ABSTRACT

Title of Dissertation: SEARCH AMONG SENSITIVE CONTENT

Mahmoud F. Sayed
Doctor of Philosophy, 2021

Dissertation Directed by: Professor Douglas W. Oard
College of Information Studies
and Department of Computer Science

Current search engines are designed to find what we want. But many collections can not be made available for search engines because they contain sensitive content that needs to be protected. Before release, such content needs to be examined through a sensitivity review process, which can be difficult and time-consuming. To address this challenge, search technology should be capable of providing access to relevant content while protecting sensitive content.

In this dissertation, we present an approach that leverages evaluation-driven information retrieval (IR) techniques. These techniques optimize an objective function that balances the value of finding relevant content with the imperative to protect sensitive content. This requires evaluation measures that balance between relevance and sensitivity. Baselines are introduced for addressing the problem, and a proposed approach that is based on building a listwise learning to rank model is described. The model is trained with a modified loss function to optimize for the evaluation measure. Initial experiments re-purpose a LETOR benchmark dataset, OHSUMED,

by using Medical Subject Heading (MeSH) labels to represent the sensitivity. A second test collection is based on the Avocado Research Email Collection. Search topics were developed as a basis for assessing relevance, and two personas describing the sensitivities of representative (but fictional) content creators were created as a basis for assessing sensitivity. These personas were based on interviews with potential donors of historically significant email collections and with archivists who currently manage access to such collections. Two annotators then created relevance and sensitivity judgments for 65 topics for one or both personas. Experiment results show the efficacy of the learning to rank approach.

The dissertation also includes four extensions to increase the quality of retrieved results with respect to relevance and sensitivity. First, the use of alternative optimization measures is explored. Second, transformer-based rankers are compared with rankers based on hand-crafted features. Third, a cluster-based replacement strategy that can further improve the score of our evaluation measures is introduced. Fourth, a policy that truncates the ranked list according to the query's expected difficulty is investigated. Results show improvements in each case.

SEARCH AMONG SENSITIVE CONTENT

by

Mahmoud F. Sayed

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2021

Advisory Committee:
Dr. Douglas W. Oard, Chair/Advisor
Dr. Ashok Agrawala
Dr. Katie Shilton
Dr. Jordan Boyd-Graber
Dr. Marine Carpuat

Dedication

To mom and late dad,

Rana, Yahia,

Mohamed, Ashraf, and Ahmed

Acknowledgments

I owe my gratitude to all the people who have made this thesis possible, and because of whom my graduate experience has been one that I will cherish forever.

First and foremost I'd like to thank my advisor, Professor Douglas William Oard for giving me the invaluable opportunity to be his student over the past few years. I will forever appreciate his time and effort he invested in me, always having open doors for help and guidance, especially during my lowest moments. He kept me pushing towards developing my knowledge and working on interesting projects, while always encouraging healthy work-life balance. It was an absolute honor to be mentored by such an extraordinary individual.

I would also like to thank Professor Katie Shilton. Without her motivating ideas and social science expertise, this thesis would have been a distant dream. Thanks are due to Professor Jason Baron for the fruitful collaboration. Thanks are also due to Professor Ashok Agrawala, Professor Jordan Boyd-Graber, and Professor Marine Carpuat for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

I've been fortunate to collaborate with both Modassir Iqbal and Caitlin Christian-Lamb. They did an amazing work in building the test collections. During that work, I had the chance to mentor 4 undergraduate students: William Cox, Jonah Lynn

Rivera, Benjamin Mao, and Nishanth Mallekav. I would like to acknowledge their amazing work in different pieces of this work.

For my parents and brothers, I owe the most gratitude for their infinite encouragement and prayers. They were always my pillars in life and the reason why I am here at this point. In addition, I would like to acknowledge my wife and partner, Rana. Your continued support, love and patience is lighting up my life and made this dream possible. Thank you for standing by me through ease and hardship.

I take this as an opportunity to mention my colleagues in the UMD IR group, and our extended family at the CLIP lab, for enriching my graduate life and building an environment full of fun and collaborations. Suraj Nair helped me by sharing his wide knowledge in neural models for information retrieval. I've been fortunate to sit in front of Petra Galuscakova who had several initiatives that benefited everyone. My interaction with Ahmed Elgohary, Amr Sharaf, Rashmi Sankepally, Joe Barrow, Han-Chin Shing, and Elena Zotkina molded many of my thoughts, in research and life alike.

I would also like to acknowledge the help and support from some of the staff members. Petra Zapf, Tom Hurst, Jennifer Story, and Fatima Bangura always did their best to assist and coordinate logistics throughout the years. Without their coordination, my graduate life would have been a mess.

Finally, I would like to acknowledge financial support from the National Science Foundation (NSF) for all the projects discussed herein.

It is impossible to mention all, and I apologize to those I've inadvertently left out. Lastly, thank you all and thank God!

