Jointly Modeling Relevance and Sensitivity for Search Among Sensitive Content

Mahmoud F. Sayed
Computer Science and UMIACS
University of Maryland, College Park
mfayoub@cs.umd.edu

Douglas W. Oard
iSchool and UMIACS
University of Maryland, College Park
oard@umd.edu

ABSTRACT

Current search engines are designed to find what we want. But unprocessed archival collections can't be made available for search if they contain sensitive content that needs to be protected. Traditionally, content if first examined through a sensitivity review process, which becomes more difficult and time-consuming as content volumes increase. To mitigate these costs and delays, search technology should be capable of providing access to relevant content while protecting sensitive content. This paper proposes an approach that leverages learning to rank techniques. We use learning to rank to optimize a loss function that balances the value of finding relevant content with the imperative to protect sensitive content. In the experiments, a LETOR benchmark dataset, OHSUMED, is used with a subset of the MeSH labels representing the sensitive documents. Results show the efficacy of the proposed approach in comparison with some simpler baselines.

CCS CONCEPTS

• **Information systems** → **Learning to rank**;

KEYWORDS

Learning to rank; sensitivity review; evaluation

ACM Reference Format:
Mahmoud F. Sayed and Douglas W. Oard. 2019. Jointly Modeling Relevance and Sensitivity for Search Among Sensitive Content. In *42nd Int'l ACM SIGIR Conference on Research & Development in Information Retrieval (SIGIR'19), July 21–25, 2019, Paris, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3331184.3331256

1 INTRODUCTION

In late 2014, presidential candidate Hillary Clinton sent 30,490 work-related email messages to the State Department and asked that they be reviewed and released as quickly as possible. Nearly a year later, with 25 people assigned to the office coordinating the review process, the review process was completed. In late 2014, presidential candidate Jeb Bush chose a different approach, releasing all of the approximately 280,000 email messages from his time as Governor of Florida. That email was posted to the Internet and then removed two days later, after independent sources identified the presence of sensitive content. Neither approach is able to provide responsive access at reasonable cost while protecting sensitive information.

The fundamental issue is that today's search engines are designed with a single fundamental goal: to help us find that which we want to see. Paradoxically, the very fact that they do this well means that there are many collections that we are not allowed to search. Citizens are not allowed to search some government records because there may be intermixed information that needs to be protected. Scholars are not yet allowed to see much of the growing backlog of unprocessed archival collections for similar reasons. These limitations, and many more, are direct consequences of the fact that today's search engines are not designed to protect sensitive information.

If sensitive content were marked as sensitive at the time of creation, protecting it would be easy. In an earlier era when information was scarce relative to human attention, segregating sensitive information from that which could be made public was the norm. For example, we could presume that newspaper articles were intended to be public, and that telephone calls were intended to be private. Today, however, it is human attention that is scarce relative to the quantity of information that we all generate, and digital media increasingly collapse what used to be segregated information contexts. As a result, our digital records are an intermixed cacophony of the sensitive, the important, and the banal.

In this paper, we begin to address that challenge by exploring a new class of ranking approaches designed to effectively search among secrets. Our goal is to balance the user's interest in finding relevant content with the provider's interest in protecting sensitive content. We propose an approach that leverages evaluation-driven Learning to Rank (LtR) techniques. In LtR, each document is represented by a feature vector of possible indicators of relevance, and (for training) a target relevance label. Then learning techniques are employed to develop a ranker that optimizes some defined loss function. LtR approaches can be broadly divided into 3 groups [17]: 1) Pointwise, 2) Pairwise, and 3) Listwise approaches. Our work focuses on listwise approaches, as they seek to directly optimize an objective function that can be matched to the evaluation measure. The key, therefore, is to design an evaluation measure that reflects the goal of finding relevant documents while protecting sensitive documents.

Of course, experimenting on sensitive content ultimately requires access to sensitive content. Initially, however, it would be useful to have some reasonable surrogate for the problem that can be widely shared. We therefore experiment in this paper on a collection of medical documents, demonstrating that listwise LtR rankers

SIGIR '19, July 21–25, 2019, Paris, France

ACM ISBN 978-1-4503-6172-9/19/07…$15.00
https://doi.org/10.1145/3331184.3331256

trained to optimize an appropriate loss function can achieve better results than the more straightforward approaches of simply filtering out sensitive documents (either before or after retrieval).

To the best of our knowledge, this is the first attempt to balance between relevance and sensitivity in a single ranking framework. The contributions of this paper are as follows.

(1) We introduce Cost Sensitive Discounted Cumulative Gain (CS-DCG), a variant to DCG that balances between relevance and sensitivity. Unlike DCG, CS-DCG can become negative, and hence we need to compute its lower and upper bounds to use the normalized form (nCS-DCG) to meaningfully average across multiple queries.
(2) We propose some baselines that are based on filtering out sensitive content (either before or after retrieval) or downgrading relevance labels for sensitive documents.
(3) We propose a novel usage of listwise learning to rank algorithms to optimize for the new evaluation metric by modifying their loss functions.
(4) We propose a simple cluster-based replacement strategy that can further improve nCS-DCG, and show that such a strategy can sometimes reduce the number of queries with negative CS-DCG results.
(5) We evaluate the effectiveness of our proposed LtR approach along with several baselines using the LETOR OHSUMED test collection, with a subset of the MeSH labels in that collection serving as a surrogate to represent sensitivity.

The remainder of this paper is organized as follows. Section 2 discusses related work. Section 3 describes the modeling of relevance and sensitivity and different approaches to tackle the issue of search among sensitive content. The experimental setup and results are presented in Sections 4 and 5 respectively, followed by discussion in Section 6. Section 7 concludes the paper and suggests directions for future research.

2 RELATED WORK

Learning to rank has become a popular approach for creation of tailored retrieval models because of its flexibility and ability to learn from large amounts of training data. LtR approaches can be broadly divided into 3 groups [17]: 1) Pointwise, 2) Pairwise, and 3) Listwise approaches. In the pointwise approach, the ranking problem is transformed into classification, or regression, and existing methods for classification or regression can be applied. In this approach, each document is treated independently. In our work, we used linear regression [7] with a squared loss function as an example from that category.

