Relevance and Reinforcement in Interactive Browsing

Anton Leuski
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts
Amherst, MA 01003 USA
leuski@cs.umass.edu

ABSTRACT

We consider the problem of browsing the top ranked portion of the documents returned by an information retrieval system. We describe an interactive relevance feedback agent that analyzes the inter-document similarities and can help the user to locate the interesting information quickly. We show how such an agent can be designed and improved by using neural networks and reinforcement learning. We demonstrate that its performance significantly exceeds the performance of the traditional relevance feedback approach.

Categories and Subject Descriptors

H.3.3 [**Information storage and retrieval**]: Information Search and Retrieval—*Relevance feedback, Selection process*; H.5.m [**Information Interfaces and Presentation**]: Miscellaneous; I.2.6 [**Artificial Intelligence**]: Learning—*Connectionism and neural nets*

General Terms

Experimentation, performance, algorithms

1. INTRODUCTION

Helping the user to locate interesting information among the retrieved material is almost as important as the retrieval itself. An information retrieval system responds to the user's query with a large set of documents and leaves the user to find the relevant information among these documents by herself. Generally, the user is provided with some assistance in form of a document organization subsystem. For example, one of the most popular approaches is to order the retrieved document by their likelihood of being relevant to the query – place them in the ranked list [14]. Other alternatives include clustering and different document visualization techniques [15, 16].

In our previous work [21] we have considered a browsing interface for an information retrieval system that integrates the traditional ranked list with clustering visualization of documents. The visualization presents the documents as spheres floating in 2- or 3-dimensional space and positions them in proportion to the inter-document similarity. If two documents are very similar to each other, the corresponding spheres will be closely located and the spheres that are positioned far apart indicate a very different document content. Thus the visualization provides additional and very important information about the content of the retrieved set: while the ranked list shows how similar the documents are to the original query, the clustering visualization highlights how the documents relate to each other.

We studied the combination of the ranked list and clustering visualization by running intensive off-line simulations and a user study [20, 21]. We showed that the inter-document similarities can be accurately visualized for retrieval purposes. We observed that users navigating the combination can do significantly better than by following the ranked list alone. However, it was also obvious that the system having access to the exact values of the inter-document similarity can provide a helpful assistance to the users.

In this paper we present an extension to that study. We design a document selection procedure that can guide the user through the retrieved set helping her to locate the relevant information as quickly as possible. We assume that the user is ready to provide the system with her relevance judgments about the documents as she is examining them. The document selection procedure can be implemented as a "wizard" that comes up right after the documents are retrieved [22]. The wizard estimates the relevance values for each unexamined document and highlights the document's screen representations – both the sphere in the visualization part and the title in the ranked list – with a color shade of intensity proportional to the relevance estimation. The user can see where the most likely to be relevant documents are located in the ranked list and in the visualization. As the user examines the documents and marks them as relevant or not, the wizard reevaluates its estimations and changes the highlighting accordingly. This interaction is very similar to the traditional relevance feedback approach. However, in contrast to the relevance feedback we concentrate on analysis of documents that are already present in the retrieved set. We do not attempt to improve the original query, repeat the search process, and bring in new documents. In addition, our analysis focuses only on the inter-document similarity information obtained after the original retrieval session and ignores all term-level statistics. We assume that any estimations the wizard assigns based on the similarity information will be easily explainable to the user by our proximity-based visualization system. The wizard keeps both the ranked list and the visualization unchanged during the whole session adjusting only the color highlighting. We believe, that reordering the ranked list or making other structural changes to the presentation will have a disorienting effect on the users. The wizard points the user to the most likely relevant documents without forcing its choice onto her.

In this paper we design such a wizard by modeling the interaction process between the user and the document set – we consider an agent that is examining the document set, selecting unknown documents, and adjusting its behavior based on whether the selected document was relevant or not. We show that the relevance feedback methodology can be effectively applied in this setting. We formulate the feedback problem in terms of reinforcement learning and show how the agent effectiveness can be significantly improved after a modest amount of training. We evaluate the agent performance by assuming that the user always follows its suggestions and run the experiments on the standard TREC collections.

In the following sections we summarize the past research in relevance feedback and neural networks in information retrieval. We define interactive browsing as a reinforcement learning problem and consider three different agent representations. We describe our experiments and discuss results. We conclude with suggestions for future work.

2. RELATED WORK

Relevance feedback has been a major research focus of information retrieval for well over 30 years and has been shown to dramatically improve retrieval performance [29]. The general model is to examine a portion of the retrieved documents assigning relevance values to each of them. The documents' content are analyzed to "shift" the query closer towards the examined relevant documents and away from the examined non-relevant documents. This process is usually done in batches – the documents in the collection are broken into training and testing sub-collections, the queries are modified with the information from the training collection and evaluated on the test collection.