Table of Contents

List of Tables

List of Figures

List of Abbreviations

BERT	Bidirectional Encoder Representations from Transformers
ESI	Electronically Stored Information
ERR	Expected Reciprocal Rank
FOIA	Freedom of Information Act
IoT	Internet of Things
IR	Information Retrieval
κ	Cohen's kappa coefficient
LR	Logistic Regression
LtR	Learning to Rank
MAP	Mean Average Precision
MeSH	Medical Subject Headings
MRR	Mean Reciprocal Rank
nDCG	Normalized Discounted Cumulative Gain
nCS-DCG	Normalized Cost Sensitive Discounted Cumulative Gain
S&P	Search and Protection
SVM	Support Vector Machine
TAR	Technology Assisted Review
TASR	Technology Assisted Sensitivity Review
TF-IDF	Term Frequency-Inverse Document Frequency
TREC	Text Retrieval Conference

Chapter 1: Introduction

Currently, we live in the era of big data where people generate a huge amount of digital content through different sources, e.g., mobile devices, tablets, laptops, and IoT (Internet of Things) devices. Generated content could belong to a wide range of topics, e.g., telecommunications, media, health, and finance. From a data analytics perspective, digital content is a great source from which to draw new insights. However, a major problem with this kind of content is that it is usually intermixed with sensitive information such as personal information and private conversations. As a result, sensitive information should be removed before making content available for search.

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.

From the content provider perspective, they need to be careful to exclude sensitive content that should not be found from the content being searched. As content volumes increase, this segregation of sensitive content becomes more expensive. One obvious approach is to ask content producers to mark sensitive content when it is first produced, but that approach suffers from at least two major problems. First, the content producer's interests may differ from the interests of future searchers, so content producers may not be incentivized to label sensitivity in ways that would facilitate future access to content that is not actually sensitive. As a simple example of this, some lawyers add a note at the bottom of every email message that they send indicating that the message may contain privileged content. Doing so can serve the lawyer's general interest in protecting privileged content, but there is no incentive for the lawyer to actually decide in each case whether such a note should be added to a specific message. Second, sensitivity can change over time, so something marked as sensitive today may no longer be sensitive a decade from now.

From the searcher's perspective, some searchers do not actually want to find everything that is relevant to their query, preferring instead that some content (e.g., content depicting violence) be filtered out. One example is Google's SafeSearch feature, which seeks to hide results containing sexually explicit content.

The problem of deciding what information can be shown in response to a request arises in many settings. It can be seen in the case of two candidates running for president: Hillary Clinton and Jeb Bush. Hillary Clinton had been using a private email server for official public communications rather than using official State

Figure 1.1: A tweet by Hillary Clinton asking the state department to release all of her work-related emails.

Department email accounts maintained on secure federal servers. While running for president, she sent 30,490 work-related email messages to the State Department and asked that they be reviewed and released as quickly as possible, as shown in Figure 1.1. These emails were being handled by 20 permanent, and 30 part-time, workers. Due to the vast backlog of Freedom of Information Act (FOIA) requests for Clinton's emails, the State Department had to add more staff members to the review team for assistance. Nearly a year later, the review team was able to finish the review process. Jeb Bush chose a different approach, releasing all the approximately 280,000 email messages from his time as Governor of Florida. These email messages were 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.

Another example is e-Discovery, which involves identifying, collecting, reviewing, and producing electronically stored information (ESI) that can be used as evidence in a civil legal case. All types of content can serve as evidence, including text, images, emails, and audio files. Figure 1.2 shows the process workflow of e-

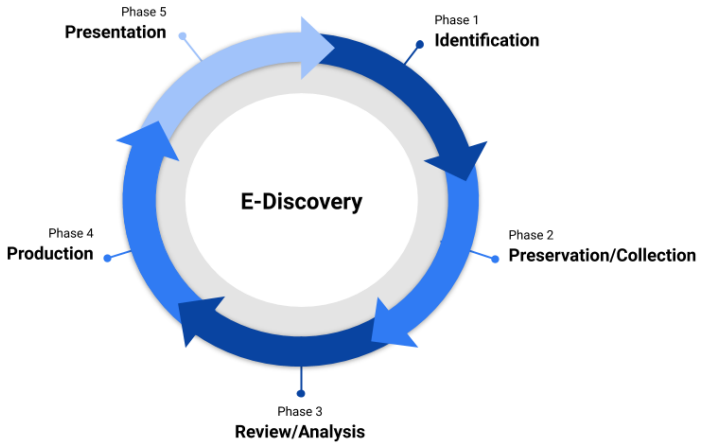


Figure 1.2: Phases of an e-Discovery process.

Discovery. In a typical e-Discovery process, review represents the most expensive phase, estimated to be around 80% for the average case [9]. In this phase, the review team, comprised of attorneys and legal professionals, assess ESI for relevance ("responsiveness" in e-Discovery). Additionally, the review team has to identify privileged documents which should be protected and can not be used for the legal case even if they contain relevant information. Privileged documents are protected under attorney/client privilege, which keeps any communication between a client and his attorney confidential among other privileges. Typically, documents are first assessed for relevance, and then relevant documents are filtered out if they are privileged.

