Learning SVM Ranking Function from User Feedback Using Document Metadata and Active Learning in the Biomedical Domain

Robert Arens

University of Iowa, Iowa City, IA, 52242
robert-arens@uiowa.edu

Abstract. Information overload is a well-known problem facing biomedical professionals. MEDLINE, the biomedical bibliographic database, adds hundreds of articles daily to the millions already in its collection. This overload is exacerbated by the lack of relevance-based ranking for search results, as well as disparate levels of search skill and domain experience of professionals using systems designed to search MEDLINE. We propose to address these problems through learning ranking functions from user relevance feedback. We hypothesize that learning from feedback will give performance similar to learning from the entire data set. We hypothesize that, by employing active learning techniques, we can achieve this performance using feedback on a fraction of the total number of results. We further hypothesize that learning from metadata, specifically the Medical Subject Heading (MeSH) terms associated with MEDLINE citations, will result in better performance than learning from texutal features. We test our hypotheses through simulation, using the OHSUMED data set. Our results show that ranking functions learned from user feedback approach the performance of ranking functions learned from the entire data set using one half of the total data available. Our results also show that learning from MeSH features greatly outperforms learning from textual features.

1 Introduction

MEDLINE, the National Library of Medicine's bibliographic database, is an ubiquitous resource used globally by biomedical researchers and professionals. It contains over 16 million journal citations, with thousands more added every week [1]. Searching MEDLINE is done most often via Entrez PubMed, the NLM's search engine operating over the database. While Entrez is a robust retrieval system, users interacting with Entrez often face information overload. In part, this is due to the enormity of the database; a search for "heart disease" returns over seven hundred thousand results. However, a greater issue is the lack of relevance-based ranking of these results. Users are limited in their choice of ranking to date of publication, author names, or journal of publication. This means that the results most relevant to the user's query may be buried under thousands of irrelevant results, forcing the user to sort through them manually.

Deeper issues exacerbate the problem. To avoid manually searching through potentially thousands of search results to find the citations they need, medical professionals must spend years developing the expertise necessary to use Entrez efficiently [2]. Even experienced users may find themselves unable to construct these efficient queries if they lack deep knowledge regarding the subject of their search.

We propose to address these problems by learning ranking functions from user feedback. A ranking function learned in this way will put the most relevant results above the less relevant, putting the best information first. Learning from feedback does not require a great amount of training, so it can be done by a novice user. Furthermore, a user does not need deep domain knowledge; giving positive feedback on citations that "look right" ensures that similar citations will be ranked highly. Finally, each ranking function will be tailored to the user training it, while traditional query-based ranking methods such as tf*idf would produce the same ordering for any given query. Joachims [3] argued that, "experience shows that users are only rarely willing to give explicit feedback." We agree with this point in general, but experience has shown us that biomedical professionals are highly motivated to give feedback if the overhead of feedback is offset by utility they gain from the system.

Our hypothesis is that a ranking function learned from feedback given on a small percentage of intelligently chosen examples from a retrieval set will perform comparably to a ranking function learned from the entire retrieval set. We will explore how to choose these examples and how much feedback to request from the user. We further propose to learn our ranking functions from document metadata, as opposed to textual features. Citations in MEDLINE are annotated using the NLM's controlled vocabulary of Medical Subject Headings, called MeSH. We hypothesize that learning from these features will be superior to learning from query-based textual features. We will evaluate the quality of the rankings produced by our method, along with the amount of feedback required to produce that ranking.

2 Learning Ranking Functions from User Feedback

Presenting retrieved documents according to the likelihood of their relevance to a user's query is a standard practice in information retrieval [4]. Here, we present a framework for learning a function to produce such a relevance ranking from feedback provided by the user. As this is an online task, two factors beyond raw system performance must be addressed. First, the system must run quickly enough to provide the user a reasonable search experience. Second, the amount of feedback required for learning must be a reasonable fraction of the number of search results. If either of these factors are not well addressed, the system will provide no benefit to users over traditional search engines. We choose to employ ranking SVMs for rank function learning because of their speed, and their performance in learning ranking functions [3][5][6].To ensure users will have to provide as little feedback as possible, we employ active learning to choose examples that will be most useful for learning [7].