In the pairwise approaches, ranking is transformed into a binary classification task in which the goals is to tell which document in a given pair is more relevant. During training, document pairs with different ground truth relevance values are used. The pairwise approach includes Ranking SVM [14, 18], RankBoost [11], RankNet [2], GBRank [31], IR SVM [5], Lambda Rank [4], and LambdaMART [3, 26]. In our work, we use LambdaMART [3, 26] with a cross entropy loss function multiplied by the change in the IR measure (e.g. NDCG) as an example of a pairwise approach.

The listwise approach addresses the ranking problem in a more straightforward way by taking ranked lists as instances for both learning and prediction. The group structure of ranking is maintained and ranking evaluation measures can be more directly incorporated into the loss functions in learning. The listwise approach includes ListNet [6], ListMLE [27], AdaRank [29], SVM MAP [30], Coordinate Ascent [19], Soft Rank [25], and others [13, 15, 20]. In our work, we use AdaRank and Coordinate Ascent as two examples of listwise approaches. We selected more than one approach because we want to explore the degree to which training a listwise approach by nCS-DCG works well with different ranking algorithms.

Multi-objective learning to rank is becoming more prevalent because search results are in practice not evaluated just on relevance, but also on other desirable characteristics (e.g. freshness, novelty, and user effort for examining documents in session search [24]). Dai et al. [8] develop a multi-objective learning to rank algorithm for freshness and relevance by extending a state-of-the-art divide and conquer ranking approach [1]. Lioma et al. [16] present two types of evaluation measures that are designed to measure effectiveness based on both relevance and credibility in ranked lists of retrieval results. Our proposed evaluation metric is similar to their Normalized Weighted Cumulative Score (NWCS), except that we don't discount the cost of sensitive documents.

Svore et al. [23] take a different approach, using two complementary measures. The use NDCG as their "top-tier" measure, and a relevance measure derived from click data as a second-tier measure; the use of this second-tier measure is shown to significantly improve while leaving the top-tier measure largely unchanged. In our work, by contrast, we are interested in balancing between relevance and sensitivity which are competing, since optimizing for relevance only may disclose sensitive content.

Dong et al. [9, 10] propose to adjust the relevance labels based on the freshness of the Web page. For stale pages, they demote the relevance label by one grade. We use this same approach as one of our proposed baselines, demoting the relevance labels for sensitive documents.

3 SEARCH AMONG SENSITIVE CONTENT

When searching among content that is not sensitive, it suffices to learn an effective ranking function. For search among sensitive content, we must also learn to identify which of the documents in that ranking contain sensitive content.

In this section, we describe different approaches to study how we can modify a search engine results so that it takes the predicted sensitivity of each document into account. Basically, our setup is comprised of the following components.

(1) A learning to rank (LtR) model that takes a query and a set of documents as input, and outputs a ranked list of the documents based on their relevance to the query.
(2) A sensitivity classifier that takes a document as an input, and outputs the document's probability of sensitivity.
(3) A sensitivity filter that excludes documents if they are predicted to be sensitive.

3.1 Cost Sensitive Performance Metric

Comparing approaches using an evaluation measure that pays attention only to relevance would not make sense in out setting, as we would naturally expect the focus of our systems on protecting sensitivity to result in lower scores by such a measure. When searching among sensitive content, we must therefore consider both the gain that results from showing a relevant document and the costs that accrue for showing a sensitive document. To illustrate how this might be done, we introduce a new evaluation measure that is an extension of Discounted Cumulative Gain (DCG), where each document in the ranked list has its gain based on the degree of relevance (e.g., highly relevant or somewhat relevant) and that gain value is then discounted based on the document's rank.

Our Cost-Sensitive Discounted Cumulative Gain (CS-DCG) measure additionally incorporates a cost based on the document's sensitivity. In the simplest model, the cost for showing a sensitive document would be applied additively in a way that is independent of where the document is in the ranked list. So CS-DCG cut off at rank position k can be defined as:

$$CS\text{-}DCG_k = \sum_{i=1}^{k}(\frac{g_i}{d_i} - c_i) \tag{1}$$

where g_i is the (relevance) gain of showing the document that is at rank position i, d_i is the discount factor for showing that document at rank position i, and c_i is the (sensitivity) cost of showing the document that is at rank position i. Sensitivity costs for showing non-sensitive documents are set to 0, and in practical applications we will be interested in cases in which sensitivity costs for showing relevant documents are fairly large in comparison with the maximum possible discounted gain (which accrues when showing a highly relevant document at the top of the ranked list).

Because the maximum value of DCG depends on the number of relevant documents and their degree of relevance, DCG values are usually normalized to the interval [0, 1] (forming nDCG) when averaging across topics [24], as is common in information retrieval evaluation. CS-DCG must similarly be normalized (forming nCS-DCG) when averaging across topics. Unlike DCG, however, CS-DCG can be negative or positive, so we need to know both its upper and lower bounds. Similarly to DCG, these bounds can be computed by the greedy rule-based algorithm shown in Algorithm 1 for the case in which $\max(c_i) > \max(g_i)$.

Upper bounds are computed similarly. Once we get the upper (best) and lower (worst) bounds for CS-DCG for a certain query q, we can normalize it as follows.

$$nCS\text{-}DCG = \frac{CS\text{-}DCG - CS\text{-}DCG_{worst}}{CS\text{-}DCG_{best} - CS\text{-}DCG_{worst}} \tag{2}$$

So nCS-DCG would be equal to 1 when the documents are ranked indistinguishably from the best possible ranking, and equal to 0 when documents are ranked indistinguishably from the worst possible ranking. Naturally, the best practical nCS-DCG is achieved when the documents that are most relevant appear as high as can be achieved in the ranked list and when sensitive documents never appear anywhere above the cutoff in the ranked list. Note that we must cut the ranked list off at some point (hence our specification of k) if we are to have any chance of avoiding showing sensitive documents, since in a full ranked list every document would appear somewhere in the list.