Aalbersberg [1] explored the notion of incremental relevance feedback, where the documents are considered one at a time and the query is modified after each document. Allan [2] conducted an intensive study of the incremental approach showing results as good as if the feedback occurred in one pass.

The most common model for implementing relevance feedback was outlined by Rocchio [27]. The terms are ranked based on the weighted sum of the terms weights in the old query, the known relevant, and known non-relevant documents. The new query is constructed with several top ranked terms:

$$w_i(\mathcal{Q}') = \alpha \cdot w_i(\mathcal{Q}) + \beta \cdot \overline{w_i(\mathcal{R})} + \gamma \cdot \overline{w_i(\mathcal{N})}$$

where $\overline{w_i(\mathcal{R})}$ and $\overline{w_i(\mathcal{N})}$ are the average weights of the ith term in the relevant and non-relevant documents, $w_i(\mathcal{Q})$ and $w_i(\mathcal{Q}')$ are the weights of the same term in the old and new queries. Parameters α, β, and γ – called *Rocchio coefficients* – control the relative impact of each component. Generally, these parameters are selected empirically and the best 10-20 terms are added to the query [4]. Buckley and Salton [9] suggested an approach called Dynamic Feedback Optimization (DFO) where the coefficients are learned by greedy exploration of the parameter space. They also demonstrated that increasing the number of expansion terms can improve performance.

Biron and Kraft [8] give a detailed overview of alternative methods for relevance feedback including the connectionist approaches. Neural networks found successful application in both retrieval and relevance feedback [6, 34, 19]. The network is constructed connecting the documents, terms, and query. The connection weights are initialized using the statistical information about the term usage and adjusted as the relevance judgments become available. In our study we consider only the similarity information between the documents and ignore the data about the individual terms.

Text classification and categorization is where the neural networks are trained to assign class labels to the documents based on the terms they contain [25, 30]. Information filtering also benefits from the neural network classification abilities. Here they are used to select relevant material from document streaming through the system. Schütze et al. [31] compared neural network-based classifiers with other classification techniques. Lewis et al. [23] and later Callan [10] studied different learning methods for the neural networks. Hull et al. [17] examined performance of combined methods where the decisions of several different classifiers including neural network were considered together.

The growth of the world wide web resulted in the development of autonomous agents that use the hyperlink environment to perform distributed search tasks on behalf of the user. Quite often a neural network forms a core part of such an agent. Menczer and Belew [24] studied the features of the web structure that can be most useful for the agents. The Wisconsin Adaptive Web Assistant [32] compiles its search instructions into a neural network allowing for adjusting of the search strategies as the training information become available. Joachims et al. [18] developed a system that performs look-ahead searches and provide the suggestions to the user on the base of reinforcement learning. Balabanovic [5] studied the agents that make use of preference information collected from multiple users. Choi and Yoo [11] considered how the multiple agents accepting user's feedback, can interact with each other and learn from their common experience. These systems make extensive use of the hyperlink structure and term level information. They slowly adapt to the user and her requests while performing the search task. We are looking at navigating unstructured text documents, training the agent off-line, and keep it unchanged during the actual search session.

3. SEARCH STRATEGY

The problem of navigating the retrieved document set can be naturally expressed as a reinforcement learning problem. Indeed, we have an agent operating in an environment – our document set. The agent interacts with the document set at discrete time steps. At each time step t the environment state is defined by the inter-document similarities, what documents were examined, and what relevance judgments were assigned. The agent receives some representation of the environment state ($\mathcal{D}_t$) and has to select an action – choose the next document d to examine. At the next time step the agent receives a numerical reward from the environment – whether the examined document is relevant or not. The agent has a specific goal – finding all relevant documents as quickly as possible.

The agent implementation is defined at each time step by a mapping between a state representation combined with an action and a numeric value: $F(\mathcal{D}_t, d)$. The agent computes the mapping for each unexamined document and selects d with the highest value of $F(\mathcal{D}_t, d)$. The process continues until all documents are examined. We call this interaction process the *search strategy* and call the agent implementation $F(\mathcal{D}_t, d)$ the *search strategy function.* Reinforcement learning methods specify how the search strategy function can be changed as a result of the interaction experience with the goal of maximizing the total reward.

In this paper we focus on a particular reinforcement learning algorithm called Temporal Difference (TD) learning [33]. A detailed description of the algorithm is the beyond the scope of this paper and can be found elsewhere [33, p.212]. Here we provide the update rule that modifies the search strategy function at each learning step and controls the outcome of the learning process:

$$\begin{aligned}\Delta\vec{\theta}_t &= \eta \cdot (r_{t+1} + \rho \cdot F_{\vec{\theta}}(\mathcal{D}_{t+1}, d_{t+1}) - F_{\vec{\theta}}(\mathcal{D}_t, d_t)) \\ &\quad \cdot \nabla_{\vec{\theta}} F_{\vec{\theta}}(\mathcal{D}_t, d_t)\end{aligned}$$

where t is the time step when we have to select a document, $\mathcal{D}_t$ is the environment state at the time step t, r_{t+1} is the reward obtained after the document d_t is examined, $\vec{\theta}$ is the parameter vector that define the search strategy function $F_{\vec{\theta}}(\mathcal{D}_t, d_t)$, η is the learning rate that controls the speed and stability of the learning process, and ρ is the discount factor that controls how much the reward at time $t+1$ is discounted by comparing to reward at time t.

