at the last time step, the reward would be positive and vice versa.

Loss Function and Optimization: We adapt a cross-entropy loss function that has been widely used in the reinforcement learning literature [31]. Assuming training queries in the training set T are drawn i.i.d, the loss function at each timestamp t is defined as:

$$\mathcal{L}^{(t)} = -\sum_{Q \in T} \text{Reward}(Q, t) \cdot \sum_{w \in S^t} p'(w|\theta_Q^t) \log p(w|\theta_Q^t) \quad (7)$$

where $p(w|\theta_Q^t)$ is the predicted feedback distribution and $p'(w|\theta_Q^t)$ is the actual feedback distribution. Multiplying the cross-entropy formula by the reward function pushes the model towards the right direction. In fact, if the model achieves a positive reward function, our RL algorithm tries to decrease the divergence between the model's output and the expanded distribution that led to a positive reward. In contrast, negative reward value helps the model to diverge from the current expanded query.

We use REINFORCE algorithm [46], one of the policy gradient methods as the optimization algorithm. In more details, for each episodes, the parameters of the policy network are updated as follows:

$$\Omega^t \leftarrow \Omega^{t-1} + \alpha \nabla \mathcal{L}^{(t)} \quad (8)$$

where α is the learning rate. By optimizing the loss function using gradient descent-based algorithms, the model parameters (i.e., Ω) will be appropriately updated. In our experiments, we use the Adam optimizer [19] that has been shown to be effective in reinforcement learning [5, 31]. Note that the parameter set Ω is randomly initialized. Indeed, we start with a random relevance feedback model and update its parameters, such that the retrieval performance increases.

3.2.2 Query Expansion at Test Time. At test time, we simply use the trained relevance feedback model $\mathcal{R}$ (i.e., θ_Q^*) for estimating the query language model for any arbitrary test query Q. Following previous work on pseudo-relevance feedback [21, 51, 54], we only keep the K terms with the highest probability and interpolate this normalized distribution with the original query language model as follows:

$$p(w|\widehat{\theta}_Q^*) = \alpha p(w|\theta_Q^{MLE}) + (1 - \alpha) p(w|\theta_Q^*) \quad (9)$$

where α controls the weight of original query in the final query model (i.e., $\widehat{\theta}_Q^*$).

3.3 Policy Network Architecture

We implement the relevance feedback model $\mathcal{R}$ using neural network. The reason for making this decision is that (1) neural networks can be easily trained as part of our reinforcement learning framework using the back-propagation algorithm [39], (2) they can be easily extended to handle various features, (3) they have recently produced promising results for various information retrieval settings, ranging from basic and fundamental tasks, such as ad-hoc retrieval [13, 16] and question answering [41], to more recent and trending tasks, such as mobile search [3] and information-seeking conversations [47].

Instead of designing a complex neural network with huge number of parameters, we intentionally design our model with few parameters (i.e., less than 300 parameters) to make sure that the model can perform effectively, even without access to large-scale training data. Figure 1 is an overview of the architecture of the policy network in our model. We empirically validate our claim by only experimenting on standard TREC collections that only contain a few hundred queries. Because of this reason, we cast the problem of estimating a relevance feedback distribution to estimating a relevance feedback weight for each term using a softmax operator, as follows:

$$p(w|\mathcal{R}) = \frac{\exp(\mathcal{R}'(w, Q, F; C, \Omega))}{\sum_{w' \in V} \exp(\mathcal{R}'(w', Q, F; C, \Omega))} \quad (10)$$

In other words, we train a single model $\mathcal{R}'$ that takes information related to a candidate feedback term as input and produces a feedback weight. This model is applied to all vocabulary terms. $\mathcal{R}'$

