Learning Routing Queries in a Query Zone

Amit Singhal[†*], Mandar Mitra[‡], Chris Buckley[††]

[†]AT&T Labs Research; singhal@research.att.com
[‡]Department of Computer Science, Cornell University; mitra@cs.cornell.edu
[††]Sabir Research, Inc.; chrisb@sabir.com

Abstract

Word usage is domain dependent. A common word in one domain can be quite infrequent in another. In this study we exploit this property of word usage to improve document routing. We show that routing queries (profiles) learned only from the documents in a query domain are better than the routing profiles learned when query domains are not used. We approximate a query domain by a *query zone*. Experiments show that routing profiles learned from a query zone are 8–12% more effective than the profiles generated when no query zoning is used.

1 Background

Document routing is an important problem in the field of information retrieval. [12] When a user has marked several articles as relevant to his/her information need, a system should be able to automatically learn the user's "profile" and should be able to route (send) new, potentially interesting, articles to the user. This problem has also been called as *selective dissemination of information* or *information filtering*. [4]

Most current state of the art routing algorithms first learn a user profile from the training examples, *i.e.*, the articles marked relevant by the user and the non-relevant articles. This learned profile, also known as the routing or the feedback query since it is obtained using user's relevance feedback, is then matched against all the new articles that a system encounters (for example any new news stories, ...). [5, 28, 2, 9] If a new article matches the user profile adequately, then this article is assumed to be of potential interest to the user and is routed to the user.

The matching algorithm used by most systems to match a new article to a user profile is relatively straight-forward. Most current IR systems use their standard text matching algorithms to match new articles to a profile. [8, 20, 3, 16] For example, the Smart system uses the standard vector inner-product similarity computation to match user profiles (which are term vectors in Smart) to the new articles (which are also term vectors). [26] The effectiveness with which a system routes new articles to users is, then, largely dependent upon the quality of the profile generated by the system. Profile creation becomes the most important step in the routing process.

To create a user profile, most routing algorithms learn a set of features and feature relationships whose presence/absence indicates potential relevance/non-relevance for a user. Features are usually single words, phrases, word cooccurrence pairs, word proximity pairs, ... Based on the occurrence characteristics of these features in a new article, that article is either considered potentially useful and is routed to the user, or it is considered potentially irrelevant. Most current routing algorithms also assign weights to the features in a user profile. These weights indicate the relative importance of the features in predicting relevance of an article. [6, 5, 21]

To learn the features and their weights, most routing algorithms usually use the probability of occurrence (or some variation of it) of a feature in the articles marked relevant by a user and the non-relevant articles in the training corpus. [22, 14] The central idea of this scheme is that if a feature occurs with a high probability in the relevant articles but with a low probability in the non-relevant articles, then it is a good indicator of relevance and should be assigned a high weight in the profile. On the other hand, if a feature occurs with high probability in the non-relevant documents and does not occur often in the relevant documents, then it is a poor indicator of relevance.

A user initially conveys his/her information-need to a system in form of a query. Defining a query's domain as the set of articles that have some topical relationship to the query, in this study we show that using a selected set of non-relevant documents that belong to a query's domain to learn the feedback query yields better feedback queries than using the entire non-relevant corpus. Results show that simple strategies to select non-relevant articles for learning the feedback queries yield significantly better feedback queries. The rest of this study is organized as follows: Section 2 describes Rocchio's algorithm for learning routing queries, Section 3 introduces our hypothesis, Section 4 discusses the zoning strategies that we use, Section 5 discusses other related research, Section 6 describes the experiments, Section 7 has the results and discussion, Section 8 concludes the study.

*This study was done when the primary author was a doctoral candidate at Cornell, and was supported in part by the National Science Foundation under grant IRI-9300124.

SIGIR 97 Philadelphia PA, USA

Word	Average Weight in Relevant Documents	Average Weight in Non-Relevant Documents		Feedback Weight $\alpha = 0, \beta = \gamma$	
		Entire Corpus	Query Domain	Entire Corpus	Query Domain
weapon	4.607	0.069	2.557	4.538	2.051
missile	5.786	0.049	3.249	5.737	2.538
arms	2.646	0.096	2.017	2.549	0.629
defense	3.425	0.161	2.300	3.264	1.126
soviet	2.965	0.155	2.499	2.810	0.467

Table 1: Feedback weights of less important terms undesirably boosted due to use of entire non-relevant corpus.