Algorithm 1

Computing a query-specific lower bound on CS-DCG_k

```
Start with an empty ranked list of size $k$ and fill from top to
bottom with non-relevant and sensitive documents that are
selected in any order.
if ranked list still has less than $k$ documents then
    fill in the empty slots from bottom to top with relevant and
    sensitive documents that are selected in any order
end if
if ranked list still has less than $k$ documents then
    fill in the empty slots from top to bottom with non-relevant
    and non-sensitive documents that are selected in any order
end if
if ranked list still has less than k documents then
    fill from bottom to top with relevant and non-sensitive documents that are selected in any order
end if
```

3.2 Proposed Baselines

In order to tackle the issue of balancing between relevance and sensitivity, we suggest different baselines for how to make our three components interact with each other.

Baseline 1: Relevance Only. In this approach, an LtR model is trained using all documents without considering their sensitivity levels, as shown in Figure 1a. At testing time, documents are fed to the resulting model along with a test query/topic, and then the model outputs the top k documents.

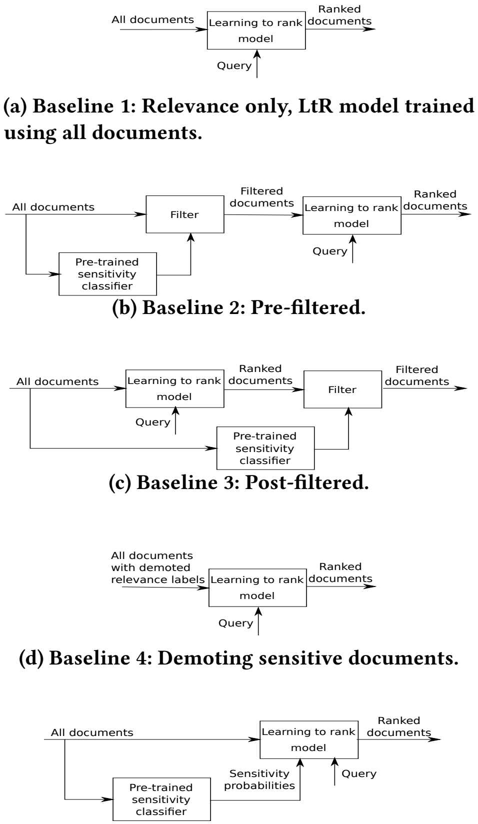
Baseline 2: Pre-filtering then Relevance Only. In this approach, at training time, documents are fed to a filter along with their sensitivity probabilities, as shown in Figure 1b. Then the filter omits any document that is predicted as sensitive. After that, the filtered documents are used to train an LtR model. At testing time, documents are fed to the resulting model along with a test query/topic, and then the model outputs the top k documents.

Baseline 3: Relevance Only then Post-filtering. Similarly to the first approach, an LtR model is trained using all documents. At testing time, documents are fed to the resulting model along with a test query/topic, and then the model outputs a ranked list. But before showing the results, documents that are predicted to be sensitive are filtered out and the remaining top k documents are returned, as shown in Figure 1c.

Baseline 4: Demoting relevance scores for sensitive documents. Inspired by Dong et al. [10] and [9], we demote the relevance scores of a query-document pair by one relevance grade when the document is predicted as sensitive. Then the modified query-document pairs are used to train an LtR model. At testing time, documents are fed to the resulting model along with a test query, and then the model then outputs a ranked list.

3.3 Proposed Approach

List-wise LtR algorithms allow us to produce models that optimize customizable loss functions. A well known example is AdaRank [29], which is based on boosting. In each iteration of AdaRank, it learns a weak ranker that minimizes a loss function. A common approach is to base the loss function on the evaluation measure (e.g., seeking



(a) Baseline 1: Relevance only, LtR model trained using all documents.

(b) Baseline 2: Pre-filtered.

(c) Baseline 3: Post-filtered.

(d) Baseline 4: Demoting sensitive documents.

(e) Opt. nCS-DCG: Jointly modeling relevance and sensitivity.

Figure 1: Alternative approaches for search among sensitive content.

to minimize 1-nDCG). Since we aim at balancing between relevance and sensitivity, we propose 1 - nCS-DCG as the loss function to be minimized. We name this approach *Opt. nCS-DCG*. In addition to modifying the loss function, we inject the probability of document being sensitive (or its complement) as part of its vector representation while building an LtR model as shown in Figure 1e.

4 EXPERIMENTAL SETUP

In this section, we describe the experiments we performed to compare between the four baselines and our proposed approach that are described in Section 3.

4.1 Test Collection

To the best of our knowledge, there is no public test collection that is annotated for both relevance and sensitivity. For our experiments we therefore chose a collection that has been labeled for multi-level "graded" relevance and for topical categories. By selecting one or more topical categories as surrogates for sensitivity, we can then simulate the task that we ultimately wish to perform.

We chose to use one of the LETOR benchmark datasets that has these characteristics, OHSUMED [22] which has been used in many research studies [5, 28, 29]. OHSUMED is a collection of articles from 270 medical journals in the period 1987-1991. The collection consists of 348,566 records, each having a title, abstract, Medical Subject Heading (MeSH) indexing terms,[1] author, source, and publication type. There are 106 topics, each consisting of patient information and a brief statement of the information need. In each case, we use only the information need statement as the query. For each topic, a subset of the documents were judged for relevance. In total, there are 16,140 judged query-document pairs, each of which has a relevance judgment that indicates whether the document is definitely, possibly, or not relevant to the query. It was the goal of the developers of the collection to span as many of the relevant documents as possible within the judged set, although of course there may be additional unjudged documents that are relevant. In keeping with common practice when using this collection, we treat documents that were not judged with respect to a topic as not relevant. Because some documents were judged for relevance to more than one topic, there are a total of 14,430 unique documents for which relevance to one or more topics has been judged. Table 1 shows the statistics for judged documents, both in aggregate and per topic.

	Highly relevant	Somewhat relevant	Not relevant
Sum	2,252	2,585	11,303
Avg.	21.25	24.39	106.63
Stdev.	23.20	20.38	46.76
Median	13	17.5	106

Table 1: Statistics about OHSUMED documents judged for relevance.