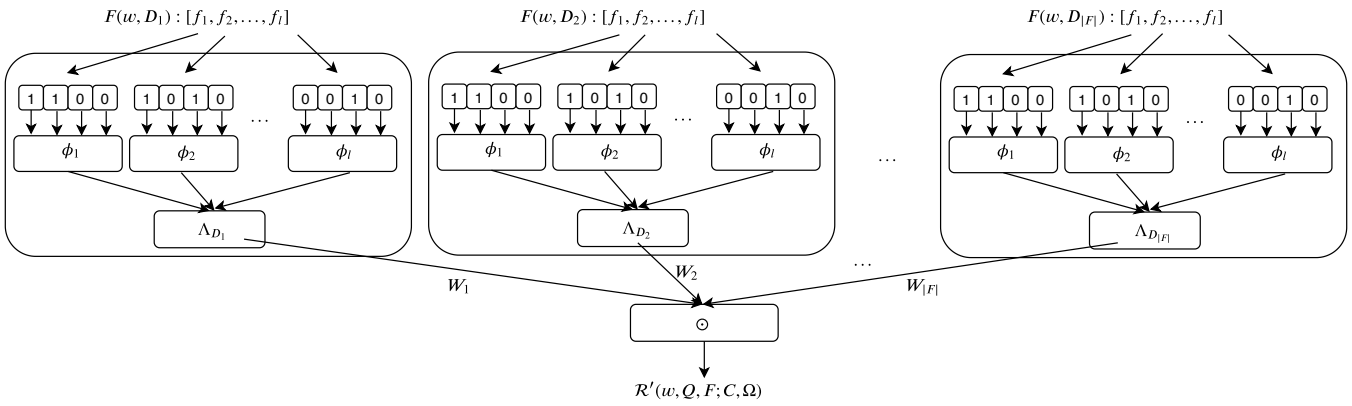

Figure 1: The architecture of policy network.

uses a component-based architecture that consists of a set of sub-networks, each modeling an aspect of relevance feedback. They are expected to produce complimentary information and final feedback weight is computed using an aggregation sub-network. Formally, we compute the feedback weight of each term w as follows:

$$\mathcal{R}'(w, Q, F; C, \Omega) = \odot_{i=1}^{|F|}(\Lambda(\phi_1(w, Q, D_i; C), \cdots, \phi_l(w, Q, D_i; C)), \mathcal{W}_i) \tag{11}$$

where $\phi_1, \phi_2, \cdots, \phi_l$ denote l components, each learning a representation for each feedback term in a feedback document. The component Λ aggregates these l representations. $\mathcal{W}_i$ denotes the relevance score of the document D_i to the query Q. Finally, the compositionality component $\odot$ aggregates all the representations obtained from all feedback documents and produces a single real number as the weight of the feedback term w. In the following, we describe how each component is developed.

The representation learning components (ϕ_is): The relevance feedback literature [21, 23, 25, 37, 51, 54] suggests that term statistics information, such as term frequency in the feedback documents and collection frequency, is essential in feedback models. This has been also validated by a set of theoretical studies based on axiomatic analysis [9, 28, 29]. For simplicity as well as fair comparison, we only use the features that are used by existing relevance feedback models, such as the Lavrenko and Croft's relevance models [21]. These features include:

- tf(w, D): term frequency in the feedback document D
- idf(w): inverse document frequency in the collection
- $|D|$: length of the feedback document
- $df(w)$: document frequency in the feedback set

However, our experiments show that using these features as inputs of the network does not perform effectively. This is due to the varying scale of these features [13]. The common feature normalization methods, such as max or z-score normalization, do not solve the issue. Therefore, we propose to use these features in a binary format. For example, assume that the frequency of a term in a feedback document is 27. We first encode this feature to binary (11011). Since the back-propagation algorithm assumes that inputs of the network are zero-mean, we compute the following feature vector by subtracting 0.5 from each digit: $[0.5, 0.5, -0.5, 0.5, 0.5]$. Let $B = \langle b_1, b_2, \cdots, b_p \rangle$ represent this binary encoding which is normalized. The simplest network to learn representation for this feature vector is probably a fully-connected network. However, since the binary digits are not independent, a fully-connected network does not perform well, which is also verified by our experiments. In other words, the i^{th} digit carries two times higher weight than the $(i+1)^{\text{th}}$ digit. In fact the weight of digit i is higher than the summation of all the weights corresponding to digits $(i+k)$ for $(1 \leq k \leq p-i)$. Therefore, instead of $\sigma(\sum_{i=1}^{p} b_i w_i + \mathfrak{b})$ which is used in fully-connected layers, we model the first layer of this sub-network as follows:

$$z_1 = \sigma(\sum_{i=1}^{p} b_i(w_i + w_{(i+1)} + \ldots + w_p) + \mathfrak{b}) \tag{12}$$

where w_i is the weight of the i^{th} digit and $\mathfrak{b}$ is a bias term. σ is an activation function (i.e., ReLU in our experiments) to let neural network learn non-linear function. The output of the first layer can then be fed to fully-connected layers.

The aggregation component (Λ): This component feeds the concatenation of the outputs of the representation learning components (ϕ_is) to a fully-connected network. The output of this component is a d-dimensional vector that is learned for each term in each feedback document.

The compositionality component ($\odot$): $|F|$ d-dimensional vectors are the outputs of the aggregation component for each term. These vectors are multiplied by the relevance score of each document $\mathcal{W}_i$ and are concatenated. This weighted concatenation is fed into a fully-connected network to produce a single real number as the weight of the feedback term.

4 EXPERIMENT

Datasets: We use three standard test collections in our experiments. Robust collection consists of thousands of news articles and is considered a homogeneous collection. Robust was used in TREC 2004 Robust Track. We also use two large scale web collections (GOV2 and ClueWeb) containing heterogeneous documents. GOV2 consists of the .gov domain web pages, crawled in 2004. ClueWeb

Table 1: Summary of TREC collections and topics.