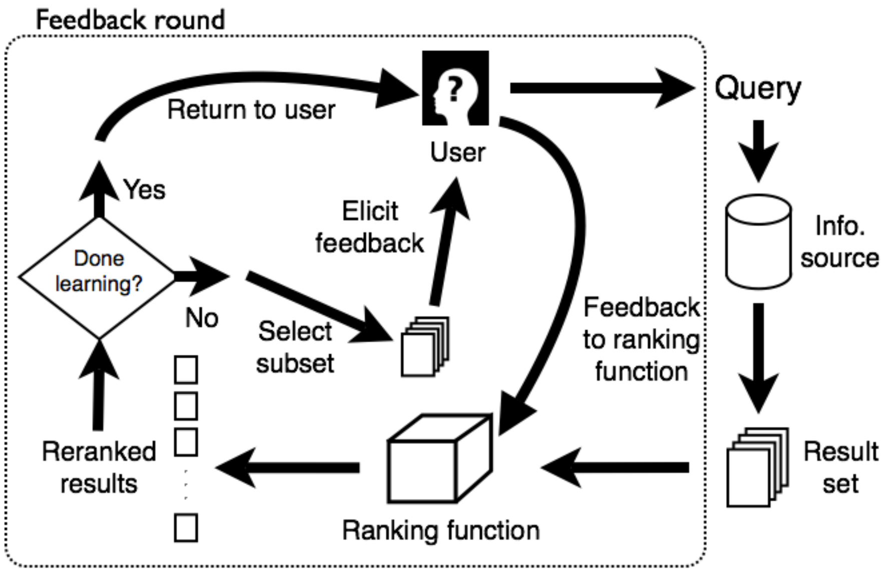

Fig. 1. Illustration of learning ranking functions from user feedback

2.1 Initial Retrieval

We take initial retrieval as given. PubMed is an excellent retrieval system, employing many recall-boosting strategies such as term expansion and synonym matching, which we do not care to replicate. For our experiments, we use the existing OHSUMED dataset (described in section 3.1) and perform no retrieval on our own. Our production system will use the Entrez eUtils[1], which will allow users to submit queries to our system just as they would to PubMed itself. Document abstracts and their associated feature vectors will be stored locally.

2.2 The Feedback Round

A "feedback round" is one iteration of choosing examples for feedback, requesting feedback from the user, learning from the feedback, and checking the stopping criterion. Future references will be made to this sequence of steps as a performance measure, indicating one measure of how much overhead the user has

[1] http://www.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html

incurred using the system. The number of rounds multiplied by the number of examples seen per round gives the total number of examples seen, which is the other overhead measure used.

2.3 Ranking

As previously stated, we learn and rank using the ranking SVM algorithm. Given a collection of data points ranked according to preference R^* with two points $\boldsymbol{d_i}, \boldsymbol{d_j} \in R^*$, and a linear learning function f, we can say

$$d_i \succ d_j \Leftrightarrow f(d_i) > f(d_j) \tag{1}$$

where $\succ$ indicates that d_i is preferred over d_j. We can define the function f as $f(\boldsymbol{d}) = \boldsymbol{w} \cdot \boldsymbol{d}$, where

$$f(d_i) > f(d_j) \Leftrightarrow \boldsymbol{w} \cdot \boldsymbol{d_i} > \boldsymbol{w} \cdot \boldsymbol{d_j} \tag{2}$$

The vector $\boldsymbol{w}$ can be learned via the standard SVM learning method using slack variables [8],

$$\begin{aligned} &\text{minimize } \langle \boldsymbol{w} \cdot \boldsymbol{w} \rangle + C \sum_{i,j \in |R|} \xi_{ij} \\ &\text{subject to } \forall (\boldsymbol{d_i}, \boldsymbol{d_j}) \in R^* : \boldsymbol{w} \cdot \boldsymbol{d_i} \geq \boldsymbol{w} \cdot \boldsymbol{d_j} + 1 - \xi_{ij} \\ &\qquad \forall (i,j) : \xi_{ij} \geq 0 \end{aligned} \tag{3}$$

Discovery of the support vectors and the generalization of the ranking SVM is done differently [3]. For data that are linearly separable, the ξ_{ij} are all equal to 0. In this case, we can view the ranking function as projecting the data points onto the separating hyperplane. In this case, the support vectors are the two points $\boldsymbol{d_i}$ and $\boldsymbol{d_j}$ nearest each other on the hyperplane. Generalization is achieved by calculating $\boldsymbol{w}$ to maximize the distance between these closest points. The distance between these two points is calculated as $\frac{\boldsymbol{w}(\boldsymbol{d_i} - \boldsymbol{d_j})}{\|\boldsymbol{w}\|}$. Taking this as our margin γ, we can, as with the classification SVM algorithm [8], maximize the margin by minimizing $\|w\|$.