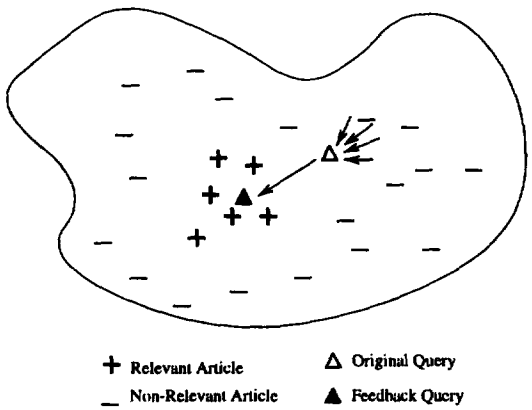

Figure 1: Migration of the query in the vector space. The arrows indicate the "moving away" of the query from the non-relevant documents and towards the relevant articles.

2 Rocchio's Algorithm

A feedback query creation algorithm developed by Joe Rocchio in the mid-1960's has, over the years, proven to be one of the most successful profile learning algorithms. [22, 23] Rocchio's algorithm was developed in the framework of the vector space model. [27] The algorithm is based upon the fact that if the relevance for a query is known, an *optimal*[1] query vector will maximize the average query–document similarity for the relevant articles, and will simultaneously minimize the average query–document similarity for the non-relevant documents. Rocchio shows that an optimal query vector is the difference vector of the centroid vectors for the relevant and the non-relevant articles.

$$\vec{Q}_{opt} = \frac{1}{R} \sum_{D \in Rel} \vec{D} - \frac{1}{N-R} \sum_{D \notin Rel} \vec{D}$$

Where R is the number of relevant articles, and N is the total number of articles in the collection. Also, all negative components of the resulting optimal query are assigned a zero weight.

In the vector space view, using Rocchio's query formulation amounts to moving the query vector away from the non-relevant vectors and closer to the relevant vectors in the vector space (see Figure 1). To maintain focus of the query, researchers have found that it is useful to include the original user-query in the feedback query creation process. Also, coefficients have been introduced in Rocchio's formulation which control the contribution of the original query, the relevant articles, and the non-relevant articles to the feedback query. These modifications yield the following query reformulation function: [25]

$$\vec{Q}_{new} = \alpha \times \vec{Q}_{orig} + \beta \times \frac{1}{R} \sum_{D \in Rel} \vec{D} - \gamma \times \frac{1}{N-R} \sum_{D \notin Rel} \vec{D} \quad (1)$$

The feedback query created by Rocchio's query reformulation process (using the documents marked relevant by a user) is now considered as the user profile. New incoming documents are matched against this profile and are routed to the user if they have a suitable match to the profile. The hope in this process is that the relevant articles in the new set of articles will be quite like the articles marked relevant by the user, and the non-relevant articles will be like the non-relevant articles in the training data. The user profile, which is designed to differentiate between relevance and non-relevance in the training set, will also do a good job of differentiating relevance from non-relevance in the new set of articles.

3 Hypothesis

Rocchio's optimal query is designed to maximize the average query–document similarity for all relevant articles, and to minimize the average query–document similarity for all known non-relevant articles. [23] The interesting part of this algorithm is its use of all known non-relevant articles, which means that an article which is completely unrelated to the user query also has a say in what the final query vector would be. This effect has its shortcomings.

Consider a query: *which disk drive should I buy for my Mac?* For this query, the word "computer" would be a good word to define the general domain of the query but would probably not be a very good word to separate articles on disk drives for Macs from other computer related articles. When we use the entire set of non-relevant articles in the training corpus to get the final weight for the word "computer", its average weight (occurrence density) in the non-relevant articles will be quite low (due to its absence from numerous non-relevant articles from domains other than *computers*). On the other hand, since most articles on disk drives for Macs will probably mention the word "computer", the average weight for this word in the relevant articles will be reasonably high. In the final Rocchio formulation, the word "computer" would appear as a very strong indicator of relevance; whereas in reality, as we mention above, it might not be a very good word to separate relevant articles from the non-relevant articles in the domain of *computers*. In this case, the use of all non-relevant articles has boosted the importance of a term that is possibly not very important.

This phenomenon is illustrated in Table 1. Table 1 shows the weight distribution for several words pertaining to the

[1] See [23] (page 315) for Rocchio's definition of an optimal query.

TREC-3 routing query (number 101): *Design of the "Star Wars" Anti-missile Defense System; Document will provide information on the proposed configuration, components, and technology of the U.S.'s "star wars" anti-missile defense system* ... For several words, Table 1 shows the average weight in the relevant articles; the average weight in the non-relevant articles when the entire non-relevant corpus is considered, and when only the non-relevant documents in the query domain are used[2]. Table 1 also shows the resulting weights of these words in the feedback query if the initial query is ignored ($\alpha = 0$ in Equation 1) and the relevant and the non-relevant articles are given equal importance ($\beta = \gamma$)[3]. All words shown in Table 1 are words that get higher final weights by using the entire non-relevant corpus than the weights we would have assigned them if we were using the non-relevant documents from the query domain only.