Collection	Genre	Queries	#docs
Gov2	webpages	701-850	25m
Robust04	news	301-450 & 601-700	0.5m
ClueWeb	webpages	1-200	50m

(i.e., ClueWeb09-Category B) is a common web crawl collection that only contains English web pages. GOV2 and ClueWeb were previously used in TREC 2004-2006 Terabyte Track and TREC 2009-2012 Web Track, respectively. The statistics of these collections are reported in Table 1. We only used the title of the topics as queries. We cleaned the ClueWeb collection by filtering out the spam documents using the Waterloo spam scorer [1] with a threshold of 60% [11]. Stop words were removed from all collections using the standard INQUERY stopword list and all documents were stemmed using the Porter stemmer.

Experimental and parameter setting: All models are implemented using TensorFlow [1, 44]. We optimized the parameter of the network using the Adam optimizer [19] which uses the back-propagation algorithm to compute the gradients. Tuning of the hyper-parameters of the network is done on the respective validation sets. The learning rate in our experiments was selected from $\{1e-3, 5e-4, 1e-4\}$. One or two hidden layers were used for each sub-network of our model. The size of each hidden layer was selected from $\{2, 5, 10, 15\}$. We sweep the number of feedback terms between $\{5, 10, 15, 20, 25\}$ and the feedback coefficient between $\{0.0, 0.2, ..., 1.0\}$. We performed 5-fold cross-validation over the queries in each collection for tuning the hyper-parameters of our model as well as baselines. For regularization, we find that the early stopping strategy [7] works well for our model.

Note that at inference and training times, we take the top 1000 retrieved documents (Not all documents in the collections) using the query likelihood retrieval model as candidate documents and re-rank them. Diaz [14] showed that re-ranking the top 1000 documents is as effective as retrieving from the entire collection at a much lower cost.

Evaluation metrics: To evaluate retrieval performance, we use mean average precision (MAP) of the top 1000 documents as the main evaluation metric. In addition, we also report precision of the top 10 and 20 retrieved documents (P@10 and P@20) and normalized discounted cumulative gain (nDCG) [17] computed for the top 20 retrieved documents (nDCG@20). To evaluate the robustness of the proposed method, we use the robustness index (RI) [10] which is defined as $\frac{N_+ - N_-}{|Q|}$, where $|Q|$ is the number of queries. N_+ and N_- denote the number queries which improved or degraded by the feedback model. The value of RI is always in the $[-1, 1]$ interval and methods with higher values are more robust[2]. Statistically significant differences of performance are determined using the two-tailed paired t-test at a 95% confidence level ($p\_value < 0.05$).

Baselines: Our baseline methods include 1) the standard language model without feedback (LM), 2) the relevance model (RM3) [21] which is an effective and unsupervised PRF model, 3) a supervised approach for learning query language model (SL) [6]. They assume that feedback terms contribute independently to the retrieval performance, so as they mentioned, we train a binary classifier to find a term if the retrieval performance increases beyond a preset threshold when that term is added to the original query. More formally, we consider a term as relevant if $(R^{'} - R)/R > 0.005$ where R and $R^{'}$ are the retrieval performance without and with expanding the query by the term. 4) the maximum entropy divergence minimization model (MEDMM) [25] which is an effective PRF model by modifying the divergence minimization model (DMM) [54]. MEDMM is also an unsupervised approach for PRF. We do not consider other methods since the performance of the chosen methods is significantly better than other models such as [23, 25].

There are a number of PRF methods that considered additional evidence e.g., term dependencies [26], term proximity [24, 27] and semantic similarity [28, 48]. Since we do not use the additional evidence in our model (see section 3.3) we do not consider these methods in our experiments to avoid unnecessary apples-to-oranges comparisons.

4.1 Results and Discussion

In this section, we discuss several research questions that are needed to be addressed and for each, we present a set of experiments along with their results and analysis to address the research questions.

RQ1: How effective is RML compared to the baselines?

In this experiment, we use AP in our reward function for training which is denoted as RML. The results obtained by the proposed method and the baselines are reported in Table 2. To have a fair evaluation, in this experiment we do not compare our model against relevance feedback methods that considered additional evidence e.g., term dependencies [26], term proximity [24, 27] and semantic similarities [28, 48].

According to this table, RML outperforms all the baselines in terms of MAP, P@10, P@20, and RI in all the collections, except in one case. The statistical t-test shows that the MAP, P@10, and P@20 improvements over LM are always significant. These improvements over the state-of-the-art baseline feedback models are also significant in terms of MAP which RML tries to maximize. These results show the effectiveness of the proposed method compared to the strong relevance feedback methods. MAP measure intrinsically is related to other measures e.g., P@10, P@20, and RI. Therefore, maximizing MAP can also help P@10, P@20, and RI improve but this improvement is not significant in all cases over the baselines. In addition to the baseline methods and RML, we report an Oracle performance bound in the last column. The Oracle is RML method when trained on the test set.