There are three important questions that we have to yet to consider: the goal of the search strategy, the reward function, and the document set state representation:

- How do we measure if a search strategy was quick enough in discovering the relevant documents? The outcome of a search strategy is a new document ranking as opposed to the original ranked list. Two rankings can be compared using traditional information retrieval measure. In this study we use the average precision. Thus our problem is to define a search strategy that will maximize the expected average precision.

- The search strategy strongly depends upon the reward function r. In this study our reward function $r_{t+1} = 1$ if the examined document d_t was relevant and 0 otherwise – we reward the search strategy for discovering the relevant documents. However, the total reward will not depend upon when the relevant documents are examined – i.e., the total reward will be the same if all relevant documents examined at the beginning of the search or at the end. The discount factor ρ is a part of the reward function that decreases the reward if it was obtained at a later time step. If $\rho < 1$, the search strategy will prefer the relevant documents to occur towards the beginning of the search. At the same time it can allow for some exploration of the document set if the information collected during the exploration will help to find the relevant documents in one quick sweep. If $\rho = 0$, the search strategy will interested in only immediate rewards and ignore any chance for exploration.

- In the next three sections we consider the question of selecting good features to represent the document set state and define three different forms for $F_{\vec{\theta}}(\mathcal{D}, d)$. Our analysis is based on the ideas from the Rocchio feedback approach, where the document ranking is adjusted based on the information from the old query and examined documents.

3.1 Simple Rocchio

The first design follows directly from the Rocchio approach. We define the search strategy function $F_1(\mathcal{D}, d)$ as a single perceptron unit that has four inputs: bias or constant input, document similarity to the query, average similarity between the document and all examined relevant documents, and average similarity between the document and all examined non-relevant documents:

$$\begin{aligned}F_1(\mathcal{D}_t, d) &= \theta_0 + \theta_1 \cdot \mathit{querysim}(d) \\ &+ \theta_2 \cdot \frac{1}{|\mathcal{R}_t|} \sum_{\forall x \in \mathcal{R}_t} \mathit{sim}(x, d) \\ &+ \theta_3 \cdot \frac{1}{|\mathcal{N}_t|} \sum_{\forall x \in \mathcal{N}_t} \mathit{sim}(x, d)\end{aligned}$$

where $\mathit{querysim}(d)$ is the similarity between the document and the original query, $\mathit{sim}(x, d)$ is the similarity between two documents, $\mathcal{R}_t$ and $\mathcal{N}_t$ are the set of all examined relevant and non-relevant documents at time step t.

3.2 Application Coefficients

Our second design results from an heuristic observation that different parts in the Rocchio-based representation should have different weights depending on how many documents were examined. For example, at the beginning of the search, while few relevant documents are known, one should rely more on the similarity to the original query than further into the process when the document relevance information is plentiful. We accommodate this intuition by dividing the search process into three distinct phases and training a separate search strategy function for each phase. Specifically, we define our second search strategy function $F_2(\mathcal{D}, d)$ as a linear combination of three instances of the search strategy function from the previous section ($F_1(\mathcal{D}_t, d)$), where the coefficients – they are called *application coefficients* in machine learning – are smooth functions of the number of the examined documents [7]:

$$F_2(\mathcal{D}_t, d) = \sum_i A_i(|\mathcal{R}_t| + |\mathcal{N}_t|) \cdot F_{1i}(\mathcal{D}_t, d)$$

where $A_i(\cdot)$ is defined as following:

$$A_i(x) = \exp\left(-\frac{(x-\mu_i)^2}{\sigma^2}\right)$$

where μ_i defines the center of the influence area for the ith subnetwork $F_{1i}(\mathcal{D}_t, d)$ and σ defines its width.

3.3 Tile Coding

Both search strategy function representations discussed in the previous sections are quite limited in the shape of the functions they can approximate: $F_1(\mathcal{D}, d)$ describes a linear combination of features or a hyperplane, while $F_2(\mathcal{D}, d)$ describes three connected hyperplanes. For our third representation we used a technique called *tile coding* that can approximate more complex functions [33, p. 204].