For example, in Table 1, the word "soviet" gets a high final weight of 2.81 when the entire corpus of non-relevant documents is considered. This shows that the occurrence frequency of the word "soviet" is much higher in the relevant documents than its occurrence frequency in the rest of the corpus (which is visible from a high average weight — 2.97 — in the relevant documents and a very low average weight — 0.15 — in the non-relevant documents); indicating that it is a good word to isolate the general domain of *anti-missile defense systems* from the rest of the corpus. But once we are in the domain, the occurrence characteristics of the word "soviet" is not very different in the relevant and the non-relevant articles. The average weight of this word in the non-relevant articles in the query domain is 2.50, which is much closer to its average weight in the relevant documents (2.97). This indicates that once we are in the general domain of *anti-missile defense systems*, the word "soviet" is not a very good discriminator of relevance for this query. Intuitively, this is correct as the query is about "star wars", a U.S. program.

A minor problem with using all non-relevant articles in profile creation is that the weights of moderately good terms can possibly be suppressed. Consider the domain of (say) *U.S. policy on Somalia* which has (hopefully) almost no connection to a query on *poaching of endangered species in Asia*. If we allow documents on *U.S. policy on Somalia* to affect the Rocchio formulation for this query on poaching, we are bound to confuse the algorithm which would assign misleading weights to some terms. An example would be a word like "killing" which might occur with reasonable chances in the *U.S. policy on Somalia* domain. As this entire domain is non-relevant, the average weight of the word "killing" in the non-relevant documents will be boosted, yielding low importance for this word in the final query. But in reality, this word might be very important in the domain of *poaching of endangered species in Asia*. In this scenario, by letting a completely unrelated domain affect the final query vector, we have undesirably suppressed the importance of a good term.

The aim of a routing algorithm is to generate a profile that discriminates between the potentially relevant and the potentially non-relevant articles in the set of all the unseen articles. A routing algorithm should learn how to make the harder distinction between the relevant articles and the non-relevant articles *that relate well to the user query*. The original Rocchio proposal, with its use of all the non-relevant articles to learn a feedback query, does a good job of differentiating the articles in the general domain of a query from the articles that are *not* in the query domain; but it fails to make the finer distinctions between the relevant and the non-relevant articles in the query domain.

[2]Here the query domain is simulated by considering the top 5,000 documents retrieved by the original query.

[3]Documents are weighted using the Ltu weighting scheme of the Smart system. [30]

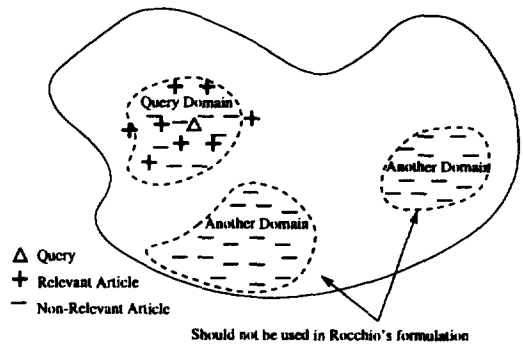

Figure 2: Hypothesis: Only use documents in query domain.

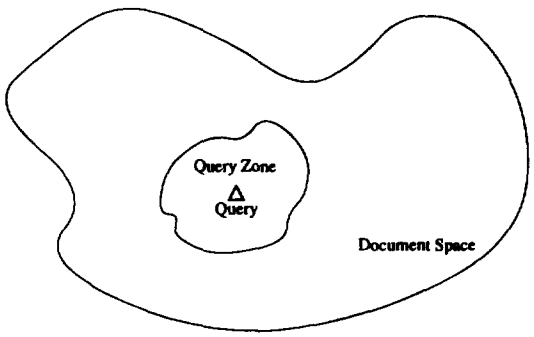

Figure 3: A Query Zone

We therefore propose that we should learn user profiles in the general domain of a query and not from the entire corpus. This means that we should only use those non-relevant documents in Rocchio's formulation that have some connection to the user query. All other non-relevant documents in the corpus should be ignored in learning a user profile. This process, as indicated above, would yield term weights that are much better representatives of the actual term importance for the query under consideration. In the vector space view, using only the non-relevant documents from the query domain involves rejecting non-relevant documents that are far away from the user query during the feedback query formulation. This hypothesis is graphically illustrated in Figure 2, which proposes that articles from domains that are completely unrelated to the query domain should not be used in the feedback process.

4 Query Zone

If query domains were well defined sets of articles, then we could have used only those articles in Rocchio's query reformulation. In absence of any formal definition of a query domain, we approximate a query domain by a *query zone*. In the vector space model, a query zone can be envisioned as a volume of the vector space, or a cloud in the vector space, around a query vector (see Figure 3). In practice, a query zone for a user query can be simulated by considering a set of articles that have some reasonable similarity to the query.