2.4 Choosing Examples

In order to learn a ranking function, we require a training set. This set will be provided to us by the user in the form of explicit preference feedback on a subset of the retrieved documents. In order to ensure that we are asking for user feedback on as few examples as possible, we choose this subset via active learning.

Active learning describes a learning method wherein the learning algorithm itself has some control over which examples are added to its training set. Specifically, we need to ask a user to provide labels for some number of unlabeled examples. The learner chooses these examples based on some measure of learning utility; for example, choosing examples which will decrease the region of uncertainty in a learned function [7]. Repeated rounds of active learning improve both the learned function and the examples chosen for learning. Taking our previous measure as an example, the reduction in the region of uncertainty produces a function with better generalization power; however, reducing the region of uncertainty has an added benefit of leaving behind only examples which continue to contribute to uncertainty.

2.5 Eliciting Feedback

Feedback is elicited by asking the user to rate a document's relevance to his/her information need. We allow users to express preference on a neutral point scale ("yes", "maybe", "no"), rather than using a forced-choice ("yes or no") method, as the former shows higher measurement reliability [9]. This facilitates simulation using OHSUMED, allowing the user to rate a document as "definitely relevant", "possibly relevant", or "not relevant".

2.6 Stopping Criterion

At some point, feedback rounds must terminate. Rather than arbitrarily choosing a number of rounds or amount of feedback required before termination, feedback ends when the lists produced by the ranking function appear to be converging towards a stable list. Our convergence criterion is based on the Kendall's tau rank correlation coefficient, calculated between on the current and immediately previous orderings produced by the ranking function. Once the correlation between these rankings exceeds a certain threshold, feedback rounds terminate.

3 Experimental Design

3.1 Simulation using OHSUMED

Experiments were carried out via simulation, using the OHSUMED data set [10]. OHSUMED is a test collection of MEDLINE citations, created from the results of 106 queries run against a five-year span of MEDLINE documents. 16,141 query-document pairs were annotated for relevance, classified as definitely, possibly, or not relevant. Five queries returned no definitely relevant citations, and have been excluded from the simulations. All experiments were run ten times per query, and the results averaged.

3.2 Features for Learning

Metadata features were created from the MeSH metadata available for each document, with feature vectors built based on binary inclusion of individual MeSH terms. Arranged as a concept hierarchy, MeSH encodes the content of the citation as well as extra-textual features. While these features, such as the type of article referred to by the citation (e.g. clinical trial, meta-analysis, etc.), may be of great importance to the user, novice users may not know how to formulate queries expressing this need. Furthermore, relevance ranking systems based solely on textual features may not reflect this importance.

Textual features were taken from the OHSUMED section of LETOR [11], a learning-to-rank benchmark data set. These consist of ten "low-level" features from the abstract and title fields of the OHSUMED documents (for a total of twenty), and five "high-level" features from the combination of title and abstract. Low-level features include traditional textual measures such as normalized term frequency, tf*idf, etc., as well as features found in [6]. High-level features include measures such as BM25.

It should be noted that MeSH terms offer a further improvement over textual features in that annotators assigning MeSH terms to citations in MEDLINE have access to the entire article, while textual features for the MEDLINE user are limited to the title and abstract.

We hypothesize that ranking functions learned from MeSH data will outperform those learned from LETOR data, both in ranking performance and the overhead required to reach similar performance levels.

3.3 Example Selection

Two active learning strategies were employed for example selection. We used random sampling, simply choosing unseen examples at random, as a baseline against which these two methods will be compared.

The first active learning method is top sampling. As discussed in Cao et. al. [6], ranking functions should have their performance optimized towards top-ranked documents. Therefore, top sampling chooses unseen documents ranked highest by the current ranking function at each round for feedback. The other active learning method is mid sampling. Similar to Cohn et. al. [7], we wish to reduce uncertainty in our ranking function. A learned ranking function will rank the best and worst documents with great confidence, but less so those in the middle. These middle-ranked documents are the ones ranked with the least confidence; therefore, learning from them should result in a stronger model.

We hypothesize that both top and mid sampling will outperform random sampling, both in ranking performance and overhead cost. We further hypothesize that top sampling will outperform mid sampling in ranking performance, as mid sampling is training to improve overall performance as opposed to focusing on the performance of highly-ranked documents.

3.4 Examples per Round

The number of examples presented for feedback in each round may influence both how quickly the ranking function is learned, and the quality of the ranking function. We investigated varying between one and five examples per round.