In tile coding the feature space is partitioned with a regular grid (*tiling*) and a single number is assigned to each cell (*tile*) in the partition. Generally, several tilings are placed in the feature space and their origins are shifted by some amount relative to each other. The set of tilings defines the final function $F_3(\mathcal{D}, d)$: given a point in the feature space, a tile containing that point is selected from each grid and the average of the corresponding numbers is returned. Detailed overview of the tile coding and a good public-domain implementation of this technique can be found elsewhere [12].

The flexibility of tile coding comes at a cost of losing some accuracy while partitioning the feature space. The accuracy of the representation can be improved by increasing the number of tilings and decreasing the size of individual tiles. The generalization ability of the approximation is proportional to the tile size as well. Requiring the large number of tilings makes the tile coding much more computationally expensive than the simple linear approximator.

The search strategy function $F_3(\mathcal{D}, d)$ is defined on the feature space of five dimensions. Four of them are the same feature that used in $F_2(\mathcal{D}, d)$: document-query similarity, average similarities to relevant and non-relevant documents, and the number of examined documents. The fifth feature is the number of examined documents squared.

4. EXPERIMENTS

We generated the retrieved document sets for our experiments by running the Inquery retrieval engine [4] on two standard TREC collections. The engine ranks the retrieved documents based on their likelihood of being relevant. In practice, Inquery assigns what is called a *belief score* to each document in the collection. We used that score as the query-document similarity value.

4.1 Document Representation

The Inquery system is based on a probabilistic model of retrieval and does not incorporate the notion of similarity between documents. Therefore, to compute the inter-document similarities we used a vector-space approach where each document is represented by a vector term weights V. The weight of the ith term in the vocabulary, v_i is computed using the Inquery weighting formula, which uses Okapi's *tf* score [26] and Inquery's normalized *idf* score:

$$v_i = \frac{tf}{tf + 0.5 + 1.5\frac{doclen}{avgdoclen}} \cdot \frac{\log(\frac{colsize+0.5}{docf})}{\log(colsize+1)}$$

where *tf* is the number of times the term occurs in the document, *docf* is the number of documents the term occurs in, *doclen* is the number of terms in the document, *avgdoclen* is the average number of terms per document in the collection, and *colsize* is the number of documents in the collection. The similarity between a pair of documents is measured by the cosine of the angle between the corresponding vectors [28].

4.2 Experimental Setup

For our experiments we used TREC ad-hoc queries with their corresponding collections and the relevance judgments supplied by NIST accessors [13]. Specifically, TREC topics 251-300 and 301-350 were converted into queries and run against the documents in TREC volumes 2 and 4 (2.1GB) and TREC volumes 4 and 5 (2.2GB) accordingly. For each TREC topic we considered four types of queries: (1) a query constructed by extensive analysis and expansion [3]; (2) the description field of the topic; (3) the title of the topic; and (4) a query constructed from the title by expanding it using Local Context Analysis (LCA) [35]. A query of the last type has size and complexity between the corresponding queries of the first and second types.

Our assumption is that during a typical retrieval session a user does not generally look beyond the first screen showing the retrieved material – that is approximately equivalent to ten retrieved documents. Thus, we are interested in analyzing just the top portion of the ranked list. For each query we selected the 50 highest ranked documents.

4.3 Training and Evaluation Procedure

Following the interaction model outlined in our previous work we extended each document set with the relevance judgments for the highest ranked relevant document and all non-relevant documents that precede it in the ranked list [21] – we model a situation where a user located the first relevant document by following the ranked list. Thus the experimental task is *given the highest ranked relevant document as the starting point, find the rest of the relevant information.* We compare performances of the search strategies by computing the average precision on the unexamined portion of the document set.

We divided the eight data sets – one for each query type on each collection – into three subsets: training, testing, and evaluation. The document sets retrieved from different collections do not overlap. The document sets retrieved from the same collection exhibit some amount of overlapping. For example, each of four queries of different types created for the same TREC topic retrieve fifty documents creating a set of two hundred documents total. Our analysis shows that there are on average a hundred unique documents in that set. We trained each search strategy function on one data set, making it four functions trained on the data sets from one collection. We selected one function out of these four that performed best on the data from that collection

Table 1: Average precision and percent improvement over the baseline (in parentheses) for the top fifty retrieved documents. The search strategies are given a starting point for exploration: the relevance judgments for the top ranked relevant document and all non-relevant documents that precede it in the ranked list. Precision is computed for the unexamined portion of the retrieved set. Asterisks indicate statistical significance by two-tailed t-test with $p < 0.05$.