RQ2: Which feedback terms are selected by RML and baselines? **(Exploration)**

To answer this question, we analyze the terms chosen by each model. For each feedback term, we first compute the total number of occurrences of the term in the feedback set (i.e., $TF(w) = \sum_{D \in F} count(w, D)$), its document frequency in the feedback set (i.e.,

[1] http://plg.uwaterloo.ca/ gvcormac/clueweb09spam/

[2] To avoid the influence of very small performance changes in the RI values, we only consider improvements/losses higher than 10% relatively.

Table 2: Comparison of proposed methods (trained to maximize MAP) and baselines. Superscripts 0/1/2/3 indicate that the improvements over LM/RM3/MEDMM/SL are significant. The highest non-oracle value in each row is marked in bold

Dataset	Metric	LM	RM3	MEDMM	SL	RML	Oracle
Gov2	*MAP*	0.2992	0.3123	0.3129	0.3138	**0.3284^{0123}**	0.3567
	P@10	0.5437	0.5462	0.5821	0.5495	**0.5855^{013}**	0.6344
	P@20	0.5198	0.5257	0.5406	0.5263	**0.5569^{0123}**	0.6104
	nDCG@20	0.4721	0.4677	0.4851	0.4677	**0.5024^{0123}**	0.5454
	RI	-	0.0540	0.0366	0.0409	**0.1571**	0.3716
Robust04	*MAP*	0.2442	0.2754	0.2737	0.2762	**0.2845^{0123}**	0.3168
	P@10	0.4242	0.4460	0.4486	0.4502	**0.4568^{01}**	0.5190
	P@20	0.3605	0.3736	0.3789	0.3712	**0.3875^{013}**	0.4283
	nDCG@20	0.4290	0.4455	0.4517	0.4480	**0.4604^{013}**	0.5144
	RI	-	0.2393	**0.2771**	0.2510	0.2588	0.5144
ClueWeb	*MAP*	0.1001	0.1055	0.1073	0.1061	**0.1093^{0123}**	0.1140
	P@10	0.3145	0.3365	0.3467	0.3412	**0.3493^{01}**	0.3565
	P@20	0.2955	0.3122	0.3159	0.3112	**0.3182^{0}**	0.3330
	nDCG@20	0.2587	0.2714	0.2768	0.2750	**0.2819^{0}**	0.2925
	RI	-	0.0850	0.1107	0.0720	**0.1602**	0.2050

Table 3: Statistics of terms extracted by different models

Dataset	Statistics	SL	RM3	MEDMM	RML	Oracle
Gov2	$\mu(tf)$	145.29	146.17	44.20	68.53	67.55
	$\mu(df)$	8.12	8.16	7.19	7.53	7.48
	$\mu(idf)$	2.67	2.66	4.70	3.43	3.33
	$\mu(dl)$	4203.84	4195.38	4368.09	4364.78	4385.28
Robust04	$\mu(tf)$	34.20	38.49	14.05	27.04	20.22
	$\mu(df)$	6.71	6.60	5.13	6.05	5.52
	$\mu(idf)$	2.67	2.54	4.09	3.06	3.35
	$\mu(dl)$	1354.39	1371.82	1405.54	1369.19	1374.44
ClueWeb	$\mu(tf)$	112.80	109.51	26.17	41.29	43.38
	$\mu(df)$	7.48	7.72	6.63	6.82	6.84
	$\mu(idf)$	3.32	3.08	5.42	4.77	4.59
	$\mu(dl)$	2272.05	2221.78	2293.54	2296.75	2301.63

$DF(w) = \sum_{D \in F} I(count(w, D) > 0)$ where $I(.)$ is an indicator function), its inverse document frequency in the collection (i.e., $IDF(w)$) and the length of the feedback documents that contain this term (i.e., $DL(w) = \sum_{D \in F} I(count(w, D) > 0) * |D|$). We then averaged these quantities overall feedback terms and queries. For example, the mean tf is computed as $\mu(tf) = \frac{1}{|Q|} \sum_{q \in Q} \sum_{i=1}^{tc} \frac{TF(w)}{tc}$ where $|Q|$ is the number of the queries and tc the number of feedback terms added to the query. Table 3 shows the above statistics. According to this table, the SL and RM3 models select feedback terms that have small *IDF*, and relatively high *TF* and *DF*. MEDMM, on the other hand, selects feedback terms that have high *IDF* and relatively small *TF* and *DF*. Therefore, in contrast to SL and RM3, MEDMM focuses more on words that are not too common (high *IDF* and small *TF*) but an advantage of SL and RM3 is that they try to select terms that occur in an acceptable number of feedback documents (high *DF*). Selected feedback terms by RML are not too general (relatively small *TF* and high *IDF*) but still occur in a sufficient number of feedback documents. We also report the same statistics for feedback terms selected by Oracle in this table.