We hypothesize that functions learned from more feedback per round will have better ranking performance than those learned from fewer examples per round. This is an obvious hypothesis to make; more examples per round means more total training examples. However, we further hypothesize that learning from more examples per round will require users to look at fewer total examples. Our intuition is that since each round of training will produce a stronger ranking function, the active learning will be better at each round compared to ranking functions trained with fewer examples.

3.5 Stopping Criterion

As previously described, our stopping criterion is based on list convergence as calculated by Kendall's tau. Our intuition is that highly correlated orderings indicate that rankings produced by the ranking function are converging, and we are therefore not learning any new information from feedback. Thresholds between 0.9 and 0.5 were investigated.

We hypothesize that higher thresholds will produce better ranking performance, but the overhead required to meet the threshold will increase.

4 Evaluation

4.1 Metrics

We evaluate ranking performance using normalized discounted cumulative gain (NDCG) [12], a commonly used measure when multiple levels of relevance are considered. Discounted cumulative gain (DCG) at position i in a ranked list of documents is calculated as

$$\text{DCG@}i = \begin{cases} r_i & \text{if } i\text{=1} \\ \text{DCG@}(i\text{-1}) + \frac{r_i}{log_2 i} & \text{otherwise} \end{cases} \tag{4}$$

where r_i is the relevance score of the document at position i. For our evaluation, relevant documents receive a score of 2, possibly relevant documents receive a score of 1, and irrelevant documents receive a score of 0. NDCG is calculated by dividing the DCG vector by an ideal DCG vector, DCG_I, calculated from an ideally ranked list (all documents scoring 2, followed by documents scoring 1, followed by documents scoring 0). Perfect ranking scores an NDCG of 1.0 at all positions. We compute NDCG@10 for our evaluation. We evaluate user overhead by counting the number of feedback rounds to produce a given ranking. Both metrics are averaged over the 101 queries used for simulation.

4.2 Results

Learning from MeSH features clearly outperformed learning from LETOR features in ranking performance. An upper bound for performance comparison was calculated by ranking documents for each OHSUMED query using a ranking

	@1	@2	@3	@4	@5	@6	@7	@8	@8	@10
MeSH	0.995	0.993	0.993	0.992	0.995	0.995	0.995	0.995	0.995	0.995
LETOR	0.624	0.634	0.622	0.617	0.606	0.604	0.596	0.593	0.596	0.596

Table 1. NDCG calculated across all queries at positions 1 through 10 for ranking SVMs trained on all data available for a query.

SVM learned from all documents in the query, for both MeSH and LETOR feature vectors. Table 1 shows SVMs learned from MeSH terms yielded nearly perfect ranking performance, vastly outperforming SVMs learned from LETOR features. A performance gain is to be expected, as the MeSH terms are tailored to this data; however, we did not expect the gain to be this great. This trend continues in the active learning experiments. As shown in table 2, across all sampling methods, thresholds, and examples per round, ranking performance of SVMs learned from MeSH features outperform their LETOR counterparts. For thresholds above 0.5, Though LETOR SVMs consistently reached convergence before MeSH SVMs, indicating a much lower overhead, the performance was consistently poor.

Random Sampling										
Threshold	MeSH					LETOR				
	1	2	3	4	5	1	2	3	4	5
0.5	0.446	0.49	0.529	0.559	0.574	0.359	0.369	0.384	0.393	0.405
	7.197	4.551	3.542	3.287	2.975	7.329	4.823	3.804	3.386	3.088
	7.2	9.10	10.63	13.15	14.88	7.329	9.646	11.41	13.54	15.44
0.6	0.464	0.5	0.532	0.566	0.604	0.358	0.373	0.388	0.401	0.411
	6.866	4.722	3.886	3.49	3.393	7.461	4.533	3.858	3.54	3.19
	6.87	9.44	11.66	13.96	16.96	7.461	9.065	11.58	14.16	15.95
0.7	0.474	0.51	0.563	0.592	0.626	0.361	0.372	0.392	0.407	0.425
	7.591	5.25	4.573	4.201	4.108	7.672	4.788	4.223	3.736	3.601
	7.591	10.5	13.72	16.8	20.54	7.672	9.576	12.67	14.94	18
0.8	0.484	0.543	0.606	0.648	0.687	0.358	0.376	0.405	0.412	0.422
	8.411	6.197	5.881	5.861	5.845	7.76	5.284	4.612	4.326	4.144
	8.411	12.39	17.64	23.45	29.22	7.76	10.57	13.84	17.3	20.72
0.9	0.521	0.604	0.668	0.726	0.77	0.37	0.384	0.419	0.44	0.456
	10.94	9.593	10.06	10.87	10.90	9.205	7.073	6.681	6.64	6.618
	10.94	19.19	30.18	43.48	54.52	9.205	14.15	20.04	26.56	33.09
Mid Sampling										
Threshold	MeSH					LETOR				
	1	2	3	4	5	1	2	3	4	5
0.5	0.455	0.488	0.504	0.533	0.549	0.346	0.367	0.381	0.389	0.399
	7.205	4.142	3.489	3.182	2.943	7.25	4.395	3.564	3.167	3.039
	7.205	8.28	10.47	12.73	14.71	7.25	8.79	10.69	12.67	15.19