Data Set		Worst	Best	RF(+10t)	RF(+100t)	$F_1(\mathcal{D}, d)$	$F_2(\mathcal{D}, d)$	$F_3(\mathcal{D}, d)$
TREC-5	Full	14.69	82.00	39.86	44.17	46.55 (5.38* %)	50.65 (14.67*%)	51.00 (15.44*%)
	Desc	11.95	88.00	45.12	50.60	52.96 (4.67 %)	53.00 (4.74 %)	53.42 (5.57* %)
	Title	10.57	74.00	38.08	41.39	41.96 (1.38 %)	43.41 (4.87* %)	43.99 (6.27* %)
	Title+Desc	11.94	64.00	34.79	38.26	40.73 (6.45* %)	43.76 (14.37*%)	44.02 (15.05*%)
TREC-6	Full	19.73	90.00	53.64	54.70	57.85 (5.76* %)	61.84 (13.06*%)	62.38 (14.05*%)
	Desc	16.10	94.00	56.04	60.39	60.52 (0.22 %)	62.85 (4.07 %)	63.29 (4.80* %)
	Title	15.21	84.00	52.00	54.74	57.26 (4.61 %)	58.36 (6.61* %)	58.95 (7.70* %)
	Title+Desc	17.74	88.00	48.68	50.07	55.38 (10.59*%)	56.66 (13.16*%)	57.03 (13.90*%)
total average		14.74	83.00	46.03	49.29	51.65 (4.79* %)	53.82 (9.18* %)	54.26 (10.08*%)

Table 2: Average precision and percent improvement over the baseline (in parentheses) for the top hundred retrieved documents. The search strategies are given a starting point for exploration: the relevance judgments for the top ranked relevant document and all non-relevant documents that precede it in the ranked list. Precision is computed for the unexamined portion of the retrieved set. Asterisks indicate statistical significance by two-tailed t-test with $p < 0.05$.

Data Set		worst	best	RF(+100t)	$F_1(\mathcal{D}, d)$	$F_2(\mathcal{D}, d)$	$F_3(\mathcal{D}, d)$
TREC-5	Full	10.99	90.00	43.50	45.81 (5.32 %)	49.86 (14.63*%)	50.59 (16.31*%)
	Desc	8.49	88.00	45.36	48.37 (6.64* %)	48.52 (6.97* %)	48.92 (7.84* %)
	Title	7.96	84.00	39.49	42.03 (6.45* %)	43.14 (9.25* %)	44.00 (11.42*%)
	Title+Desc	9.16	70.00	34.25	37.82 (10.42*%)	39.94 (16.59*%)	40.80 (19.12*%)
TREC-6	Full	14.75	94.00	51.48	55.61 (8.04* %)	60.01 (16.58*%)	60.29 (17.12*%)
	Desc	10.98	96.00	57.30	61.69 (7.66* %)	64.41 (12.41*%)	64.74 (12.98*%)
	Title	10.78	92.00	52.43	55.71 (6.26* %)	55.20 (5.28 %)	56.48 (7.72* %)
	Title+Desc	12.43	92.00	51.07	53.86 (5.45 %)	55.79 (9.24* %)	56.22 (10.08*%)
total average		10.69	88.25	46.86	50.11 (6.94* %)	52.11 (11.20*%)	52.75 (12.58*%)

overall. We then evaluated that function on the four data sets retrieved from another collection.

We trained the search strategies by running them on the training set multiple times. Each search strategy function began with all parameters ($\vec{\theta}$) initialized to zero. We have experimented with different values for the TD-algorithm parameters. The learning rate $\eta = 0.1$ and discount factor $\rho = 0.4$ worked well in our experiments. The parameters for application coefficients ($A(\cdot)$) were defined as $\mu = 1, 25, 50$ and $\sigma = 6$. The tile coding representation used 256 tilings, each tile side was equal to one third of the corresponding feature range. The learning process was terminated at the point when the average precision failed to improve for several iterations.

4.4 Baseline

We have used the Inquery relevance feedback subsystem as our baseline: starting with the ranked list we follow it until the first relevant document is found. At that point all examined documents are analyzed and a new query is created by expanding the old query with several top ranked terms from the examined documents. The rest of the unexamined documents are re-ordered using the modified query and the process continues until all documents are examined.

5. RESULTS

Tables 1, 2, and 3 show the average precision numbers obtained for our baseline and three different search strategies and the percent improvement over the baseline. The first two columns in each table show the average precision numbers for two hypothetical search strategies: one that always examines all non-relevant documents before any relevant ones ("worst") and the other that considers all relevant documents before all non-relevant("best"). These numbers provide a scale for the performance results.

In the earlier study we considered the same baseline relevance feedback algorithm each time expanding the query with the top 10 highest ranked terms [21]. In our latest experiments we observed that adding more terms to the query increases performance – Table 1 shows a significant increase in average precision while expanding the query with top 100 terms instead of top 10. Increasing the number of terms beyond that amount (i.e., adding 500 terms) degrades the performance. Thus we use the expansion algorithm with 100 terms as our baseline.