The aim then is to select a set of non-relevant articles

(in the training set) that relate well to the user query to be used in Rocchio's formulation. In this study we explore several possible definitions of a query zone, and compare the feedback query generated by the use of a query zone to the feedback query generated if the entire corpus was used in Rocchio's formulation. The definitions of the query zone that we explore are :

- No-QZ: **No query zoning**. This acts as the baseline for all comparisons. All non-relevant documents, *i.e.*, all documents judged non-relevant as well as all the unjudged documents[4], are used to learn a routing query.

  Parameters involved: α, β, γ (see Equation 1).

- QZ-1: **All documents in the top K documents as ranked by the original query and any missing relevant documents.** Rank the training data according to the original query, and assume that the top K (say 5,000) documents form the query zone. If some relevant document is ranked below the top K documents, include it in the query zone. This strategy was used in our TREC-5 participation. [7]

  Parameters involved: α, β, γ, and K.

- QZ-2: **All documents with similarity to the original query greater than some threshold (S).** Queries can have narrow or broad domains. A query from a very narrow domain should have a smaller query zone than a broad query. To capture this notion, we use similarity thresholds to simulate query domains, the assumption being that if a query has, in general, low similarity to the documents, then it is a narrow query, and vice-versa. Under this assumption, all documents that pass a certain similarity threshold (say S) are considered to be in the query zone. If some relevant document does not pass the similarity threshold, it is included in the query zone.

  Parameters involved: α, β, γ, and S.

- QZ-3: **Dynamic query zoning.** This strategy is motivated by Buckley and Salton's dynamic feedback optimization (DFO) technique. [5] DFO aims at improving feedback weights obtained from a feedback technique (like Rocchio) by changing individual term weights and studying the effect of the change, retrospectively, on the training set of documents. In essence, DFO selects the best feedback query vector from a set of feedback query vectors (it generates the set of vectors it explores using a simple iterative algorithm). In dynamic query zoning, we use multiple zoning schemes to get multiple feedback queries and measure the retrieval effectiveness of the resulting queries on the *training corpus*, selecting the best query (in effect the best zoning strategy). Dynamic query zoning is not as computationally expensive as the current DFO techniques. For the same user query, we generate multiple feedback queries by using different rank cut-offs in QZ-1. We then run these queries on the training corpus retrospectively and select the query that has the best average precision performance on the training corpus.

  Parameters involved: α, β, γ, and a list of rank cut-offs.

[4] This situation arises in TREC because pooling is used for relevance judgments. [11, 12] We assume that all the unjudged documents are non-relevant.

5 Related Work

In [15] Hull, and in [28] Schutze, Hull, and Pedersen want to reduce the dimensionality of the feature space dramatically for use with strong learning methods. They use singular valued decomposition (SVD) of the document space (actually the documents–to–features matrix) to obtain a small number of LSI factors to be used with their learning methods. Since SVD is extremely computationally intensive, they reduce the number of documents they use in their document space, and instead work in a "local region". In [15] Hull uses only the relevant documents for a query to learn the LSI factors. In [28] Schutze, Hull, and Pedersen use the top 2,000 documents retrieved by a query generated using Rocchio's feedback to learn the LSI factors needed in their classifiers. The main motivation in these studies is to reduce the size of the training data for it to be usable with computationally intensive SVD methods. In [28] they mention that local LSI also has the advantage of using the non-relevant articles that are most difficult to distinguish from the relevant documents (see [28] page 233), but have not explicitly measured the advantages from their local LSI technique vs. a global one.

Allan et al. use a technique which only considers the top R *non-relevant* articles to learn a routing query, where the query has R relevant articles in the training set. [3] This is motivated by the need to have a balance between the number of the positive and the negative examples in Rocchio's learning of feedback queries. [1] We experimented with a similar strategy which selects the top $k \times R$ *non-relevant* (where k is an integer, we used 1, 2, 4, 6, 8, 10, 12, 14, and 16) articles for a query that has R relevant articles in the training set. Space restrictions prohibit us from discussing the results in detail but this particular zoning strategy yields very similar results as QZ-1 and QZ-2 (presented in Section 7). Using the top $6 \times R$ non-relevant documents was the best from TREC-3, and using the top $14 \times R$ non-relevant documents was best for TREC-4.

Our techniques can also be thought of as sampling techniques that selectively use non-relevant documents for training. Sampling techniques are well studied in the machine learning community. Our techniques come closest to *uncertainty sampling* of Lewis and Gale [18], which is motivated by the *query by committee* technique of Seung, Opper, and Sompolinsky. [29] In uncertainty sampling, only the training examples whose class membership is uncertain (given the current classifier) are proposed to a user for membership judgment. Most Sampling techniques in machine learning aim at reducing the size of the training set [19] and are not motivated by improving the classification accuracy (except for a few like [10]). Our zoning techniques, on the other hand, are specifically aimed at improving the routing effectiveness and are not motivated by the size of the training corpus.