From the previous examples, we can observe that the review process is time-consuming. For example, an e-Discovery process may take several months. Additionally, the review process requires money to pay lawyers for their reviewing effort. To reduce the time and cost of the review process, Technology Assisted Review

Figure 1.3: Workflow for search and protection engines.

(TAR) seeks to have computer software electronically classify documents based on input from expert reviewers. The main objective of TAR is to find as nearly all the relevant documents in a collection as possible, with reasonable effort. This can be achieved by reducing effort spent in reviewing non-relevant documents. In a similar sense, for some other types of sensitivity review, Technology Assisted Sensitivity Review (TASR) identifies sensitive documents which should not be released to the public. Both TAR and TASR can significantly reduce the time and cost of review.

Although technology helps reviewers reduce time spent judging for relevance and privilege, neither review process is fast enough that the requesting party can access content through interactive search. In this dissertation, we propose to create *Search and Protection* (S&P) engines that are designed to accomplish that task as depicted in Figure 1.3. We adopt the term S&P engines to refer inclusively to all such systems, regardless of who it is (the content provider, the user, or a third party) that wishes the sensitive content to be protected.

The central question of this dissertation asks how we can build the search technology to effectively search among sensitive content. The principal properties of search and protection engines are listed below.

1. They protect sensitive information from disclosure.

2. They retrieve relevant documents. Of course, it is expected to have some information loss as a result of hiding sensitive information that could be relevant [81].

3. They are fast, in order to provide efficient access to information through interactive search. As a result, searchers will be able to learn to create better queries for their information needs.

4. They are more affordable than exhaustive manual review.

To address the first two properties, we need to look at how search engines are built. They are built by first quantifying what we wish them to do (i.e., designing the evaluation measure that will characterize their effectiveness) and then by automatically tailoring search algorithms to produce the best possible results according to that evaluation measure. This process, generically referred to as Learning to Rank (LtR), produces better results than any other known technique. We therefore begin by designing a class of evaluation measures that balance relevance (the ability to find what we want) with sensitivity (the ability to identify that which needs protecting). We then instantiate evaluation measures for the type of tasks we are interested in, and develop new machine learning techniques to maximize the ability of a search engine to optimize those specific measures. To address the last two properties, we propose to use automatic sensitivity classification, which is similar to TASR. We adapt TASR models to different domains, because our test collections are different from the government records for which TASR was developed.

The first two goals of an S&P engine are increasing the quantity of relevant content shown while decreasing the quantity of sensitive content shown. These two goals are conflicting in that optimizing only towards relevance risks showing sensitive information, and optimizing only towards sensitivity only may suffer from information loss.

1.1 Thesis Statement

Search and protection engines can provide a useful degree of access to unsegregated collections, while providing a useful degree of protection for sensitive content.

1.2 Questions

We support the thesis statement, stated above, by addressing the following broad questions:

Q1. How to build a search and protection engine? What are the main components that assemble such an engine?

Q2. How do we evaluate the performance of search and protection engines?

Q3. What are the desired properties of evaluation measures for search among sensitive content?

Q4. How well can a sensitivity classifier detect sensitive information?

Q5. How well can we learn search cutoffs to handle search topics of different sensitivity distributions?

We address the first question by exploring a new class of ranking approaches designed to effectively search among secrets. We inferred the common components that assemble these ranking approaches (Q1). Our goal is to balance the user's interest in finding relevant content with the provider's interest in protecting sensitive content (Q3). To answer Q2, we need access to sensitive content, which is challenging. So we create a novel test collection that has realistic sensitive content coupled with relevance information with respect to some information needs. The new test collection helps us build a sensitivity classifier that can effectively classify sensitive information (Q4). Then finally, we develop two search cutoff policies as we noticed that search topics are of different difficulties based on the number of sensitive documents that can appear in the result (Q5).

1.3 Contributions

This dissertation introduces a new retrieval task that does not just aim at retrieving relevant content, but it aims at protecting sensitive content even if it is relevant. The contributions of this work can be divided into three main categories: System contributions (S), Test collection contributions (T), and Evaluation measure contributions (E).

1.3.1 Evaluation Measure Contributions

E1 We propose a set of potentially desirable properties for evaluation measures that balance for relevance and sensitivity. We express these as axioms, drawing inspiration from prior axiomatic analysis of relevance measures and of measures that balance relevance with other desiderata (e.g., credibility [67]) (Section 4.5).

E2 We propose the TERN and SENS evaluation measures that support human interpretation of the relevance and sensitivity contributions for a given query (Sections 4.1 and 4.2).

E3 We introduce Cost Sensitive Discounted Cumulative Gain (CS-DCG), a variant of the Discounted Cumulative Gain (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 compare different search systems across several queries (Section 4.3).

E4 We propose a variant of CS-DCG that penalizes each sensitive document deferentially, called γCS-DCG, with the largest penalties for the sensitive documents most highly ranked. Normalizing γCS-DCG produces nγCS-DCG, which we empirically show it can be used for training ranking models that outperform other models trained using TERN or SENS measures (Section 4.4).