4.2 Sensitivity Classification

As a surrogate for sensitivity, we selected from among the MeSH labels used in our test collection. In OHSUMED, one document can have more than one MeSH label. We selected a set of labels, *S*, to be considered as the sensitive. If any document has at least one of the labels belonging to S, then the document is considered sensitive; otherwise it is considered non-sensitive. For our experiments, we selected 2 MeSH labels for *S*: C12 (Male Urogenital Diseases) and C13 (Female Urogenital Diseases and Pregnancy Complications). We initially chose these labels simply because we felt they were topics that some people might actually consider to be sensitive, but before settling on them we also checked their prevalence among the full set of OHSUMED documents (8.4%) and among the documents that have been judged for relevance (12.2%). These seem to us to reflect a sensitivity prevalence representative that we might see in some real application for search among sensitive content. As Figure 2 shows, some queries have many relevant documents that are sensitive, whereas others do not. This means means that the challenges posed by sensitivity are topic-dependent, which also reflects what we would expect to see in real applications.

To approximate the degree of accuracy that might be seen in an actual application for search among sensitive content, we trained a scikit-learn [21] logistic regression text classifier. We used the

[1] A full list of MeSH terms can be found at ftp://nlmpubs.nlm.nih.gov/online/mesh/

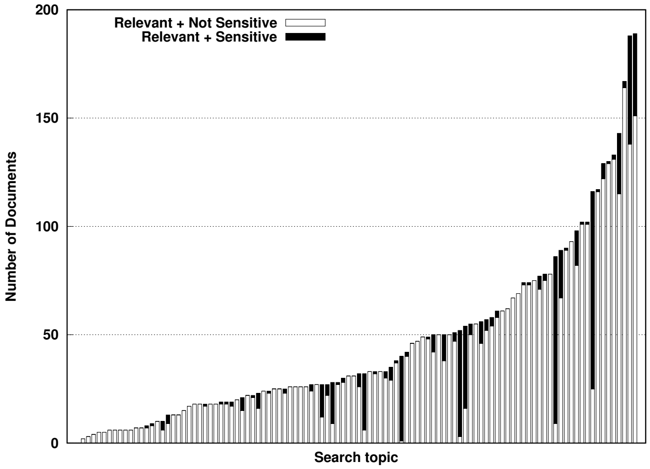


Figure 2: Number of relevant documents (to any degree) per topic in the test set, with documents that are both relevant and sensitive shown in red at the top of each bar.

14,430 documents that were judged for relevance as our test set, and the remaining 334,136 documents in the OHSUMED collection for training. Each document is represented by a tf-idf vector for the words found in the title and abstract. We tuned the hyper-parameters using a grid-search with 5-fold cross-validation on the training set. Also, by varying the probability threshold, we found out that having a threshold of 0.84 maximizes F_1. The resulting binary classifier has precision = 0.81, recall = 0.70, F_1 = 0.75, and accuracy = 94.3%. From the fact that our classes are unbalanced and that a false negative (mistakenly predicting sensitive document as non-sensitive) is more important to avoid that a false positive, we may tune the hyper-parameters to optimize for F_4, which gives considerably more weight to recall than precision.[2]

4.3 Selection of Learning to Rank Algorithms

For testing our baselines and our approach, we used different algorithms to build an LtR model. We selected at least one representative algorithm from each of three LtR approaches [17]: 1) Pointwise, 2) Pairwise, and 3) Listwise. We also built a *BM25* baseline, which is a simple and often quite effective retrieval model that is not based on learning to rank.

Linear regression [7] is a pointwise approach that learns a linear ranking function which maps a feature vector to a relevance score. Then documents are ranked based on their predicted relevance scores. In this approach each document is treated independently.

$$L(F(x), y) = \sum_{i}^{m} (f(x_i) - y_i))^2 \tag{3}$$

where f is the ranking model, x_i is the feature vector of document i, and y_i is the true relevance label.

LambdaMART [26] is a pairwise approach which is based on Multiple Additive Regression Trees (MART) [12] where the output of the model is a linear combination of the outputs of a set of regression trees. The loss function is defined on the basis of pairs of documents with different relevance scores.

$$l(f; x_i, x_j, y_{ij}) = -\overline{P}_{ij} \log P_{ij}(f) - (1 - \overline{P}_{ij}) \log (1 - P_{ij}(f)) \tag{4}$$

where and y_{ij} is the true label of whether document i has a higher rank than document j or not. $\overline{P}_{ij}$ is the target probability, which is equal to 1 when y_{ij} is 1, and 0 otherwise, and $P_{ij}(f)$ is the model output probability. Then the loss function is multiplied by the change in the IR measure, e.g. Δ NDCG, as in LambdaRank [4]. In our experiments, we use the validation set to determine the ensemble of trees with the best score on the evaluation measure.

AdaRank [29] is a listwise approach which is based on boosting. In each iteration, it learns a weak ranker that minimizes a loss function. A common approach is to base the loss function on the evaluation measure (in our case, 1 minus nCS-DCG). Then the final model is a linear combination of the weak learners. In our experiments, we use the validation set to determine the number of weak rankers that collectively achieve the best score on the evaluation measure.

$$L(F(x), y) = \sum_{i}^{m} \exp\{-E(\pi(q_i, d_i, f)), y_i\} \tag{5}$$

where $E(\pi(q_i, d_i, f))$ is any evaluation measure whose values are in the range [-1, 1]

Coordinate Ascent [19] is another listwise approach which is iteratively optimizing a multivariate objective function by solving a series of one-dimensional searches while holding other parameters fixed. In our experiments, we use the validation set to determine the set of weights with the best score on the evaluation measure over 5 restarts.

4.4 Implementation

Our implementation is based on the Ranklib library which is part of the Lemur project.[3] It contains implementations of several learning to rank algorithms written in Java. We added classes for our new evaluation measures (CS-DCG and nCS-DCG) and made some minor changes to the interface code to handle more command-line parameters. The code is available at https://github.com/mfayoub/SASC.