Recently Kwok and Grunfeld have used a sampling technique based on genetic algorithms that selects the best training subset of the relevant articles to be used in creation of a feedback query. [17] Since we have a small number of relevant articles (positive examples) for a typical routing query, we did not consider ignoring any of those in our training phase. Also Kwok and Grunfeld did not obtain very encouraging results by ignoring some of the relevant articles. [17] It might be possible to combine sampling techniques for the relevant articles (like Kwok's) and our techniques for sampling the non-relevant articles to obtain an even richer set of training data.

$\alpha = 8$				
	$\gamma = 32$	$\gamma = 64$	$\gamma = 128$	$\gamma = 256$
$\beta = 16$	0.3870	0.3872	0.3794	—
$\beta = 32$	0.3931	**0.3973**	0.3955	0.3932
$\beta = 64$	0.3860	0.3914	0.3967	0.3962
$\beta = 128$	—	0.3790	0.3863	0.3915

Table 2: Average precision for different α, β, γ: TREC-3, no query zoning.

$\alpha = 8$				
	$\gamma = 128$	$\gamma = 256$	$\gamma = 512$	$\gamma = 1024$
$\beta = 32$	0.3496	0.3339	0.3068	—
$\beta = 64$	0.3509	0.3551	0.3349	0.3090
$\beta = 128$	0.3366	0.3510	**0.3555**	0.3346
$\beta = 256$	—	0.3341	0.3495	0.3543

Table 3: Average precision for different α, β, γ: TREC-4, no query zoning.

6 Experiments

We test the effectiveness of the routing queries learned using zoning (QZ-1, QZ-2, QZ-3) by comparing them to the routing queries learned when query zoning is not used (No-QZ). We use two benchmark routing tasks for all our experiments — the TREC-3 routing task, queries: TREC topics 101–150, training corpus: disks 1 and 2, test corpus: disk 3; and the TREC-4 routing task, queries: fifty TREC topics, training corpus: disks 1, 2, and 3, test corpus: 804 Mbytes of text from Ziff, FR 1994, and the Internet. [11, 12] We select a reasonable set of Rocchio parameters (α, β, γ) for No-QZ. To avoid multi-dimensional tuning of parameters, we select a good set of Rocchio parameters for $K = 5,000$ for QZ-1 and use the same α, β, γ across *all* query zoning strategies. Document vectors are individually weighted using the Ltu weighting scheme of the Smart system. [30]

Table 2 shows the average precision results obtained by using various α, β, γ values for the TREC-3 routing task when no query zoning is used. Table 3 shows the same tuning for the TREC-4 routing task. Since we don't have the luxury of retrospectively tuning Rocchio parameters, we should select one set of parameters to be used across tasks. The set of parameters $\alpha = 8$, $\beta = 64$, $\gamma = 256$ is a reasonable compromise when no query zoning is used. Using $\alpha = 8$, $\beta = 64$, $\gamma = 256$, the results for TREC-3 and TREC-4 (average precision values 0.3962, and 0.3551, respectively) are very close to the results obtained if the best set of parameters was used for both the tasks (0.3973, and 0.3555, respectively). We use this set of parameters as the baseline run in all our comparisons.

Tables 4 and 5 show the results from using query zoning QZ-1 with $K = 5,000$ for the two tasks. We observe that Rocchio parameters $\alpha = 8$, $\beta = 64$, $\gamma = 64$ yield the best results for the TREC-3 task (see Table 4) when query zoning is used. For the TREC-4 task, the best set of parameters is $\alpha = 8$, $\beta = 32$, $\gamma = 32$ (see Table 5). Once again, since we cannot retrospectively learn these parameters, we select one set ($\alpha = 8$, $\beta = 64$, $\gamma = 64$) as a reasonable set and use it across query zones.

We use $\alpha = 8$, $\beta = 64$, $\gamma = 64$ in *all* our query zoning experiments. This reduces the number of parameters

$\alpha = 8$			
	$\gamma = 32$	$\gamma = 64$	$\gamma = 128$
$\beta = 32$	0.4243	0.3911	0.2967
$\beta = 64$	0.4159	**0.4302**	0.3849
$\beta = 128$	0.3981	0.4186	0.4266

Table 4: Average precision for different α, β, γ: TREC-3, QZ-1 ($K = 5,000$).