To get a sense of what is learned by RML and RM3, in Table 7 we report the top 10 expansion terms for a sample query. According to these tables, the terms added to the query by the RML model are more relevant to individual query terms. In this example, for the query "dinosaur", the feedback terms that are selected by RML are kinds of dinosaurs and RML learns to select and weight them because these terms help the model to get more reward. But RM3 select feedback terms with high *TF* and *DF* that do not necessarily help to improve the retrieval performance. Therefore, RM3 selects terms that obviously are not related to these queries in most cases.

Table 4: Weighing selected feedback terms by RML with RM3

	MAP	P@10	P@20	NDCG@20	RI
RM3	0.3123	0.5462	0.5257	0.4677	0.0540
RM3 + RML	0.3211	0.5755	0.5489	0.4896	0.0885
RML	0.3284	0.5855	0.5569	0.5024	0.1571

***RQ3: How effective is RML in weighing feedback terms?* (Exploitation)**

As mentioned before, RML initially tries to select good terms among other feedback terms and then weights the selected terms in an appropriate way to get more reward. To analyze the exploitation task in the RML model we chose the feedback terms that are selected by RML and weight them by RM3. The results of this experiment reported in Table 4. According to this table, although using selected feedback terms by RML improves the RM3 performance but still RML outperform. This shows that RML not only is effective in finding relevant feedback terms (exploration) but also can effectively weight them (exploitation).

Table 5: Effect of each feature of proposed model

	MAP	P@10	P@20	nDCG@20	RI
RM3	0.3123	0.5462	0.5257	0.4677	0.0540
TF	0.3182	0.5648	0.5356	0.4777	0.0904
TF (weighted document)	0.3230	0.579	0.5503	0.4973	0.1147
TF (weighted document) + DocLen	0.3252	0.5822	0.5495	0.4946	0.1100
TF (weighted document) + DocLen + IDF	**0.3284**	**0.5855**	**0.5569**	**0.5024**	**0.1571**

Table 6: Maximizing RML with α-nDCG to improve diversification and novelty on ClueWeb only. Superscripts 0/1/2 indicate that the improvements over LM/RM3/RML are significant. The highest value in each row is marked in bold

Method	Measure					
	MAP	*P@10*	*P@20*	*nDCG@20*	*RI*	*α-nDCG@20*
LM	0.1001	0.3145	0.2955	0.2587	-	0.3497
RM3	0.1055	0.3365	0.3122	0.2714	0.0850	0.3569
RML	**0.1093**	**0.3493**	**0.3182**	**0.2819**	**0.1602**	0.3574
RML $_{\alpha-nDCG@20}$	0.1057	0.3421	0.3044	0.2759	0.0342	$\mathbf{0.3665^{012}}$

Table 7: The top terms added to the query "dinosaur" (topic 14) by RM3 and RML methods.

RM3	Weight	RML	Weight
price	0.1611	tyrannosaurus	0.1185
parti	0.1377	stegosaurus	0.1132
regular	0.0788	jurass	0.0982
99	0.0574	triceratop	0.0532
rex	0.0506	skull	0.0512
dino	0.0474	prehistor	0.0488
toy	0.0447	mammal	0.0477
birthday	0.0388	skeleton	0.0467
prehistor	0.0333	bone	0.0392
game	0.0326	plush	0.0372

Table 8: The top terms added to the query "403b" (topic 151) by RML $_{\alpha-nDCG@20}$ and RML (feedback terms are stemmed by Porter).

Subtopic 1:	What is a 403b plan?
Subtopic 2:	What is the difference between 401k and 403b retirement plans?
Subtopic 3:	What are the withdrawal limitations for a 403b retirement plan?

RML $_{\alpha-nDCG@20}$	Weight	RML	Weight
401k	0.3817	choos	0.1645
limit	0.1062	portion	0.1563
portion	0.0973	roth	0.1301
guidanc	0.0743	ira	0.1222
requir	0.0557	rollov	0.0761
ira	0.0468	individu	0.0585
incom	0.0397	retir	0.0492
offer	0.0336	estat	0.0420
rollov	0.0328	organ	0.0318
individu	0.0180	minor	0.0277

RQ4: How effective is each feature in RML?

To study the effect of each feature in the retrieval performance, we add one feature to RML at a time. The results obtained by this experiment are reported in Table 5. According to this table, RML beats RM3 and also other baselines (Table 2) even if it only uses the term frequency (TF) of feedback terms. This shows that RML can learn an appropriate representation for term frequency. In addition, RML performance improves when the document length of a feedback document and the inverse document frequency (IDF) of the term are added to the model.