E5 We determine which of the properties in **E1** are satisfied by each of the evaluation measures we propose (Section 4.6).

1.3.2 Test Collection Contributions

T1 We build two test collection built from Avocado email collection [87] that are now annotated for relevance and sensitivity [56,103]. We plan to submit these test collections to the Linguistic Data Consortium (Sections 3.2, 3.3, and 3.4).

T2 We repurposed the LETOR OHSUMED test collection for our retrieval task by selecting two topical categories as surrogates for sensitivity (Section 3.1).

1.3.3 System Contributions

S1 We propose three baselines that are based on filtering out sensitive content (either before or after retrieval) or downgrading relevance labels for sensitive documents (Section 5.3.1).

S2 We propose a novel usage of listwise learning to rank algorithms to optimize for new evaluation measure, from Chapter 4, by modifying the loss function (Section 5.3.2).

S3 We evaluate the effectiveness of our proposed LtR approach along with several baselines using the OHSUMED and Avocado test collections (Section 5.4).

S4 We build a learning model to predict the sensitivity of emails (Section 5.1.2).

S5 We apply LtR, combining signals from both traditional features and the output of neural models, to balance relevance and sensitivity objectives. Results show increased effectiveness over traditional LtR techniques by using transformer-based features instead of hand-crafted features (Section 6.2).

S6 We propose a cluster-based replacement strategy, and show that such a strategy can sometimes substantially reduce the number of queries that have at least one sensitive result among the top ranks (Section 6.3).

S7 We compare two rank cutoff policies to improve results over that can be achieved with a fixed rank cutoff. These policies aim at limiting the risk of showing sensitive results by displaying a few results for topics that are expected to have a high number of sensitive documents (Section 6.4.2).

S8 We study the effect of sensitivity classifier effectiveness on search and protection engines. Specially, we seek to explore whether optimizing for an intrinsic measure (e.g., F_1 or F_2), also results in improvements when evaluated using an extrinsic measure (Section 5.5).

1.4 Dissertation Outline

The remainder of this dissertation is structured as follows. We survey related work in Chapter 2. In Chapter 3, we describe the test collections and their creation process. In Chapter 4, we study different measures that can be used for optimizing and evaluating search and protection engines based on a set of desired properties for

evaluation measures that balance between relevance and sensitivity. In Chapter 5, we describe our modeling of relevance and sensitivity, and different approaches to using those models for search among sensitive content. Then in Chapter 6, we present extensions to improve the quality of retrieved results with respect to our evaluation measures. We conclude in Chapter 7, scoping our contributions in the context of experimental limitations, looking to future directions, and articulating some broader impacts of our work.

Chapter 2: Background

This chapter outlines different research work that our thesis dissertation relies on. Examples include evaluation measures for information retrieval systems, learning to rank, sensitivity classification, etc. Also, since our objective is to balance between relevance and sensitivity, we describe some related work in the scope of ranking with multiple objectives.

2.1 Evaluation Measures for Information Retrieval Systems

The aim of evaluation is to measure how well a system achieves its intended goal. In the scope of information retrieval, systems are evaluated by measuring how well results satisfy a user's information need, e.g. finding what the user wants. In an abstract way, an information retrieval system can be seen purely as a classifier, which classifies each document in the search result list as relevant or not. Classification evaluation measures, such as precision or recall, can be used to assess the effectiveness of a retrieval system. Precision is the proportion of retrieved documents that are relevant, and recall is the proportion of documents that are relevant that are retrieved.

Deciding which evaluation measure to use is a crucial task, because the eval-

uation measure should correlate with user's satisfaction criteria and discriminate between different retrieval results. In this section, we discuss different evaluation measures for two different environments: 1) Binary vs or graded relevance assessments, and 2) Complete vs incomplete relevance judgments.

First, measures based on binary relevance are numerous and widely used. Examples include Precision@k (P@k) [52], R-precision (R-prec) [54], Mean Average Precision (MAP) [114], and Mean Reciprocal Rank (MRR) [115]. Mean average precision is one of the most commonly used measures of retrieval effectiveness. It is known to be a stable [15] and informative measure [7]. On the other hand, relevance assessments can have more than 2 levels (e.g., highly relevant, somewhat relevant, and non-relevant). Evaluation measures that use graded relevance include Normalized Discounted Cumulative Gain (nDCG) [61], and Expected Reciprocal Rank (ERR) [23].

Second, all previous measures assume that relevance judgments are complete, i.e. each document is judged whether it is relevant or not with respect to all search topics. This requirement is hard to meet in practice and is often approximated by assuming all documents that were not retrieved as non-relevant. However, with the rapid growth of content and with improved search technology making it possible to find new content, some relevant documents may be missed during the retrieval process of building test collections, and hence the relevance judgments become less useful as a basis for evaluation [6, 14, 86]. It is mandatory to address this setup environment as in Binary Preference (bPref) which how frequently relevant documents are retrieved before non-relevant documents [14]. When comparing systems over

test collections with complete judgments, MAP and bpref are reported to be equivalent. With incomplete judgments, bpref is shown to be more stable. Measures, e.g. induced AP, subcollection AP, and inferred AP have been shown robustness to random deletions of relevance judgments [125] although missing relevance judgements in practice are not random.