$\alpha = 8$			
	$\gamma = 16$	$\gamma = 32$	$\gamma = 64$
$\beta = 16$	0.3518	0.3138	0.2159
$\beta = 32$	0.3637	**0.3786**	0.3056
$\beta = 64$	0.3366	0.3510	0.3758

Table 5: Average precision for different α, β, γ: TREC-4, QZ-1 ($K = 5,000$).

involved in QZ-1 and QZ-2 to just one (K and S, respectively). QZ-3 involves only a list of rank cut-offs but then automatically picks the best cut-off on a per query basis. So QZ-3, in effect, has no parameter to train for. We use the following rank cut-offs for QZ-3: 1,000, 2,000, 4,000, 6,000, 8,000, and 10,000. We compare all the results from query zoning to No-QZ results with $\alpha = 8$, $\beta = 64$, $\gamma = 256$. Also, in all our experiments, we retain the highest weighted 100 single terms and 10 phrases in a feedback query, as weighted by the Rocchio formulation.

7 Results and Discussion

We use our three query zoning strategies on the two routing tasks.

QZ-1: The average precision figures for QZ-1 for various rank cut-offs K are shown in Table 6. We immediately observe that the profiles learned from query zone QZ-1 are generally better than the profiles learned when no query zoning is used. The improvements from using a query zone are more marked for the TREC-3 routing task (where we get an improvement of 8–12%). The improvements for the TREC-4 routing task are not as marked (where we get an improvement of 5–6%).

Table 6 shows that a "tight" query zone (small number of top documents) is more useful for the TREC-3 task, whereas a "loose" query zone (large number of documents) is more useful for the TREC-4 task. The main reason for this, we believe, is the similarities/differences in the characteristics of the training and the test datasets in the two tasks. Rocchio's formulation (or any other routing method, in its own way) "models" the relevant articles for a query (by averaging the relevant vector and coming-up with a canonical relevant vector), and it also "models" the non-relevant articles. The closer the test set is to the model built by a routing algorithm, the better is the routing effectiveness of the feedback query. If the test set is widely different from the training set, then routing algorithms are known to perform poorly[5].

The TREC-3 test dataset (disk 3) resembles the training dataset (disks 1 and 2) to a very large extent — majority of documents in the two sets have the same source for documents: WSJ, AP, and ZIFF. The test set does have

[5] This is one of the main reasons why the general performance in the TREC-4 and the TREC-5 routing tasks is much lower than the performance for the TREC-3 routing task. [12, 13]

	K (rank cut-off)					
	1,000	2,000	4,000	6,000	8,000	10,000
TREC-3 Average Precision	0.4368	0.4415	0.4326	0.4279	0.4235	0.4213
TREC-3 vs. No-QZ (0.3962)	+10.2%	+11.4%	+ 9.2%	+ 8.0%	+ 6.9%	+ 6.3%
TREC-4 Average Precision	0.3223	0.3535	0.3735	0.3777	0.3776	0.3773
TREC-4 vs. No-QZ (0.3551)	-9.2%	-0.5%	+ 5.2%	+ 6.4%	+ 6.3%	+ 6.3%

Table 6: Average precision for different rank cut-offs: QZ-1.

QZ-1 Size	1,000	2,000	4,000	6,000	8,000	10,000
Retrospective (Training Data)	0.3192	**0.3317**	0.3282	0.3244	0.3205	0.3171
Predictive (Test Data)	0.3223	0.3535	0.3735	**0.3777**	0.3776	0.3773

Table 7: Retrospective and predictive performance of feedback queries learned from QZ-1 of different sizes: TREC-4 task.

some new type of articles (U.S. Patents) but they are very few (only 6,711 out of 336,310). In such a scenario, one can afford to "aggressively" build a restricted model for the non-relevant documents by using the top few non-relevant documents only. Since the test set will be very close to this model, the performance of the feedback query will be good.

On the other hand, when the test set has a very different composition than the training set — as is the case with the TREC-4 routing task where the test set (Ziff, FR, and Internet material) has many articles (113,205 out of 329,780) from internet news-groups, IR digest, and virtual worlds, which have completely different characteristics than the articles in the learning set (disks 1, 2, and 3) — then we cannot use a restrictive set of non-relevant documents to model the non-relevant documents that will be encountered in the test set. If we do, we tend to overfit our feedback queries to the training data. This can be observed in Table 7 which shows the retrieval effectiveness of the feedback queries learned from query zones of different sizes (tight — 1,000–2,000 documents — to loose — 8,000–10,000 documents) retrospectively on the training dataset and predictively on the test dataset for TREC-4 routing queries. We observe in Table 7 that by using a loose query zone (top 6,000 documents) in place of a tight query zone (top 2,000 documents), even though the resulting queries perform worse on the training dataset, the performance improves noticeably on the test dataset. This indicates that even though building a restricted model of the non-relevant articles is good for the training data, since the test data has differing characteristics, a more general model of the non-relevant documents is actually better. Results in Table 6 also show that when we use a very strict query zone of top 1,000 documents only, we badly overfit our feedback queries to the training data and get routing queries which are, in effect, noticeably worse that using the complete set of non-relevant articles.