Continued next page

Threshold	MeSH					LETOR				
	1	2	3	4	5	1	2	3	4	5
0.6	0.451	0.495	0.519	0.535	0.561	0.358	0.37	0.385	0.4	0.396
	7.065	4.701	3.853	3.324	3.269	7.172	4.564	3.776	3.312	3.122
	7.065	9.402	11.56	13.3	16.35	7.172	9.129	11.33	13.25	15.61
0.7	0.465	0.505	0.531	0.559	0.576	0.358	0.37	0.38	0.392	0.401
	7.43	5.064	4.2	3.961	3.794	6.957	4.662	3.979	3.703	3.46
	7.43	10.13	12.6	15.85	18.97	6.957	9.324	11.94	14.81	17.3
0.8	0.475	0.529	0.551	0.59	0.607	0.361	0.38	0.395	0.395	0.408
	8.45	5.961	5.34	5.017	4.967	7.987	5.195	4.395	4.168	4.068
	8.45	11.92	16.02	20.07	24.84	7.987	10.39	13.19	16.67	20.34
0.9	0.51	0.567	0.612	0.635	0.655	0.362	0.381	0.407	0.427	0.441
	10.54	9.471	9.303	9.079	8.866	9.195	6.574	6.413	6.014	5.975
	10.54	18.94	27.91	36.32	44.33	9.195	13.15	19.24	24.06	29.88
Top Sampling										
Threshold	MeSH					LETOR				
	1	2	3	4	5	1	2	3	4	5
0.5	0.463	0.522	0.562	0.616	0.669	0.374	0.389	0.412	0.443	0.457
	6.956	4.425	3.779	3.446	3.464	7.42	4.571	3.774	3.391	3.192
	6.956	8.85	11.34	13.78	17.32	7.42	9.143	11.32	13.56	15.96
0.6	0.481	0.543	0.6	0.666	0.724	0.373	0.395	0.427	0.447	0.468
	7.332	4.992	4.443	4.184	4.122	7.433	4.862	3.944	3.677	3.33
	7.332	9.984	13.33	16.74	20.61	7.433	9.725	11.83	11.71	16.65
0.7	0.49	0.58	0.667	0.742	0.802	0.382	0.406	0.438	0.464	0.472
	7.666	5.958	5.642	5.548	5.451	7.677	4.922	4.163	3.785	3.56
	7.666	11.92	16.93	22.19	27.26	7.677	9.844	12.49	15.14	17.8
0.8	0.536	0.66	0.767	0.827	0.866	0.387	0.413	0.456	0.483	0.502
	8.862	8.228	8.195	7.975	7.761	8.208	5.469	4.822	4.501	4.414
	8.862	16.46	24.58	31.9	38.81	8.208	10.94	14.47	18	22.07
0.9	0.648	0.799	0.866	0.897	0.918	0.407	0.461	0.498	0.538	0.541
	14.97	14.94	14.75	14.05	12.83	9.432	7.502	6.89	6.843	6.285
	14.97	29.88	44.25	56.18	64.14	9.432	15	20.67	27.37	31.43

Table 2: Performance for all sample methods, examples per round, and thresholds. Top value in each row is NDCG@10, middle value is number of rounds until the convergence threshold is met, bottom value is total number of examples seen until convergence.

Top sampling produced better ranking functions than the other two methods. Curiously, mid sampling performed worse than random sampling. This may be due to the fact that mid sampling is more likely to encounter documents ranked as possibly relevant as opposed to definitely relevant or irrelevant than random sampling. Mid sampling did incur less overhead than the other active learning methods, but it appears as though it converged to a poor final ranking.

In all cases, a greater number of examples per round produced better ranking performance. This is to be expected, as more examples per round yields a larger set of training data. More examples per round also decreased rounds to convergence; however, the decrease in the number of rounds was never great enough to lead to a decrease in the total number of examples seen.