Axiomatic analysis has been used to understand the properties of evaluation measures for specific information retrieval tasks. The analysis starts by defining a set of potentially desirable properties, then evaluation measures under test are checked in terms of whether they satisfy these properties or not [3, 42, 78]. Amigó et al. [4] proposed a set of ten axioms relating to relevance and diversity. Lioma et al. [67] proposed 8 potentially desirable properties for evaluation measures that balance relevance and credibility, and proposed different ways for combining standard relevance measures and credibility measures. In this work, we propose a set of desired properties for search among sensitive content. Then, we check which properties are captured by our proposed evaluation measures for relevance and sensitivity.

Another way to study evaluation measures is to use statistical tests such as Sakai's intuitiveness test [101] and Amigo et al.'s Metric Unanimity [4]. Such tests quantitatively characterize the degree to which aggregate measures capture specific properties that are well captured by a suite of simpler measures. As we presently lack suitable test collections for search and protection engines, this approach is not yet possible for our application.

2.2 Multi-Criteria Evaluation Measures

In Section 2.1, we describe some work done in developing evaluation measures based only on relevance. In this section, we discuss some of the evaluation measures that incorporate additional objectives.

Relevance labels from different sources. Svore et al. [109] introduce the notion of a graded measure, which is composed of multiple IR measures, each of which maps a given ordering of documents for a given query to a score, e.g. nDCG as a top-tier measure and relevance measure derived from click data as a second-tier measure. This is an important when search engines needed to be optimized for several evaluation measures simultaneously. The authors extend LambdaMART to optimize those graded measures. Results show the second-tier measure is significantly improved, while leaving the top-tier measure largely unchanged.

Credibility vs Relevance. Lioma et al. [67] present two types of evaluation measures that are designed to measure the effectiveness of both relevance and credibility in ranked lists of retrieval results. Type I measures define different ways of measuring the effectiveness of both relevance and credibility based on differences in the rank position of the retrieved documents with respect to their ideal rank position (when ranked only by relevance or credibility). Unlike Type I, Type II measures operate directly on document scores of relevance and credibility, instead of rank positions. In our work, balancing between sensitivity and relevance, we found out that our proposed evaluation measure is similar to the Type II measure "Normalized Weighted Cumulative Score" (NWCS), except that we do not discount the cost of

sensitive documents.

Fairness vs Relevance. Fairness of search has attracted recent attention. Das et al. [35] propose an evaluation measure that is based on distributional fairness. Distributional fairness is defined as the difference between ranked result distribution compared with target distribution (either uniform or a priori known distribution) using KL divergence.[1] To integrate between relevance and fairness, there could be different ways, e.g. arithmetic mean, geometric mean, or harmonic mean.

In our work, we are interested to balance between relevance and a new criterion, which is sensitivity. We propose a class of measures that incorporate gains for showing relevant documents and costs of showing sensitive results.

2.3 Learning to Rank

Learning to rank is an extensively studied research field, and there are many optimization algorithms that are flexible and are able to learn from large amount of training data. LtR approaches can be broadly divided into 3 groups [65, 68]: 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. Therefore, the group structure of ranking is ignored in this approach. Each sample of training data contains features of the query-document pair, and the label represents the relevance score for that pair. The loss function in learning is pointwise in the sense that it is

[1] `https://en.wikipedia.org/wiki/Kullback\OT1\textendashLeibler_divergence`

defined on a single object (feature vector). For ranking documents, assuming the learned model outputs real numbers, we can use the model to rank documents (i.e., sort documents according to the scores given by the model) for a given query. In our work, we used linear regression [27] with a squared loss function as an example from a pointwise approach.

In the pairwise approach, the ranking is transformed into a binary classification to decide which document is more relevant given a pair of documents. The preference pairs can be viewed as instances and labels in a new classification problem. Each sample of training data contains features for two documents, and the label represents the relative order between that pair of documents. In the pairwise approach, the group structure of ranking is also ignored. The loss function in learning is pairwise because it is defined on a pair of feature vectors (documents). For ranking documents, assuming the learned model outputs real numbers, we can use the model to rank documents (i.e., sort documents according to the scores given by the model) for a given query. The pairwise approach includes Ranking SVM [62,70], RankBoost [45], RankNet [16], GBRank [128], IR SVM [21], Lambda Rank [17], and LambdaMART [121]. In our work, we use LambdaMART [18,121] 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 natural way. Specifically, it takes ranked lists as instances in 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. Each sample of train-

ing data contains the labeled documents for a given query. For ranking documents, assuming the learned model outputs real numbers, we can use the model to rank documents (i.e., sort documents according to the scores given by the model) for a given query. The listwise approach includes ListNet [22], ListMLE [122], AdaRank [124], SVM MAP [127], Coordinate Ascent [75], Soft Rank [111], and others [60,63,76]. In our work, we use AdaRank and Coordinate Ascent as two examples from that category. We selected more than one approach because we want to show that training a listwise approach with a modified loss function works well with different ranking algorithms.