Table 1 shows a significant improvement in the average precision for all search strategies over the baseline. The performance increases as more information is added to the document set representation. The first search strategy learns a better set of Rocchio coefficients and exhibits a 5% improvement over the baseline. The second strategy allows these coefficient to change as the browsing process progresses and achieves a 9% improvement. The third search strategy that uses tiling to represent its function demonstrates a 10% improvement.

Table 4: Rocchio coefficients from three different sources.

System	Old query (θ_1)	Relevant (θ_2)	Non-rel. (θ_3)
Inquery [3]	0.5	4	-1
DFO [9]	0.25	8	-1
$F_1(\mathcal{D}, d)$	0.5	2	-1

We observed a similar increase in average precision while considering the top 100 documents instead of the top 50 (Table 2).

The search strategies were beginning their exploration from a known starting point: the top ranked relevant document and all non-relevant documents positioned above it in the ranked list were marked as examined and the corresponding relevance judgments were provided to the search strategies. We have considered the importance of that starting point: the search strategies were to start without any relevance information and to explore the whole retrieved set. Table 3 shows the average precision numbers computed in those settings. Despite that the search strategies *were not retrained* to take into account the lack of the starting point, we observed a significant improvement in average precision over the baseline, albeit a smaller one.

5.1 Interpretation

The first two search strategy functions $F_1(\mathcal{D}, d)$ and $F_2(\mathcal{D}, d)$ are easily interpreted by considering the neural network coefficients learned in the training process. The size (256 tilings and more than 20000 tiles) and complexity (5 dimensions) of the tile coding representation make the interpretation of the final $F_3(\mathcal{D}, d)$ function very difficult and we do not attempt it in this paper.

The first representation – the straightforward adaptation of the Rocchio approach – showed values that are significantly different from those discussed in the past studies (Table 4). Inquery relevance feedback subsystem uses the relevant to non-relevant ratio of 4 : 1 and it has been shown to work well in TREC experiments [3]. We implemented a neural network that used the Inquery values and the search strategy exhibited performance very similar to what we have observed from our baseline. Buckley and Salton [9] recommend the ratio of 8 : 1. We implemented another neural network that used this ratio and observed a slight but not statistically significant drop in average precision while comparing it to the baseline. In our experiments the network learned the same 2 : 1 ratio for the Rocchio coefficients on each individual training set.

The second search strategy function representation was allowed to change the Rocchio coefficients depending on how many documents were examined. Figure 1 shows the learned coefficients vary with each examined document. We observed that the magnitude of the coefficient for the average similarity to non-relevant documents ($|\theta_3|$) steadily decreases as more documents are examined. The magnitude of the coefficient for the average similarity to relevant documents ($|\theta_2|$) displays an increase for the middle part of the process, while the importance of the document-query similarity ($|\theta_1|$) shows exactly the opposite behavior.

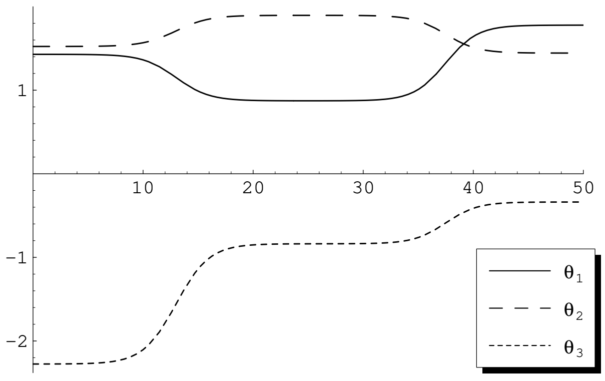

Figure 1: The Rocchio coefficients vary significantly with the number of examined documents.

6. CONCLUSIONS

Relevance feedback has been shown to increase performance of an information retrieval system on multiple occasions. We demonstrated that the technique also works in highly interactive settings even when no additional documents are retrieved. The relevance feedback methods can be effectively applied to support a user browsing the top portion of the ranked list and helping her to find all relevant documents as quickly as possible.

We modeled the feedback process with an agent that is interacting with the retrieved set. We formalized the agent description and formulated the problem in terms of reinforcement learning. We showed how the agent performance can be improved after a modest amount of training.

We have compared three different representations for search strategy function and showed how taking into account the number of examined documents improves our results. We demonstrated that the technique is very successful when only the inter-document similarity data is available and no term information is provided. Thus it easy to design a very efficient implementation of the feedback algorithm making it a good candidate for on-line search environments.

7. DISCUSSION AND FUTURE WORK

In our earlier work [21] we have conducted a similar study of interactive browsing. That analysis was mostly done in the visualization space considering the Euclidean distances between document representations in 2 and 3 dimensions. We only used the information about relevant documents in our experiments. The highest numbers reported earlier did not much exceed the interactive relevance feedback baseline with 10 terms expansion (Table 1, column 3). In this paper we focused on the similarity relationships among high-dimensional document vector representations, took into account both relevant and non-relevant judgments, and further improved the system by training. The final system (with tile coding representation) shows 13% increase in average precision over the best performance we have reported in the past [21].