RQ5: How does the reward function affect the retrieval performance?

Diversification and Novelty: As mentioned before, different reward functions can be used in RML. In this experiment, we use a different measure for evaluation that used for diversification in previous work. Specifically, we add α-nDCG [8] to our evaluation as a reward function and try to maximize it in RML. Note that this experiment is done on the ClueWeb collection since there are no diversification judgments of the other two collections. The results obtained by this experiment are reported in Table 6. Interestingly, RML $_{\alpha-nDCG@20}$ outperforms compared the baselines in terms of α-nDCG when it tries to maximize this metric. In other words, in this case, RML $_{\alpha-nDCG@20}$ selects and weights terms in order to improve the diversification in the retrieval. According to this table, we can see that although α-nDCG improves, the other measures decrease. That is because the relation between α-nDCG and other measures is less than MAP and other measures. Maximizing α-nDCG can hurt other measures.

As an example, we report top 10 terms that are added to the query "403b" by RML which is maximized with MAP and α-nDCG in Table 8. Three subtopics of this query also are shown in this table. According to this table, the selected terms by RML using α-nDCG as the reward function are more diverse. For example, "401k" obviously is a term relevant to the second subtopic, "limit" can be relevant to the third subtopic and other terms are more relevant to the first subtopic. In contrast, RML that is maximized with MAP, find terms that are more relevant to the first subtopic.

Parameter Sensitivity: To capture the sensitivity of the RML to the number of feedback terms and the feedback interpolation coefficient we plot the MAP values achieved by sweeping these parameters in Figure 2. Appropriate selection of the number of feedback terms for Robust collection is more important and 15 terms can be added to the query to get the best result in this collection. For the interpolation coefficient parameter, our experiments show that 0.5 or 0.6 for Rebuts and Gov2 collections is suitable and for ClueWeb collection 0.8 is better.

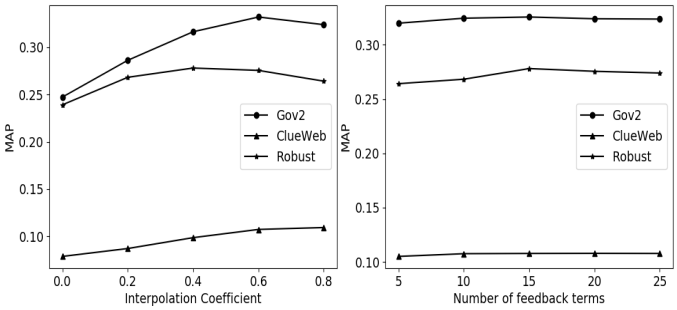

Figure 2: Sensitivity of the RML method to the number of feedback terms (right) and the interpolation coefficient (left).

Learning Curve: Figure 3 shows the learning curve of RML in terms of MAP. From the figure, we can see the ranking accuracy of RML improves, as the training goes on, until the accuracy on the validation and the test sets reach to the peak. As mentioned before, we find that the early stopping [7] strategy works well for our model.

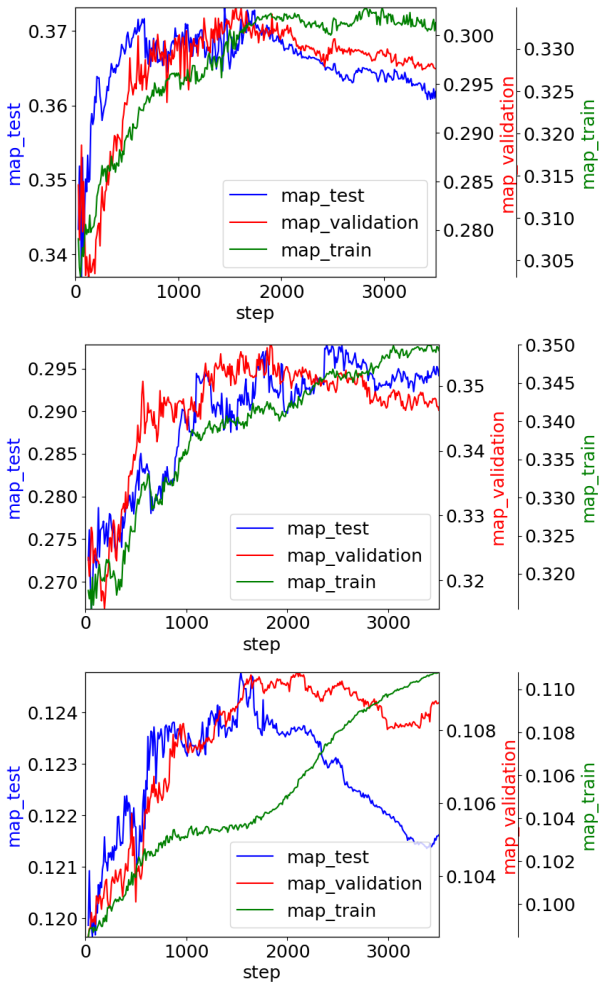

Figure 3: Learning curves for Robust (top), Gov2 (middle), and ClueWeb (bottom).