With the hype of adapting BERT models to text ranking, it can be seen that the resulting model belongs to the pointwise category as each document is treated independently as in monoBERT [83]. Same authors developed a pairwise version called "duoBERT" which takes a query and two documents formulated in one input sequence, and then it predicts which document should come first [85]. Han et al. [53] build a LtR model, using TF-Ranking [94], on top of BERT representations to the [CLS] token. The LtR model can be optimized towards minimizing pointwise, pairwise, and listwise losses.

2.4 Multi-Criteria Ranking

Multi-criteria learning to rank is becoming more prevalent because search results are not just evaluated on relevance, but on other aspects (e.g., freshness, novelty, diversity, fairness, and cost [93, 110, 120]).

Freshness vs Relevance. Dai et al. [32, 33] show that different query types require different ranking optimization, e.g. temporal queries such as breaking news, relevance and freshness are highly correlated. Therefore, a ranker optimized for returning fresh documents may produce satisfactory results. However, for queries that are not usually time-sensitive, paying too much attention to freshness may significantly hurt ranking effectiveness in terms of relevance. As a result, Dai et al. [32] develop a different ranker for each type of queries. This approach is known as "divide and conquer" (DAC) [11]. In DAC, queries are clustered based on their feature representations, and separate rankers are trained with each for one cluster simultaneously. At test time, the query is compared against the generated cluster centroids and is ranked under all rankers with the weights depending on query-cluster similarity values. Results show that DAC outperform other approaches relying on building one ranking model.

Dong et al. [39, 40] propose a retrieval system which automatically detects and responds to recency sensitive queries. The system detects recency sensitive queries using a high precision classifier. The system adjusts the relevance labels based on the freshness of the web page. So for stale pages, they demote the relevance label by one grade (e.g., from highly relevant to somewhat relevant). We use the same approach in one of our proposed baselines, by demoting the relevance labels for sensitive documents. But unlike this work, we have a balanced amount of training for both relevance and sensitivity.

Efficiency vs Effectiveness. Wang et al. [116] propose a framework for automatically learning ranking functions that optimize the trade off between efficiency

and effectiveness. They focus on a class of linear feature-based ranking functions, e.g. linear regression. In this setup, dropping weights of some features improves the ranking efficiency. Results show a linear ranking function optimized in this manner is capable of balancing between retrieval effectiveness and efficiency, as desired. Learned ranking functions achieved similar strong effectiveness as a state-of-the-art learning to rank model, but with significantly decreased average query execution times; the variance of query execution times was reduced as well, which makes them desirable from a practical point of view.

To the best of our knowledge, this is the first attempt to study how to build search engines so that they find what we want while protecting sensitive content. In the scope of relevance and sensitivity, they are competing each other, i.e. optimizing for relevance only may disclose sensitive content and paying too much attention to sensitivity may hide some relevant content.

2.5 Email Classification

There has been work on classifying emails for different tasks [79]. An automatic email classifier is a system that automatically classifies emails into one or more of a discrete set of predefined categories. Examples include folder categorization [10], spam filtering [126], separating private from official email [2, 58, 118], detecting gossip [77], complaint email classification [28], inquiry email classification [57], extracting email threads [59], and recipient recommendation [50].

The problem of email classification is different from the standard problem

of text classification due to the nature of email. An email message consists of structured fields, e.g. from and to fields, and unstructured fields, e.g. subject and body [10]. Additionally, messages have relationship data, e.g. which emails belong to the same thread [117]. As a result, different set of features can be used to successfully discriminate email messages according to the task [31, 100].

For research use, there are two large public real email datasets. First, the Enron corpus contains over 600,000 emails generated by 158 employees collected by the Federal Energy Regulatory Commission (FERC) during its investigation of the company [64]. Second, the Avocado Research Email Collection consists of emails and attachments taken from 279 accounts of a defunct information technology company referred to as "Avocado" [87]. In our work, we focus on the Avocado email collection as its license better protects sensitive content, and it did not go under several redactions which could possibly remove substantial information that could help in our analysis.

Alkhereyf [2] et al. present an empirical study on email classification into two categories: Business and Personal. The authors use lexical features as well as social network features extracted from the email exchange network of both Enron and Avocado. Their experiments show that when the social network features combined with lexical features outperforms the lexical features alone. In this dissertation, we study a different classification task which is predicting email sensitivity that is defined based on a persona description. The specification of sensitive content does not necessarily relate to whether an email is business or personal.

2.6 Technology Assisted (Sensitivity) Review

The Freedom of Information Act (FOIA) generally provides that any person has the right to request access to federal agency records or information, except to the extent the records are protected from disclosure by specific exemptions [44]. Before release to the public, documents must therefore first be manually reviewed to identify and protect any sensitive information [8]. Exhaustive review is becoming infeasible, however, given the huge amounts of digital information we generate.