5 RESULTS

We start our experiments by verifying the expected results of our baseline techniques and our proposed approach when with a perfect (oracle) sensitivity classifier built using the actual MeSH class labels in the collection (accuracy = 100%). Then we investigate the performance of our baselines and our proposed approach using both CS-DCG and nCS-DCG. We prefer approaches for which CS-DCG is almost always non-negative (thus indicating that searching is better than not searching), and among such approaches we prefer approaches that maximize nCS-DCG. Note that these two measures provide complementary views: CS-DCG tells us what is happening with individual queries, but it can not be meaningfully averaged; nCS-DCG tells us when an approach is doing well on average, but

[2] Building an actual sensitivity classifier was not our objective in this paper, so we claim only that a classifier with the characteristics we report can be built given these categories and this (rather large) amount of training data.

[3] https://www.lemurproject.org/

	Highly relevant	Moderately relevant	Not relevant
Retrieved	G_h	G_m	0
Not retrieved	0	0	0

Table 2: Gain matrix for documents depending on the relevance level

	Sensitive	Not sensitive
Retrieved	C_s	0
Not retrieved	0	0

Table 3: Cost matrix for documents depending on the sensitivity level

it can not tell us whether that approach would be better than doing nothing. For all our experiments we report results from 5-fold cross-validation in which three folds are used for training, one fold is used for validation, and one fold is used for testing.

We numerically interpret the relevance levels and compute the discount using the default approach of Ranklib as shown in equation (1).

$$g_i = 2^{rel} - 1, d_i = \log(i + 2) \tag{6}$$

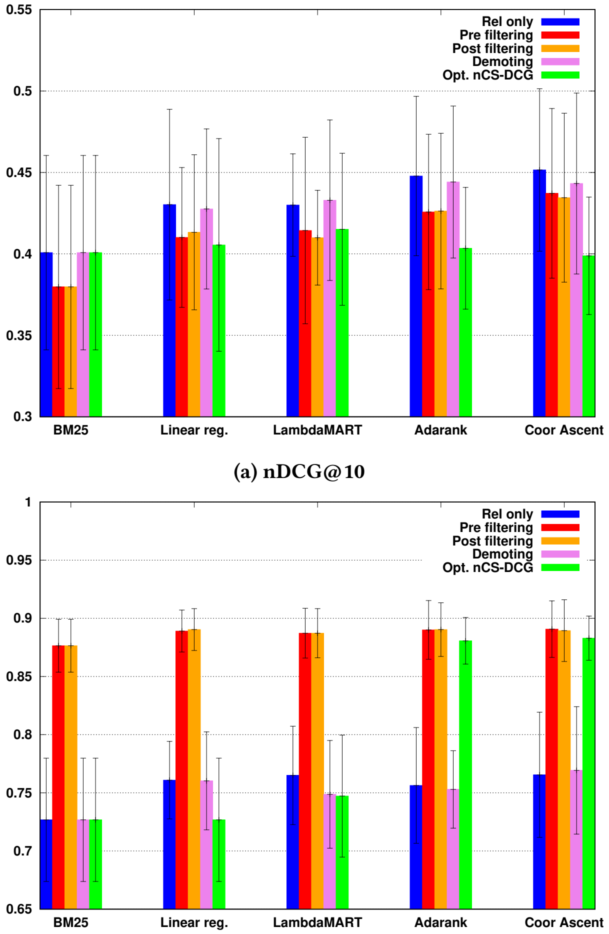
where i is the rank and rel is relevance score. In OHSUMED, rel is equal to 2, 1, or 0 if the document is highly relevant, moderately relevant, or not relevant, respectively, with respect to the query. So we set $G_h = 3$ and $G_m = 1$ in Table 2.

We are interested in cases where showing a sensitive document incurs substantial penalty, but we are not interested in cases in which that penalty is so large as to be effectively infinite (because returning any documents would not be rational in such cases). For a cutoff k, we therefore chose to study cases in which the cost of showing a sensitive document is a bit less than the maximum discounted cumulative gain we could get if all documents in the first k positions were relevant. In other words, we are interested in cases in which the system has the potential to recover from one mistakenly shown sensitive document by showing many relevant documents that are not sensitive documents. As Table 1 shows, on average a query has 21.25 documents that are highly relevant. For a cutoff = 10, the maximum DCG for the first 10 positions is thus equal to 13.63. We therefore set the cost $C_s = 12$ in Table 3.[4]

5.1 Oracle Upper Bounds

Under the ideal conditions when we have a perfect sensitivity classifier (accuracy = 100%), we expect that no sensitive documents would appear in any ranked result list. Indeed, that's what we see.

We evaluate our baselines and our proposed system using nDCG and nCS-DCG in order to illustrate the difference between the two measures. Considering nDCG, we should find that ranking based only on relevance would achieve the best results, and as Figure 3a shows, that's what we see.



(a) nDCG@10

(b) nCS-DCG@10

Figure 3: Ranking performance of different approaches with a perfect sensitivity classifier. Error bars are the standard deviations of performance across five cross-validation folds

Moreover, as we would expect, Figure 3b shows that pre-filtering or post-filtering outperform other approaches. This is because with a perfect classifier there are simply no sensitive documents remaining that might be shown. In some ranking algorithms, post-filtering performs better than pre-filtering, as we might expect from the larger LtR training folds that are available with post-filtering (in pre-filtering, the sensitive documents are removed before training; in post-filtering they are removed after testing, which of course is also after training).

5.2 Using an Imperfect Sensitivity Classifier

In this part of our experiments, we investigate how different approaches perform if the sensitivity classifier results has some misclassification, as would be expected in practice. Obviously, as shown in Figure 4a, we expect lower nDCG scores for those approaches which rely on filtering or penalizing to discourage showing sensitive documents, since some sensitive documents could be relevant.

[4] In this example, we illustrate graded relevance using three levels and graded sensitivity using two levels, but the formalism is easily extended (or collapsed) to any number of gradations along each dimension.

We also note that training a ranking model on nCS-DCG should decrease the resulting nDCG for the same reason, and also because the training measure is used to calculate scores on the validation set, which affect when to stop for some ranking algorithms. As expected, we see a lower nDCG in every case.