5 CONCLUSIONS AND FUTURE WORK

We proposed a novel reinforcement algorithm for learning relevance feedback model, referred to as RML. In contrast to the most existing models, RML directly optimizes retrieval metrics, including average precision for effective retrieval and α-nDCG for diverse retrieval. Furthermore, since the existing feedback models are built upon some underlying theoretical framework, it is not always possible to easily incorporate new features.

RML employs a policy network to learn the relevance feedback function. Based on our results, RML can effectively explore and exploit the solution space to learn an effective feedback function. According to our results, RML select feedback terms that are not too general (i.e., feedback terms that have small *TF* and high *IDF*) but still occur in a sufficient number of feedback documents. Experimental results on three TREC datasets show that RML significantly outperforms the baseline methods. RML finds and weights diverse feedback terms when trained by optimizing α-nDCG.

In this study, we learn a query model in order to maximize the retrieval performance but we do not modify the retrieval model. As future work, the agent could also learn to find the similarity of the query and document model in the main retrieval. Proposing an effective reward function to consider two or more retrieval metrics at the same time is another aspect of future work.

6 ACKNOWLEDGEMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant #IIS-1617408. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Martín Abadi and Agarwal A Barham P TensorFlow. 2016. Large-scale machine learning on heterogeneous distributed systems. In *OSDIâĂŹ16*. 265–283.

[2] Nitish Aggarwal and Paul Buitelaar. 2012. Query Expansion Using Wikipedia and Dbpedia.. In *CLEF (Online Working Notes/Labs/Workshop)*.

[3] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2018. Target Apps Selection: Towards a Unified Search Framework for Mobile Devices. In *SIGIR '18*. 215–224.

[4] R. Attar and A. S. Fraenkel. 1977. Local Feedback in Full-Text Retrieval Systems. *J. ACM* 24, 3 (1977), 397–417.

[5] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2016. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086* (2016).

[6] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting good expansion terms for pseudo-relevance feedback. In *SIGIR '08*. ACM, 243–250.

[7] Rich Caruana, Steve Lawrence, and C Lee Giles. 2001. Overfitting in neural nets: Backpropagation, conjugate gradient, and early stopping. In *Advances in neural information processing systems*. 402–408.

[8] Charles LA Clarke, Maheedhar Kolla, Gordon V Cormack, Olga Vechtomova, Azin Ashkan, Stefan Büttcher, and Ian MacKinnon. 2008. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*. ACM, 659–666.

[9] Stéphane Clinchant and Eric Gaussier. 2013. A theoretical analysis of pseudo-relevance feedback models. In *ICTIR '13*. ACM, 6.

[10] Kevyn Collins-Thompson. 2009. Reducing the risk of query expansion via robust constrained optimization. In *CIKM '09*. ACM, 837–846.

[11] Gordon V. Cormack, Mark D. Smucker, and Charles L. Clarke. 2011. Efficient and Effective Spam Filtering and Re-ranking for Large Web Datasets. *Inf. Retr.* 14, 5 (2011), 441–465.

[12] W. B. Croft and D. J. Harper. 1979. Using Probabilistic Models of Document Retrieval Without Relevance Information. *J. of Documentation* 35, 4 (1979), 285–295.

[13] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. 2017. Neural Ranking Models with Weak Supervision. In *SIGIR'17*. 65–74.

[14] Fernando Diaz. 2015. Condensed list relevance models. In *ICTIR '15*. ACM, 313–316.

[15] Fernando Diaz, Bhaskar Mitra, and Nick Craswell. 2016. Query expansion with locally-trained word embeddings. *arXiv preprint arXiv:1605.07891* (2016).

[16] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *CIKM'16*. 55–64.

[17] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems (TOIS)* 20, 4 (2002), 422–446.

[18] Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. 133–142.

[19] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *ICLR'15*.

[20] John Lafferty and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *SIGIR '01*. ACM, 111–119.

[21] Victor Lavrenko and W. Bruce Croft. 2001. Relevance Based Language Models. In *SIGIR'01*. 120–127.

[22] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. 2018. NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. *arXiv preprint arXiv:1810.12936* (2018).

[23] Yuanhua Lv and ChengXiang Zhai. 2009. A Comparative Study of Methods for Estimating Query Language Models with Pseudo Feedback. In *CIKM'09*.

[24] Yuanhua Lv and ChengXiang Zhai. 2010. Positional relevance model for pseudo-relevance feedback. In *SIGIR '10*. ACM, 579–586.