Automatic classification for sensitivity is thus an important part of the review process that is designed to cope with the rapid growth of information [1, 112]. As in e-discovery [89, 90, 113], technology-assisted review is a combination of input from expert human reviewers and computer software to partially automate the classification of records.

The objective of technology-assisted review (TAR) in electronic discovery (e-discovery) is to find as nearly all the relevant documents in a collection with reasonable effort [88,91]. Grossman et al. [51] show that TAR can be more efficient than an exhaustive manual review to identify the responsive documents. Cormack et al. [25] provide an overview of machine learning approaches for TAR in the context of e-discovery. Each of the approaches, first, identifies an initial seed set of documents that are then used to train a document classifier to identify the k documents to be reviewed. The reviewed documents are then used to re-train the classifier, and the process continues in an iterative cycle until a decision is made that close to all the relevant documents have been discovered. Gain is the number of relevant documents

presented to the human during training and review, while cost is the total number of relevant and non-relevant documents presented to the human during training and review.

In review for relevance, there is a request that represents the information need, and the target is to find all documents relevant to that need. However, in sensitivity review, there is no equivalent to the request for production (i.e., there is no text description of the sensitivities in the collection) that can be used as a query to generate an initial pool of documents for training a classifier. In theory, it is possible to manually construct queries with keywords that a reviewer might expect to be related to a specific sensitivity. However, in practice, due to the range of potential sensitivities, manually constructing separate queries for each type of sensitivity could result in an unmanageably large number of result sets. Moreover, identifying sensitivity by manually generating queries is limited to searching for the sensitivities that a reviewer expects to be in a collection. However, since the actual sensitivities are unknown, this approach is likely to result in low recall of sensitive information.

McDonald et al. [73] work on automatically classifying 2 FOIA exemptions: 1) International Relations, and 2) Personal Information using text classification [108]. Improvements had been found by incorporating semantic features using word embeddings to text classification [71]. Overall, when the sensitivity classifier is more accurate, this will result in less number of documents a reviewer would need to review.

In our work, we have learned a lot from best practices in sensitivity review.

We get useful insights from McDonald et al. [73] for learning how to build a sensitivity classifier, especially in the context of personal information. One difference between our work and theirs is how the sensitivity classifier fits in the whole system. Previously, sensitivity review is used in e-discovery to help in filtering out privileged documents. So in that case, the objective is to find almost all privileged documents in a collection. In our work, a sensitivity classifier is used along with a ranking model to answer queries and produce a ranked list of documents of a certain length (e.g., 10) for each of submitted queries. This difference results in the process of training the sensitivity classifier through active learning. In our work, if deployed online (Section 7.2.3), submitted queries can guide which documents that need human review and then added to train the sensitivity classifier. In this way, search and protection engines are capable of knowing what searchers are interested in and adapt with new sensitivities in the collection. On the other hand, in previous work, active learning relies on labeling documents that the classifier is most uncertain about or those getting high scores of sensitivity.

The next chapter describes the test collections we used for our experiments.

Chapter 3: Test Collections[1]

Test collections enable controlled experiments to characterize retrieval effectiveness. Typically, a test collection is thought of as having 3 components: documents, topics (i.e., information need statements), and relevance judgements (which record the degree of relevance of some document to some topic). Our work adds a fourth component: annotations that indicate not just which relevant documents should be found, but also which documents should be protected because they contain content that someone—the content provider or the searcher—would consider inappropriate to show. We refer to what should not be shown—even if relevant—as *sensitive*. Thus, in addition to relevance judgments, we also need sensitivity judgments.

In different applications, e.g. e-discovery [51], TREC Legal Track developed similar test collections, where documents are judged for responsiveness (i.e., being relevant to the legal and factual issues that are the focus of the litigation), and privileged (i.e., withheld from production because of attorney-client privilege or work-product protection) [26]. Unfortunately, there were different topics used for identifying relevant documents and privileged ones. As a result, documents for each

[1]Some parts of this chapter were taken from two publications by Sayed, Iqbal, Cox, Rivera, Christian-Lamb, Oard, and Shilton [56, 103]

topic are retrieved from different samples of the whole dataset. As a result, we can not use such test collection, because not the same documents were assessed for relevance and privilege.

It was a challenge for us to find a public test collection that is annotated for both relevance and sensitivity. This challenge motivated us to choose 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 (Section 3.1). Also, we describe the process of creating a test collection out of Avocado Email Collection which resulted into two sub-test collections (Section 3.2). Apart from the description of our test collections, we describe the relevance features extracted for query-document pairs for building different ranking models (Section 3.3). Also, we describe the sensitivity features extracted from documents for building different sensitivity models (Section 3.4).

3.1 Topical Sensitivity: OHSUMED

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 work on a collection of medical documents. We chose to use one of the LETOR benchmark datasets that has these characteristics, OHSUMED [97] which is used in many research studies [21, 123, 124]. 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,[2] author, source, and publication type. The collection is distributed by the National Library of Medicine and is restricted to the data not being used for any non-experimental clinical, library, or other setting, and any human users of the data will be explicitly told that the data is incomplete and out-of-date.