Turning now to evaluation using our new evaluation measure (nCS-DCG), we see a quite different picture. As shown in Figure 4b, we get lower scores for baseline 1 as expected, since it doesn't take document sensitivity into account in any way. Additionally, prefiltering and postfiltering no longer have the best scores (as they did when evaluated using nDCG in Figure 3b) because they rely on an imperfect classifier (recall = 0.7, and precision = 0.81). Our proposed approach which takes advantage of sensitivity probabilities fed to the ranking model does well, by contrast, even when those probabilities come from an imperfect classifier.

As the comparison between Figures 3b and 4b shows, nCS-DCG results drop for all approaches as misclassification rates increase. Because the cost of showing a sensitive document is relatively high, false negatives are more important than false positives. Unsurprisingly, it is therefore the false negatives that are responsible for this drop in absolute scores.

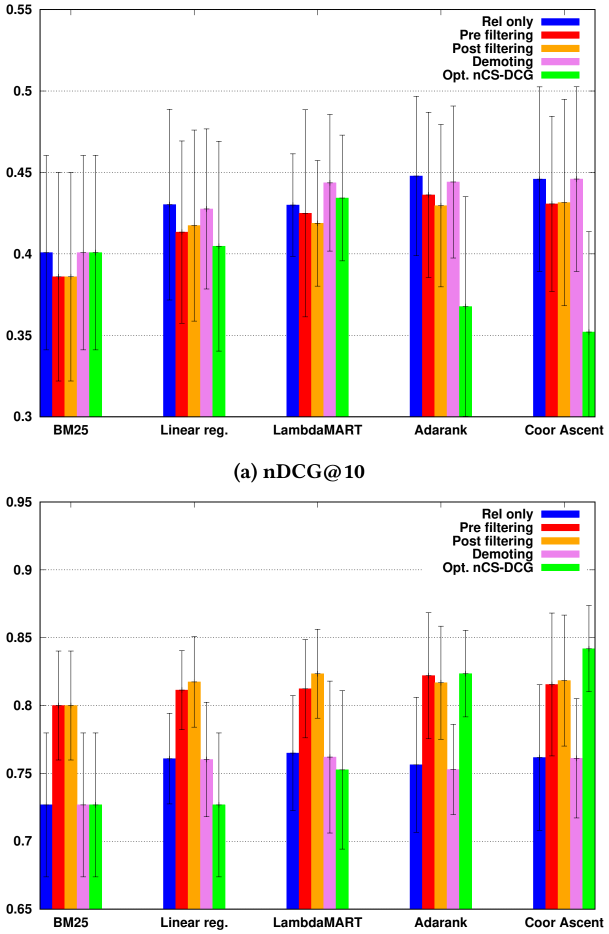
As shown in Figure 4b, listwise approaches trained with nCS-DCG have a higher nCS-DCG scores than other baselines. For AdaRank, our approach is marginally better than the baselines, although none of the differences are statistically significant. On the other hand for Coordinate Ascent, our results are statistically significantly better by a one-tailed paired t-test ($p < 0.05$)

However, for any given level of misclassification error, the key question is which approach is best under those conditions. Here, we see that which approach is best changes as the misclassification rate of the sensitivity classifier changes.

5.3 Cluster-Based Replacement

One simple trick that we can use to limit the risk of false negatives is to replace any potentially sensitive document with a similar document that is less sensitive. This idea is inspired by the usual approach to diversity ranking, in which the documents in a result set are clustered and then the most relevant document(s) from a cluster are selected for display. If instead we choose the least sensitive document(s) from the cluster, we would get a sensitivity-averse analogue to diversity ranking.

We can apply this idea to any ranked list. Given the result set of a query, we first cluster all of the documents that were judged for relevance in order to group together sets of documents for which it might suffice to replace one document in a cluster with another, in the hope that if the first document was relevant, the second one we select will also be relevant. The process starts by stepping through the ranked list, replacing each document with the as-yet unchosen document in the same cluster that has the lowest sensitivity probability (if such a document exists in the cluster). A selected replacement document is then removed from the cluster to avoid selecting it more than once and the process repeats for the next document in the ranked list (which may be from the same cluster, or a different one). The process stops after processing the first k documents in the ranked list. Note that this process has no diversity objective – its sole goal is to minimize the aggregate probability of showing a sensitive document without ever selecting a document that wasn't at least in the same cluster as some document in the ranked list.



(a) nDCG@10

(b) nCS-DCG@10

Figure 4: Ranking performance of different approaches with a sensitivity classifier achieving recall = 0.7 and precision = 0.81. Error bars are the standard deviations of performance across five cross-validation folds

We used CLUTO software for clustering documents,[5] where each document is represented by a word count vector. The clustering program treats each document as a vector in a high-dimensional space, and performs repeated bisectioning until the desired number of clusters is reached. We set the number of clusters to be 20, which was chosen to be higher than our rank cutoff (@10).

As Figure 5 shows, this cluster-based replacement strategy helps to limit the risk of getting a negative CS-DCG score, and it achieves that result for every one of our five approaches. In each case, the (generally) lower line that dips sharply at the right shows the CS-DCG scores achieved by each query, sorted from best to worst. The superiority of tuning AdaRank with a nCS-DCG loss function

[5] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/overview

is clearly indicated in this figure since fewer queries yield poor CS-DCG results. As the (generally) upper blue line shows, it is the hardest queries, those with the lowest CS-DCG, that see the greatest improvement from cluster-based replacement. As Table 4 shows, improvements in nCS-DCG are evident as well for some approaches.

6 DISCUSSION

We showed that when the classifier accuracy is high, filtering has the best performance among other approaches. This is because the classifier has an impact on which documents tare used for training and which of them should be filtered out. In these cases, we also showed that training listwise models based on nCS-DCG is not as good as perfect filtering. On the other hand, when the classifier accuracy is more realistic, our proposed approach yields better results than all of the baselines.