The learning phase for the agent stops when it is finished

Table 3: Average precision and percent improvement over the baseline (in parentheses) for the top fifty retrieved documents. The search strategies start without any relevance information: precision is computed for the whole data set. Asterisks indicate statistical significance by two-tailed t-test with $p < 0.05$.

Data Set		worst	best	RF(+100t)	$F_1(\mathcal{D}, d)$	$F_2(\mathcal{D}, d)$	$F_3(\mathcal{D}, d)$
TREC-5	Full	15.64	94.00	42.58	46.06 (8.18*%)	47.58 (11.75*%)	48.19 (13.17*%)
	Desc	12.56	90.00	44.31	44.65 (0.77 %)	45.77 (3.31 %)	47.27 (6.68* %)
	Title	11.24	84.00	39.15	42.01 (7.31*%)	42.88 (9.53* %)	43.37 (10.77*%)
	Title+Desc	12.34	68.00	34.82	36.21 (4.00 %)	40.01 (14.90*%)	40.32 (15.80*%)
TREC-6	Full	20.92	94.00	54.66	57.15 (4.55 %)	62.44 (14.24*%)	60.46 (10.61*%)
	Desc	17.15	96.00	58.59	58.95 (0.62 %)	60.09 (2.56 %)	60.45 (3.19 %)
	Title	16.35	92.00	57.27	57.95 (1.19 %)	59.19 (3.34 %)	60.24 (5.18* %)
	Title+Desc	18.55	92.00	55.18	58.37 (5.78*%)	60.09 (8.89* %)	60.16 (9.02* %)
total average		15.59	88.75	48.32	50.17 (3.83*%)	52.26 (8.15* %)	52.56 (8.77* %)

with the training data set. We have experimented with an idea of allowing the agent to learn while running it on the evaluation set. Specifically, at each time step during the evaluation phase the agent was allowed to learn for several iterations using only the examined subset of the data. After each query the agent parameters were restored to their initial (learned on the training set) values. We observed that this learning did not have any effect on average: we saw an increase in average precision for some queries and a degradation for others.

There are several key points that made the reinforcement learning and TD-learning in particular a good choice for our problem. Firstly, the search strategy is learned from experience by interacting with the training examples. That interaction has an intuitive connection to how a user would interact with the system.

Secondly, the exhaustive knowledge about the problem space is not required – the TD-learning method allows the search strategy to be learned from samples. Several theoretical results guarantee convergence of the learning process to an optimal search strategy – the strategy that maximizes the expected total reward. These results are based on the assumption that the problem satisfies the Markov property [33]. We did not study if and by how much the problem in our formulation violates the Markov property. However, our experimental results show that convergence does occur and we leave the question about the Markov property for future work.

Lastly, unlike supervised learning, the reinforcement learning does not require knowledge about the perfect search strategy. The learning process is directed by means of providing rewards and punishments for each choice made by the search strategy. Such a framework is very flexible – different reward schemes will result in the search strategies maximizing different utility functions. It also requires a careful consideration while selecting a particular reward function. An intuitive choice of using the relevance value as the reward number requires from the search strategy to maximize "total discounted relevance" (TDR):

$$TDR(\rho) = \sum_{\forall d_t \in \mathcal{R}} \rho^t$$

while we evaluate the performance using the average precision (P):

$$P = \frac{1}{|\mathcal{R}|} \sum_{\forall d_t \in \mathcal{R}} \frac{|\mathcal{R}_t|}{|\mathcal{R}_t| + |\mathcal{N}_t|}$$

where $\mathcal{R}$ is the subset of all relevant documents in the data set and d_t is the document examined at time step t. Notice that $|\mathcal{R}_t| + |\mathcal{N}_t| = t$.

Despite that these are two different measures, the resulting search strategies exhibit high values of average precision in our experiments. That can probably be explained by a high correlation between $TDR(\rho)$ and P for low values of ρ. An alternative would be to use P as the reward at the end of the search ($r_{|\mathcal{D}|+1}$) with all intermediate rewards set to zero. Then the search strategy should learn to maximize the average precision. However, we have observed that using such a reward function results in a very slow learning process and a very similar search strategy. Currently we are experimenting with other different forms for the reward function and evaluation measure.

Acknowledgments

I am deeply grateful to James Allan for his support and contributions to the project. I also thank Andrew Barto and Michael Kositsky for their advice and comments about the reinforcement learning part of this work.

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623, and SPAWARSYSCEN-SD grant number N66001-99-1-8912. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] I. J. Aalbersberg. Incremental relevance feedback. In *Proceedings of ACM SIGIR*, pages 11–22, 1992.