QZ-2: Some queries might cover a broad topic or even multiple topics, whereas other queries might be very specific. The notion of a fixed size query zone for every query does not capture this essence of user queries. To take this reality into account, we can define that a document belongs to a query zone only if it exhibits a certain degree of semantic relatedness to a query. In the vector space model, we simulate this by including in a query zone all documents that pass a certain similarity threshold with the query. A low similarity threshold will result in a loose (or large) query zone and vice versa. Suppose the similarity threshold is S, if a query has a narrow domain, not too many documents might have a similarity $\geq S$ with the query; whereas for a broad query, many documents might cross the similarity threshold of S.

	S (similarity threshold)			
	0.15	0.20	0.25	0.30
TREC-3 Average Precision	0.4202	0.4332	0.4408	0.4218
TREC-3 vs. No-QZ (0.3962)	+ 6.1%	+ 9.3%	+11.3%	+ 6.5%
TREC-4 Average Precision	0.3717	0.3731	0.3699	—
TREC-4 vs. No-QZ (0.3551)	+ 4.7%	+ 5.1%	+ 4.2%	—

Table 8: Average precision for different thresholds: QZ-2.

Query document similarity is dependent on the document length as well as the query length, and we do normalize the document vectors for length, [30] for a similarity threshold based scheme to be effective across queries, we will need to length-normalize the query vectors as well. Since our similarity formulation is linear, normalizing a query vector by the sum of the individual (unnormalized) term weights is

	No Query Zoning	QZ-3
TREC-3	0.3962	0.4440 +12.1%
TREC-4	0.3551	0.3868 + 8.9%

Table 9: Average precision for dynamic query zoning: QZ-3.

a reasonable solution[6]. Using sum-normalized queries, the results from using similarity threshold based query zoning are shown in Table 8. We observe in Table 8 that there is no noticeable difference in using a threshold based query zone (QZ-2) and using a fixed size query zone (QZ-1, see Table 6). Once we select a compromise parameter value ($K = 4,000$ for QZ-1 and $S = 0.20$ for QZ-2), both strategies yield above 9% improvement for TREC-3 queries and above 5% improvement for TREC-4 queries over using no query zoning.

These improvements are important, especially in light of the fact that there is not much extra effort involved in doing query zoning over not doing it. To use QZ-1, a routing algorithm can periodically modify a user profile by retrieving top 4,000–5,000 documents and creating a profile vector. Maintaining a profile is even easier for QZ-2. As and when an article passes the similarity threshold with the initial user query, depending upon its relevance, it can be added to a "sum vector" for the relevant or the non-relevant articles. A profile can then be created 'on the fly' by just dividing the current sum vectors by the number–relevant and the number–non-relevant figures and taking a vector difference.

QZ-3: If we somehow did know that a certain query zone size will be the best for a given query, then we can obtain better results than the results obtained from using a compromise parameter value across queries. For example, if we use a compromise query zone size of 4,000 for QZ-1, the average precision for the TREC-3 task is 0.4326 (Table 6); if we somehow knew that for this query set, a query zone of 2,000 documents will be better, we could have obtained slightly better results. Motivated by dynamic feedback optimization of Buckley and Salton, [5] we try to "learn" a good query zone size on a per query basis from the training data. To do this, we generate various profiles for a user query by using query zones of different sizes. For a given user query, we generate several feedback queries by using QZ-1 with top 1,000, 2,000, 4,000, ..., 10,000 documents retrieved. We then evaluate the retrieval effectiveness of each feedback query retrospectively on the *training data*. The best feedback query for that user query is then used in a predictive setting for actual document routing. It is thus possible that one user's profile is built from a query zone of size 1,000 whereas another user's profile is built from a query zone of size 10,000.

This query zoning scheme does not require any parameter, instead, it learns the parameter involved in QZ-1 automatically on a per query basis. The results from using such dynamic query zoning are listed in Table 9. Table 9 shows that by using dynamic query zoning, the results for the TREC-3 task (average precision 0.4440) are marginally better than the best parameter setting in QZ-1 (average precision 0.4415 for $K = 2,000$) as well as in QZ-2 (average precision 0.4408 for $S = 0.25$). Also, for the TREC-4 task, dynamic query zoning (average precision 0.3868) is marginally better than the best settings for both QZ-1 (0.3777 for $K = 6,000$) and QZ-2 (0.3731 for $S = 0.20$). The results obtained from this query zoning strategy are impressive. Important improvements over not using query zoning are obtained for both the routing tasks, and there is no parameter involved. These improvements are especially impressive given the fact that the comparison baseline for these results is quite high.