To get a better ideaof what's happening with cluster-based replacement, we tracked the swaps that include at least one sensitive document (either added, removed, or replaced one with another). Such swaps have the largest impact on nCS-DCG. We found that the number of sensitive documents added is less than the number of sensitive documents being removed. That's why there is an improvement when this replacement policy is used in the *Relevance Only* and *Demoting* approaches, as shown in Table 4. In approaches where filtering is used, and in our proposed approach, there are also fewer sensitive documents used for replacement, but nCS-DCG doesn't improve. Two factors contribute to these aggregate results. First, on the positive side, cluster-based replacement causes fewer relevant documents to appear in the result list. As depicted in Figures 5b, 5c, and 5e, the cluster-based replacement successfully reduces the number of sensitive documents for hard queries (those at the far right), and hence yields better CS-DCG scores. But there are relatively few hard queries. For the relatively many easy queries (these with CS-DCG scores above zero), clustered CS-DCG scores are lower than the unclustered CS-DCG scores. Both are near the x-axis, but normalization can amplify these differences. We also note that some approaches (notably our proposed approach in Figure 5e result in fewer hard queries to begin with, and therefore less opportunity for improvement in the arithmetic mean. From this we might conclude that an evaluation measure in which the hard queries receive more emphasis (e.g., a geometric mean) might yield different results.

In our proposed approach, we showed that a listwise learning to rank model can be trained to optimize an objective function derived from the evaluation measure (nCS-DCG). It might be argued that our simple definition of CS-DCG, and its normalized form nCS-DCG, doesn't capture the full complexity of search among sensitive content and that other proposals may be better. For example, sensitivity might be discounted (e.g., in screen-sized units, if the ranked list is so long as to require scrolling), sensitivity might be graded into multiple degrees of sensitivity, or costs and benefits might be combined in some nonlinear way. Our proposed approach is sufficiently flexible to be used with other evaluation measures, at least as long as they satisfy some reasonable constraints. For example, AdaRank requires that the values of the loss function be in the range [-1, 1].

7 CONCLUSION AND FUTURE WORK

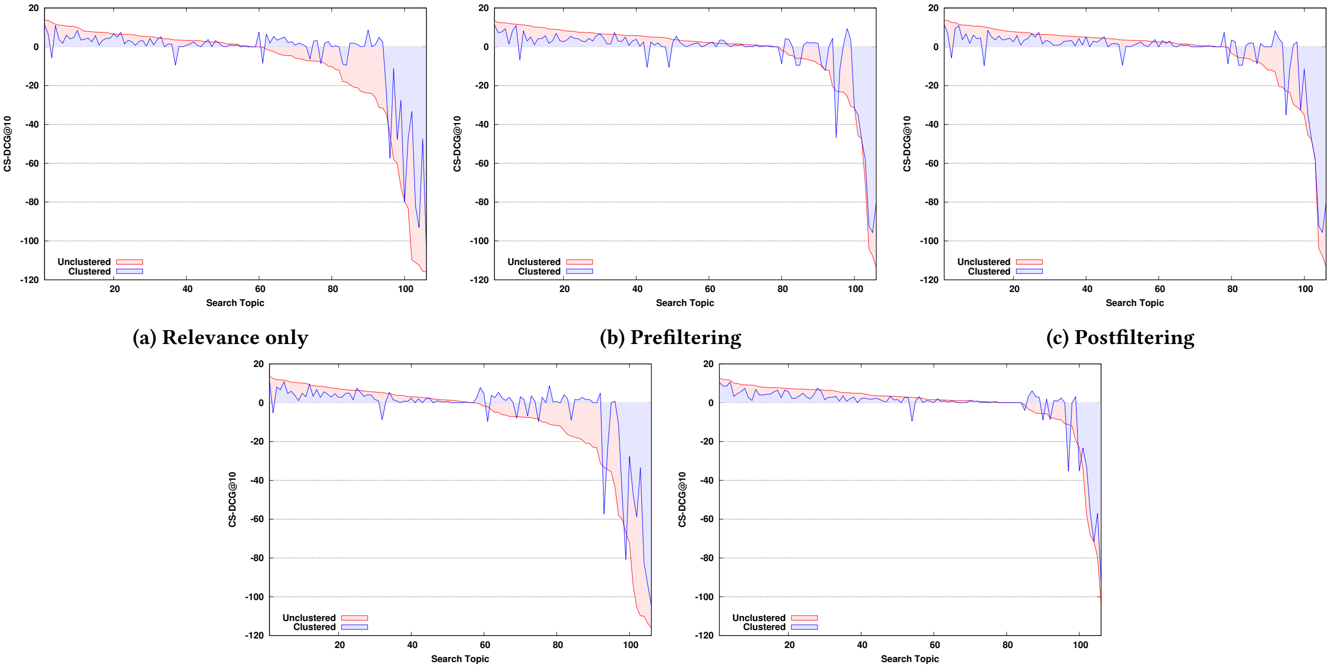
We started out by asking how to construct a ranked retrieval system that knows what not to find. As is often the case, that question led to another more fundamental question: how would we know if we had done so? Answering that led us to propose CS-DCG and nCS-DCG as evaluation measures. Unlike the more familiar nDCG, we need to consider both unnormalized and normalized measures because, unlike DCG, CS-DCG can become negative (and in such cases it would have been better not to search at all!). We then went on to show that at some levels of classifier accuracy that might be seen in practice (e.g., with recall at or below 70%), training a learning to rank model could be better than the more straightforward approach of simply filtering out sensitive documents (either before or after retrieval). We also have been able to show that a simple cluster-based replacement strategy can further improve nCS-DCG, and that such a strategy can reduce the number of queries with negative CS-DCG results.

Of course, much remains to be done. One question that remains open is how best to model the actual cost structure of real tasks. Our present formulation of CS-DCG might be extended in many ways. For example, we might discount sensitivity at deeper ranks (on the grounds that few users would scroll that far), we might model multiple degrees of sensitivity, or we might investigate non-linear cost functions in which a few mistakes might be forgiven but many mistakes might be highly penalized (which, for example, might more faithfully model the situation in e-discovery review). We might also explore smaller rank cutoffs (e.g., @5) as a way of further reducing the risk of showing sensitive documents. Another idea we might explore is modifying our cluster-based replacement strategy to also include some element of diversity ranking. Also, we have seen that the cluster-based replacement works well for hard queries. So we might explore the potential for applying different ranking approaches depending on a query's predicted difficulty. Improvements to our evaluation design are possible as well. For example, we have trained a classifier for the categories we use as surrogates for sensitivity by using very large amounts of training examples; we should of course also experiment with classifiers that are built using more limited training data, as would likely be the case in some practical applications of this technology.