[2] J. Allan. Incremental relevance feedback for information filtering. In *Proceedings of ACM SIGIR*, pages 270–278, 1996.

[3] J. Allan, J. Callan, B. Croft, L. Ballesteros, J. Broglio, J. Xu, and H. Shu. Inquery at TREC-5. In *Fifth Text REtrieval Conference (TREC-5)*, pages 119–132, 1997.

[4] J. Allan, J. Callan, W. B. Croft, L. Ballesteros, D. Byrd, R. Swan, and J. Xu. Inquery does battle with TREC-6. In *Sixth Text REtrieval Conference (TREC-6)*, pages 169–206, 1998.

[5] M. Balabanovic. An adaptive web page recommendation service. In *Proceedings of the First International Conference on Autonomous Agents*, pages 378–385, 1997.

[6] R. K. Belew. Adaptive information retrieval: using a connectionist representation to retrieve and learn about documents. In *Proceedings of ACM SIGIR*, pages 11–20, 1989.

[7] H. Berliner. On the construction of evaluation functions for large domains. In *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, 1979.

[8] P. V. Biron and D. H. Kraft. New methods for relevance feedback: improving information retrieval performance. In *ACM symposium on Applied computing*, pages 482–487, 1995.

[9] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proceedings of ACM SIGIR*, pages 351–357, 1995.

[10] J. P. Callan. Learning while filtering documents. In *Proceedings of ACM SIGIR*, pages 224–231, 1998.

[11] Y. S. Choi and S. I. Yoo. Multi-agent learning approach to www information retrieval using neural network. In *Proceedings of International Conference on Intelligent User Interfaces*, pages 23–30, 1999.

[12] CMAC. `http://www.ece.unh.edu/robots/cmac.htm`.

[13] D. Harman and E. Voorhees, editors. *The Fifth Text REtrieval Conference (TREC-5)*. NIST, 1997.

[14] D. Harman and E. Voorhees, editors. *The Sixth Text REtrieval Conference (TREC-6)*. NIST, 1998.

[15] M. A. Hearst and J. O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of ACM SIGIR*, pages 76–84, Aug. 1996.

[16] M. Hemmje, C. Kunkel, and A. Willet. LyberWorld - a visualization user interface supporting fulltext retrieval. In *Proceedings of ACM SIGIR*, pages 254–259, July 1994.

[17] D. A. Hull, J. O. Pedersen, and H. Schütze. Method combination for document filtering. In *Proceedings of ACM SIGIR*, pages 279–287, 1996.

[18] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI*, pages 770–778, 1997.

[19] K. L. Kwok. A network approach to probabilistic information retrieval. *ACM Transactions on Information Systems*, 13(3):324–353, 1995.

[20] A. Leuski and J. Allan. Evaluating a visual navigation system for a digital library. *International Journal on Digital Libraries*, 2000. Forthcoming.

[21] A. Leuski and J. Allan. Improving interactive retrieval by combining ranked lists and clustering. In *Proceedings of RIAO'2000*, pages 665–681, April 2000.

[22] A. Leuski and J. Allan. Lighthoise: showing the way to relevant information. In *Proceedings of InfoVis'2000*, October 2000.

[23] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of ACM SIGIR*, pages 298–306, 1996.

[24] F. Menczer and R. K. Belew. Adaptive information agents in distributed textual environments. In *Proceedings of the second international conference on Autonomous agents*, pages 157–164, 1998.

[25] H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perception learning, and a usability case study for text categorization. In *Proceedings of ACM SIGIR*, pages 67 – 73, 1997.

[26] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In D. Harman and E. Voorhees, editors, *Third Text REtrieval Conference (TREC-3)*. NIST, 1995.

[27] J. J. Rocchio, Jr. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall, Inc., 1971.

[28] G. Salton. *Automatic Text Processing*. Addison-Wesley, 1989.

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

[30] R. E. Schapire, Y. Singer, and A. Singhal. Boosting and rocchio applied to text filtering. In *Proceedings of ACM SIGIR*, pages 215–223, 1998.

[31] H. Schütze, D. A. Hull, and J. O. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of ACM SIGIR*, pages 229–237, 1995.

[32] J. Shavlik, S. Calcari, T. Eliassi-Rad, and J. Solock. An instructable, adaptive interface for discovering and monitoring information on the world-wide web. In *Proceedings of the 1999 international conference on Intelligent user interfaces*, pages 157–160, 1999.

[33] R. S. Sutton and A. G. Barto. *Reinforcement learning: an introduction*. The MIT Press, 1998.

[34] R. Wilkinson and P. Hingston. Using the cosine measure in a neural network for document retrieval. In *Proceedings of ACM SIGIR*, pages 202–210, 1991.

[35] J. Xu and W. B. Croft. Querying expansion using local and global document analysis. In *Proceedings of ACM SIGIR*, pages 4–11, 1996.