As expected, higher thresholds for convergence resulted in higher ranking performance, at the cost of more feedback rounds. While performance climbed steadily, there was a marked jump in overhead between thresholds of 0.8 and 0.9.

5 Discussion

Overall, our results are encouraging. We have achieved ranking performance within 8.2% of perfect ranking after an average of 12.8 feedback rounds and 64.14 examples seen, using top sampling with five examples per round and a convergence threshold of 0.9.

Experimentation has shown that our hypotheses regarding ranking performance are correct, with the exception of the performance of mid sampling. However, our assumptions regarding the overhead incurred to reach convergence as it relates to features for learning and sampling methods seem to have been incorrect. Poor performance was linked with less overhead, with better performance always demanding more overhead. While rounds to convergence fell slightly as the number of examples per rounds increased, this small decrease is insignificant as the overall number of examples seen by the user increased.

This led us to investigate whether the number of examples seen was the dominant predictor of performance. As shown in figure 2, however, sampling method played a greater role in performance. Top sampling provided better ranking performance for any number of examples seen, in many cases requiring fewer than half the number of examples to reach performance similar to the other active learning methods.

A note must be made regarding the stopping criterion. Since termination is determined as a function of learning, it effectively falls to the active learning technique to ensure that termination is not premature, as choosing uninformative examples will cause little to no shift in the ranking function. If this were the case, the learning process would not fulfill its potential, denied the chance to exhaust its stock of “good” examples to learn from. The effect of this would be that an active learning method which could potentially perform as well as another method would have worse performance and fewer total examples seen than a method which did not end prematurely. Examples of this happening may be present in this work, especially at high thresholds looking at 4 or 5 examples per round.

We argue that this effect is likely to be minimal. It is clear that at lower thresholds and lower examples per round, the active learning method itself is the dominant factor for performance. In figure 3, we see that each active learning method tends to improve as the number of examples increases; however, at no

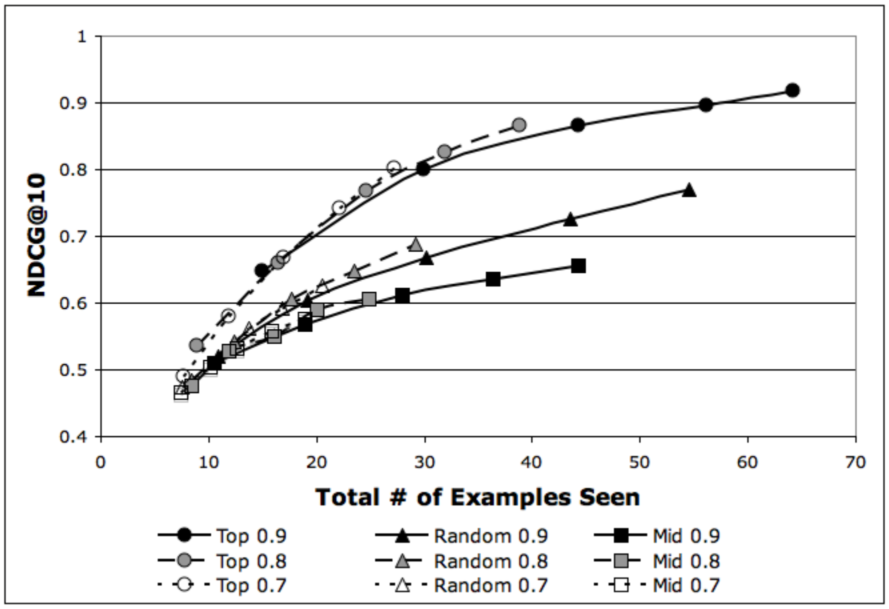

Fig. 2. Comparison of the total number of examples seen to NDCG@10 for all sampling methods, at thresholds 0.7, 0.8 and 0.9. Markers indicate number of examples per round, from one to five.