Ultimately we will want to try any and all of these ideas with real content, and content that has real sensitivities, with real users. We see this paper as a first step in that direction, helping to sharpen our questions, and to give us a way of reasoning about what it would mean to do well at such a task.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their insightful comments. This work was funded by National Science Foundation grant IIS-1618695.



(a) Relevance only **(b) Prefiltering** **(c) Postfiltering**

(d) Demoting **(e) Opt. nCS-DCG**

Figure 5: CS-DCG@10 for Adarank with a sensitivity classifier recall of 0.7 with (upper blue line) and without (lower red line) cluster-based replacement.

	Relevance only		Prefiltering		Postfiltering		Demoting		Opt. nCS-DCG	
	unclustered	clustered	unclustered	clustered	unclustered	clustered	unclustered	clustered	unclustered	clustered
BM25	0.727	**0.779***	**0.800**	0.797	**0.800**	0.797	0.727	**0.779***	0.727	**0.779***
Linear regression	0.761	**0.764**	**0.811***	0.785	**0.817***	0.785	0.760	**0.763**	0.727	**0.790***
LambdaMART	0.765	**0.771**	**0.812***	0.788	**0.823***	0.792	0.762	**0.770**	0.753	**0.786***
Adarank	0.756	**0.779**	**0.822***	0.792	**0.817***	0.791	0.753	**0.780**	**0.823***	0.799
Coordinate Ascent	0.762	**0.781**	**0.816***	0.791	**0.818***	0.790	0.761	**0.779**	**0.842***	0.805

Table 4: nCS-DCG@10 comparison between different ranking algorithms under different approaches. For each pair, nCS-DCG@10 is averaged across five cross-validation folds when cluster-based replacement isn't (left) or is (right) used. Symbol (*) denotes statistically significant difference according to a two-tailed paired t-test ($p < 0.05$) over the corresponding clustered/unclustered performance score in the same approach.

REFERENCES

[1] Jiang Bian, Xin Li, Fan Li, Zhaohui Zheng, and Hongyuan Zha. 2010. Ranking Specialization for Web Search: A Divide-and-Conquer Approach by Using Topical RankSVM. In *Proceedings of the 19th international conference on World wide web*. ACM, 131–140.

[2] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. 2005. Learning to Rank using Gradient Descent. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 89–96.

[3] Christopher JC Burges. 2010. From RankNet to LambdaRank to LambdaMART: An Overview. *Learning* 11, 23-581 (2010), 81.

[4] Christopher J Burges, Robert Ragno, and Quoc V Le. 2007. Learning to Rank with Nonsmooth Cost Functions. In *Advances in neural information processing systems*. 193–200.

[5] Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. 2006. Adapting Ranking SVM to Document Retrieval. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 186–193.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th international conference on Machine learning*. ACM, 129–136.

[7] David Cossock and Tong Zhang. 2006. Subset Ranking Using Regression. In *International Conference on Computational Learning Theory*. Springer, 605–619.

[8] Na Dai, Milad Shokouhi, and Brian D Davison. 2011. Learning to Rank for Freshness and Relevance. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*. ACM, 95–104.

[9] Anlei Dong, Yi Chang, Zhaohui Zheng, Gilad Mishne, Jing Bai, Ruiqiang Zhang, Karolina Buchner, Ciya Liao, and Fernando Diaz. 2010. Towards Recency Ranking in Web Search. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*. ACM, 11–20.

[10] Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. 2010. Time is of the Essence: Improving Recency Ranking Using Twitter Data. In *Proceedings of the 19th International Conference on the World Wide Web*. ACM, 331–340.

[11] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. 2003. An Efficient Boosting Algorithm for Combining Preferences. *Journal of machine learning research* 4, Nov (2003), 933–969.

[12] Jerome H Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. *Annals of statistics* (2001), 1189–1232.

[13] Kalervo Järvelin and Jaana Kekäläinen. 2000. IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 41–48.

[14] Thorsten Joachims. 2002. Optimizing Search Engines using Clickthrough Data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 133–142.

[15] Thorsten Joachims. 2005. A Support Vector Method for Multivariate Performance Measures. In *Proceedings of the 22nd international conference on Machine learning*. ACM, 377–384.

[16] Christina Lioma, Jakob Grue Simonsen, and Birger Larsen. 2017. Evaluation Measures for Relevance and Credibility in Ranked Lists. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 91–98.

[17] Tie-Yan Liu et al. 2009. Learning to Rank for Information Retrieval. *Foundations and Trends® in Information Retrieval* 3, 3 (2009), 225–331.

[18] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. 2006. High Accuracy Retrieval with Multiple Nested Ranker. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 437–444.

[19] Donald Metzler and W Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval* 10, 3 (2007), 257–274.

[20] Donald A Metzler, W Bruce Croft, and Andrew McCallum. 2005. *Direct Maximization of Rank-Based Metrics for Information Retrieval*. Technical Report. Citeseer.

[21] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[22] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[23] Krysta M Svore, Maksims N Volkovs, and Christopher JC Burges. 2011. Learning to Rank with Multiple Objective Functions. In *Proceedings of the 20th international conference on World wide web*. ACM, 367–376.

[24] Zhiwen Tang and Grace Hui Yang. 2017. Investigating per Topic Upper Bound for Session Search Evaluation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 185–192.

[25] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. 2008. SoftRank: Optimizing Non-Smooth Rank Metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*. ACM, 77–86.

[26] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13, 3 (2010), 254–270.

[27] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. 2008. Listwise Approach to Learning to Rank - Theory and Algorithm. In *Proceedings of the 25th international conference on Machine learning*. ACM, 1192–1199.

[28] Jun Xu, Yunbo Cao, Hang Li, and Yalou Huang. 2006. Cost-Sensitive Learning of SVM for Ranking. In *European conference on machine learning*. Springer, 833–840.

[29] Jun Xu and Hang Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 391–398.

[30] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 2007. A Support Vector Method for Optimizing Average Precision. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 271–278.

[31] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. 2008. A General Boosting Method and its Application to Learning Ranking Functions for Web Search. In *Advances in neural information processing systems*. 1697–1704.