[25] Yuanhua Lv and ChengXiang Zhai. 2014. Revisiting the divergence minimization feedback model. In *CIKM '14*. ACM, 1863–1866.

[26] Donald Metzler and W Bruce Croft. 2007. Latent concept expansion using markov random fields. In *SIGIR '07*. ACM, 311–318.

[27] Jun Miao, Jimmy Xiangji Huang, and Zheng Ye. 2012. Proximity-based rocchio's model for pseudo relevance. In *SIGIR '12*. ACM, 535–544.

[28] Ali Montazeralghaem, Hamed Zamani, and Azadeh Shakery. 2016. Axiomatic analysis for improving the log-logistic feedback model. In *SIGIR '16*. ACM, 765–768.

[29] Ali Montazeralghaem, Hamed Zamani, and Azadeh Shakery. 2018. Theoretical Analysis of Interdependent Constraints in Pseudo-Relevance Feedback. In *SIGIR '18*. ACM, 1249–1252.

[30] Rodrigo Nogueira and Kyunghyun Cho. 2017. Task-oriented query reformulation with reinforcement learning. *arXiv preprint arXiv:1704.04572* (2017).

[31] Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304* (2017).

[32] Jay M. Ponte and W. Bruce Croft. 1998. A Language Modeling Approach to Information Retrieval. In *SIGIR'98*. 275–281.

[33] Ivaylo Popov, Nicolas Heess, Timothy Lillicrap, Roland Hafner, Gabriel Barth-Maron, Matej Vecerik, Thomas Lampe, Yuval Tassa, Tom Erez, and Martin Riedmiller. 2017. Data-efficient deep reinforcement learning for dexterous manipulation. *arXiv preprint arXiv:1704.03073* (2017).

[34] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732* (2015).

[35] Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7008–7024.

[36] Stephen E. Robertson and Karen Sparck Jones. 1988. Document Retrieval Systems. Taylor Graham Publishing, London, UK, UK, Chapter Relevance Weighting of Search Terms, 143–160. http://dl.acm.org/citation.cfm?id=106765.106783

[37] J. J. Rocchio. 1971. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*. Prentice Hall, 313–323.

[38] Corby Rosset, Damien Jose, Gargi Ghosh, Bhaskar Mitra, and Saurabh Tiwary. 2018. Optimizing Query Evaluations using Reinforcement Learning for Web Search. *arXiv preprint arXiv:1804.04410* (2018).

[39] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *nature* 323, 6088 (1986), 533.

[40] Gerard Salton and Chris Buckley. 1990. Improving retrieval performance by relevance feedback. *Journal of the American society for information science* 41, 4 (1990), 288–297.

[41] Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *SIGIR '15*. ACM, 373–382.

[42] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, and others. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354.

[43] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction*. MIT press.

[44] Yuan Tang. 2016. TF. Learn: TensorFlow's high-level module for distributed machine learning. *arXiv preprint arXiv:1612.04251* (2016).

[45] Zeng Wei, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2017. Reinforcement learning to rank with Markov decision process. In *SIGIR '17*. ACM, 945–948.

[46] Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning* 8, 3-4 (1992), 229–256.

[47] Liu Yang, Minghui Qiu, Chen Qu, Jiafeng Guo, Yongfeng Zhang, W Bruce Croft, Jun Huang, and Haiqing Chen. 2018. Response Ranking with Deep Matching Networks and External Knowledge in Information-seeking Conversation Systems. *arXiv preprint arXiv:1805.00188* (2018).

[48] Hamed Zamani and W Bruce Croft. 2016. Embedding-based query language models. In *ICTIR '16*. ACM, 147–156.

[49] Hamed Zamani and W. Bruce Croft. 2017. Relevance-based Word Embedding. In *SIGIR'17*. 505–514.

[50] Hamed Zamani and W Bruce Croft. 2017. Relevance-based word embedding. In *SIGIR '17*. ACM, 505–514.

[51] Hamed Zamani, Javid Dadashkarimi, Azadeh Shakery, and W. Bruce Croft. 2016. Pseudo-Relevance Feedback Based on Matrix Factorization. In *CIKM'16*.

[52] Wei Zeng, Jun Xu, Yanyan Lan, Jiafeng Guo, and Xueqi Cheng. 2018. Multi Page Search with Reinforcement Learning to Rank. In *SIGIR '18*. ACM, 175–178.

[53] Chengxiang Zhai and John Lafferty. 2001. Model-based feedback in the KL-divergence retrieval model. In *CIKM '01*. 403–410.

[54] Chengxiang Zhai and John Lafferty. 2001. Model-based Feedback in the Language Modeling Approach to Information Retrieval. In *CIKM'01*. 403–410.

[55] Chengxiang Zhai and John Lafferty. 2004. A Study of Smoothing Methods for Language Models Applied to Information Retrieval. *ACM Trans. Inf. Syst.* 22, 2 (2004).