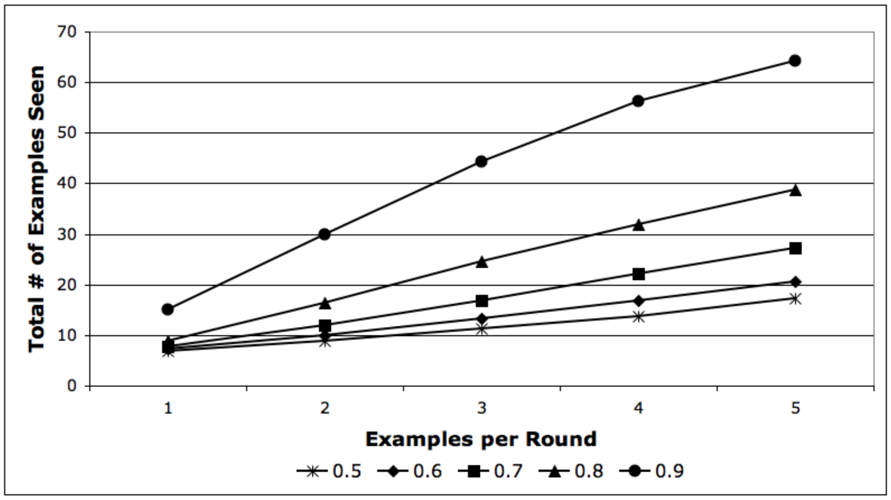

Fig. 3. The effect of increasing the number of examples per round on the total number of examples seen, across all thresholds, for top sampling.

point does it appear that a method would "catch up" to a higher performing method if allowed to continue learning.

The remainder of our analysis focuses on factors affecting top sampling. Something to note in figure 2 is that performance gains began leveling off after reaching an NDCG@10 of around 0.8, requiring increasingly more examples for smaller gains in performance. Considering the OHSUMED queries returned 152.27 documents on average, it may appear that decent performance requires feedback on an unreasonable percentage of the returned data. Recall, however, that queries to MEDLINE often result in thousands of results. Further investigation is required to see if queries which return such large results sets require feedback on a similar percentage of documents, a similar number of documents, or something in between.

We see in figure 3 that increasing examples per round increased the total number of examples seen before the convergence threshold was reached. This ran counter to one of our hypotheses; we expected that seeing more examples in each feedback round would reduce the total number of examples required to meet the convergence threshold. As this was not the case, and since examples per round had only a small effect on rounds until convergence, rounds until convergence must be dependent almost entirely on the convergence threshold.

We must conclude, therefore, that examples per round and the convergence threshold may be largely immaterial to the learning process. If ranking performance is tied only to the active learning method and number of examples seen, there may simply be a lower bound on the number of rounds required for effective active learning to take place, requiring a number of examples per round equal to the number of examples needed to reach the desired ranking performance divided by this number of rounds. Further investigation is required to determine this lower bound, if indeed it exists.

6 Related Work

Much of the literature on active learning for SVMs has focused on classification, as opposed to ranking [13][14]. Brinker [15] applied active learning to learning ranking SVMs; however, this research focused on learning label ranking functions, which is a fundamentally different task from document ranking. Tong and Chang [16] used pool-based active learning for image retrieval. Their method of selecting examples for labeling was based on the finding that by choosing examples which shrink the size of the version space in which the optimal weight vector $\mathbf{w}*$ can lie, the SVM learned from those examples will approximate $\mathbf{w}*$. Therefore, examples are chosen which will most nearly bisect the version space they occupy. This was achieved in practice by choosing examples based on their proximity to the SVM boundary; examples close to the boundary are likely to be more centrally located in the version space, and are thus more likely to bisect it.

Ranking SVMs have been used to learn ranking functions from implicitly generated feedback [3][17]. Yu [5] noted that methods such those in [16] could not be extended to learning ranking SVMs, as the ranking problem is more complex, and thus carried out selective sampling for learning ranking SVMs by selecting the most ambiguously ranked examples for labeling. This was done by noting that ambiguity in ranking could be measured based on how similarly the examples were ranked, with the closest pairs of examples being the most ambiguous. This method for selection is directly analogous to that in [16], even though it does not address reduction in version space; just as the support vectors in a classifying SVM are those examples closest to the SVM boundary, the support vectors in a ranking SVM are those examples that are most closely ranked. You and Hwang [18] used a similar framework and data set to learn ranking in a context-sensitive manner. Both of these works focused on general data retrieval, as opposed do document retrieval.

MeSH terms have been used as a tool to aid biomedical professionals both in performing searches on MEDLINE, and in analyzing search results [19][20][21]. In particular, Lin et. al. [21] used MeSH terms along with keywords to generate labels for documents clustered together using textual features, while Blott et. al. [20] clustered based on the MeSH terms themselves. To the best of our knowledge, ours is the only existing work which attempts document ranking using solely MeSH terms.