[6] In the course of our experiments, we also used the well-known cosine normalization scheme for the queries, [24] but sum-based normalization yields more consistent results across query sets.

8 Conclusions

Learning routing queries by using a restricted set of non-relevant documents in place of the entire set of non-relevant documents is effective. We suggest the use of dynamic query zoning. For the TREC-3 routing task, routing queries obtained by using query zoning perform 12% better than not using query zoning. This improvement is 9% for the TREC-4 task. These improvements are important, especially given the fact that using a query zone does not involve much extra work for a system. Our results show that a routing algorithm should learn to discriminate between the relevant articles and the non-relevant articles *that do have some relationship to the user query*. Learning to discriminate between the relevant articles and all non-relevant articles including those that are completely unrelated to a user query will not yield very effective routing queries.

References

[1] J. Allan. Personal Communication.

[2] J. Allan. Incremental relevance feedback for information filtering. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 270–278. Association for Computing Machinery, New York, Aug. 1996.

[3] J. Allan, L. Ballesteros, J. Callan, W. Croft, and Z. Lu. Recent experiments with INQUERY. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 49–64. NIST Special Publication 500-236, October 1996.

[4] N. Belkin and W. Croft. Information filtering and information retrieval: Two sides of the same coin? *Communications of the ACM*, 35(12):29–38, 1992.

[5] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357. Association for Computing Machinery, New York, July 1995.

[6] C. Buckley, G. Salton, and J. Allan. The effect of adding relevance information in a relevance feedback environment. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 292–300. Springer-Verlag, New York, July 1994.

[7] C. Buckley, A. Singhal, and M. Mitra. Using query zoning and correlation within SMART : TREC 5. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. NIST Special Publication, 1997. To appear.

[8] C. Buckley, A. Singhal, M. Mitra, and G. Salton. New retrieval approaches using SMART : TREC 4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 25–48. NIST Special Publication 500-236, October 1996.

[9] J. Callan. Information filtering with inference networks. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269. Association for Computing Machinery, New York, Aug. 1996.

[10] D. T. Davis and J.-N. Hwang. Attentional focus training by boundary region data selection. In *International Joint Conference on Neural Networks*, pages I–676 to I–681, Baltimore, MD, June 7–11 1992.

[11] D. K. Harman. Overview of the third Text REtrieval Conference (TREC-3). In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 1–19. NIST Special Publication 500-225, April 1995.

[12] D. K. Harman. Overview of the fourth Text REtrieval Conference (TREC-4). In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 1–24. NIST Special Publication 500-236, October 1996.

[13] D. K. Harman. Overview of the fifth Text REtrieval Conference (TREC-5). In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, 1997 (to appear).

[14] D. Harper. *Relevance Feedback in Document Retrieval Systems*. PhD thesis, University of Cambridge, England, 1980.

[15] D. Hull. Improving text retrieval for the routing problem using latent semantic indexing. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 282–291. Springer-Verlag, New York, July 1994.

[16] K. Kwok and L. Grunfeld. TREC-4 ad-hoc, routing retrieval and filtering experiments using PIRCS. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 145–152. NIST Special Publication 500-236, October 1996.

[17] K. Kwok and L. Grunfeld. TREC-5 english and chinese retrieval experiments using PIRCS. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, 1997 (to appear).

[18] D. Lewis and W. Gale. A sequential algorithm for training text classifiers. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12. Springer-Verlag, New York, July 1994.

[19] M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, 4(2):305–318, Mar. 1993.

[20] S. Robertson, S. Walker, M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at TREC–4. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, pages 73–96. NIST Special Publication 500-236, October 1996.

[21] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC–3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, pages 109–126. NIST Special Publication 500-225, April 1995.

[22] J. Rocchio. *Document Retrieval Systems–Optimization and Evaluation.* PhD thesis, Harvard Computational Laboratory, Cambridge, MA, 1966.

[23] J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System—Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[24] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

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

[26] G. Salton and M. McGill. *Introduction to Modern Information Retrieval.* McGraw Hill Book Co., New York, 1983.

[27] G. Salton, A. Wong, and C. Yang. A vector space model for information retrieval. *Communications of the ACM*, 18(11):613–620, November 1975.

[28] H. Schutze, D. Hull, and J. Pedersen. A comparison of classifiers and document representations for the routing problem. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 229–237. Association for Computing Machinery, New York, July 1995.

[29] H. Seung, M. Opper, and H. Sompolinsky. Query by committee. In *Proceedings of the Fifth Annual ACM Workshop on Computational Learning Theory*, pages 287–294. ACM Press, July 1992.

[30] A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29. Association for Computing Machinery, New York, Aug. 1996.