7 Conclusion and Future Work

We have presented a framework for learning ranking functions from user feedback, designed for biomedical professionals searching MEDLINE. We have shown that learning these functions using MeSH metadata is superior to learning from textual features, and that by employing active learning we can achieve near perfect ranking performance using less than half of the available data. Questions remain regarding whether the amount of data required to achieve this performance is proportional to the size of the number of documents retrieved, and whether there is a lower bound on the number of feedback rounds required to gain the benefit of active learning.

Future work will investigate these questions. It will also include implementation of the system as an web-based search utility. A user study will be conducted with parameters similar to experiments presented here in order to assess its performance on “live” queries. System capabilities will be expanded to allow users to save results of previous searches along with their associated ranking functions, as well as applying previously learned ranking functions to new search results.

The system presented here will also be adapted to other domains, particularly that of legal discovery. As discovery is often carried out over collections containing millions of documents, we can expect to see feedback on hundreds or thousands of documents, instead of dozens. However, we clearly cannot expect users to give feedback on a percentage of retrieved documents similar to the percentage used in these experiments. Answering the question of how much feedback is required to learn a reasonable ranking function will be central to the feasibility of application to this domain.

References

1. National Library of Medicine: MEDLINE fact sheet (May 2008)
2. Bronander, K.A., Goodman, P.H., Inman, T.F., Veach, T.L.: Boolean search experience and abilities of medical students and practicing physicians. Teaching and Learning in Medicine **16**(3) (Sep 2004) 284–9
3. Joachims, T.: Optimizing search engines using clickthrough data. In: Proc. ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (SIGKDD '02). (2002) 133–142
4. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley (1999)
5. Yu, H.: SVM selective sampling for ranking with application to data retrieval. In: Proc. Intl. ACM SIGKDD Conf. on Knowledge Discovery and Data Mining (SIGKDD '05). (Jun 2005)
6. Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W.: Adapting ranking SVM to document retrieval. (2006)
7. Cohn, D., Atlas, L., Ladner, R.: Improving generalization with active learning. Machine Learning **15**(2) (May 1994) 201–221
8. Christianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods. Cambridge University Press (2000)
9. Churchill Jr, G.A., Peter, J.P.: Research design effects on the reliability of rating scales: A meta-analysis. Journal of Marketing Research **21**(4) (1984) 360–375
10. Hersh, W., Buckley, C., Leone, T., Hickam, D.: OHSUMED: An interactive retrieval evaluation and new large test collection for research. In: Proc. ACM SIGIR Intl. Conf. on Information Retrieval (SIGIR '94). (1994)
11. Liu, T.Y., Xu, J., Qin, T., Xiong, W., Li, H.: Letor: Benchmark dataset for research on learning to rank for information retrieval. In: Proc. ACM SIGIR Intl. Conf. on Information Retrieval (SIGIR '07). (2007)
12. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of ir techniques. ACM Transactions on Information Systems **20**(4) (October 2002) 422–446
13. Tong, S., Koller, D.: Support vector machine active learning with applications to text classification. Journal of Machine Learning Research (2001) 45–66
14. Ertekin, S., Huang, J., Bottou, L., Giles, C.L.: Learning on the border: Active learning in imbalanced data classification. In: Proc. ACM Conf. on Information and Knowledge Management (CIKM '07). (2007)
15. Brinker, K.: Active learning of label ranking functions. In: Proc. Intl. Conf. on Machine Learning. (2004)
16. Tong, S., Chang, E.: Support vector machine active learning for image retrieval. ACM Int. Conf. on Multimedia (MM '01) (2001)
17. Radlinski, F., Joachims, T.: Query chains: Learning to rank from implicit feedback. In: Proc. ACM Intl. Conf. on Knowledge Discovery and Data Mining (SIGKDD '05). (2005)
18. You, G., Hwang, S.: Personalized ranking: A contextual ranking approach. In: Proc. ACM Symp. on Applied Computing (SAC '07). (2007)
19. Haynes, R.B., McKibbon, K.A., Wilczynski, N.L., Walter, S.D., Werre, S.R.: Optimal search strategies for retrieving scientifically strong studies of treatment from MEDLINE: analytical survey. BMJ (British Medical Journal) **330**(7501) (May 2005) 1179
20. Blott, S., Camous, F., Gurrin, C., Jones, G.J.F., Smeaton, A.F.: On the use of clustering and the MeSH controlled vocabulary to improve MEDLINE abstract search. (2005)

21. Lin, Y., Li, W., Chen, K., Liu, Y.: A document clustering and ranking system for exploring MEDLINE citations. Journal of the American Medical Informatics Assn. **14**(5) (2007) 651–661