3.1.1 Relevance Judgments

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 highly, moderately, 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 this 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 query as not relevant. Because some documents were judged for relevance to more than one query, there are a total of 14,430 unique documents for which relevance to one or more queries has been judged. Table 3.1 shows the numbers of relevant documents (to any degree) and those documents that are not relevant.

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

	Highly relevant	Moderately relevant	Not relevant
Sum	2,252	2,585	11,303
Avg.	22.30	24.61	106.63
Std.	23.15	20.24	46.54
Median	14	18	106

Table 3.1: Relevance statistics about judged OHSUMED documents.

3.1.2 Sensitivity Judgments

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 represent the sensitive content to be protected. 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. Table 3.2 shows the number of sensitive documents and some of their statistics over search topics.

	Sensitive	Not sensitive
Sum	1,961	14,179
Avg.	19.81	133.76
Std.	29.35	57.16
Median	8	132

Table 3.2: Sensitivity statistics about judged OHSUMED documents.

Statistic	OHSUMED
Mean Relevant	29.37%
Median Relevant	25.51%
Mean Sensitive	12.0%
Median Sensitive	5.22%

Table 3.3: Per query relevance and sensitivity statistics.

3.1.3 Empirical Analysis

As Figure 3.1 shows, some queries have many relevant documents that are sensitive, whereas others do not. This means that the challenges posed by sensitivity are topic-dependent, which also reflects what we would expect to see in real applications. As will be explained later in Section 6.4.2, this observation suggests that queries with different sensitivity distributions should be handled differently by varying the search cutoffs.

In a typical case, a topic with no relevant documents should be removed from the test collection, as it can not be used to compare different retrieval systems. In our case, however, topics containing at least one relevant *or sensitive* document are useful, since a search and protection engine should protect sensitive documents even when no relevant documents exist. Every topic in our test collections has at least one relevant or sensitive document. Table 3.3 shows relevance and sensitivity statistics for the OHSUMED test collection.

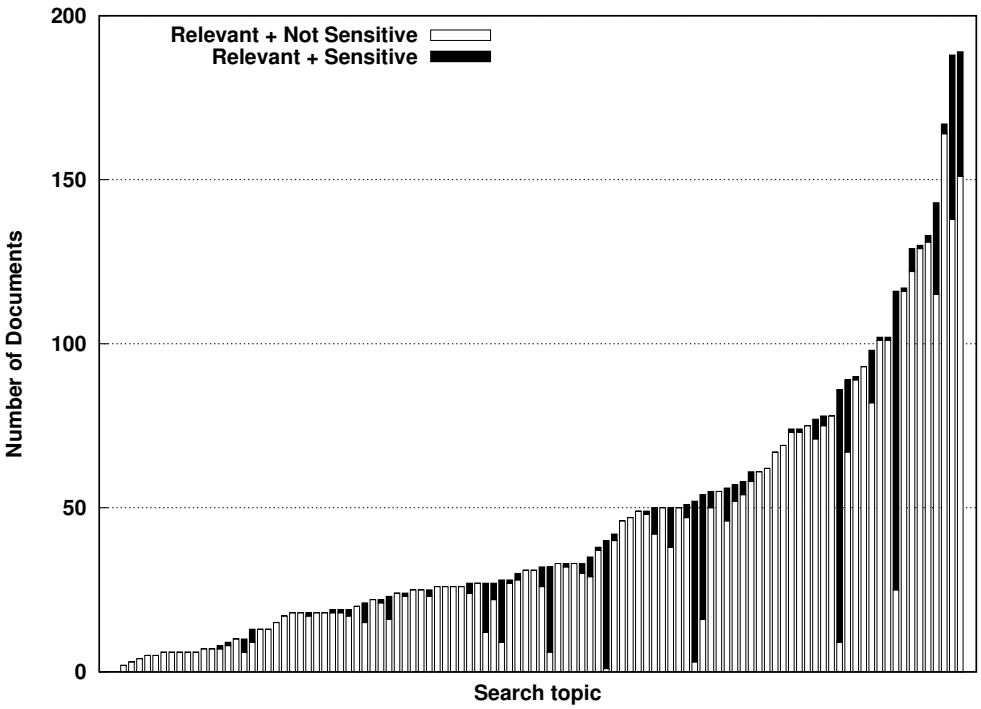


Figure 3.1: Number of relevant documents (to any degree) per topic in the OHSUMED test collection, with documents that are both relevant and sensitive shown in black at the top of each bar.

3.2 Realistic Sensitivity: Avocado Email Collection

In this section, we describe such a test collection for email, in which the sensitivities to be protected are those of the (modeled) content provider. This is not the first information retrieval test collection to contain sensitivity judgments. Hearst reports on the creation of annotations for about 1,700 messages from the (then newly released) Enron email collection [55]. The messages were annotated as part of a class project for categories such as secrecy, shame, and purely personal.[3] These might be considered sensitivity annotations, which were added in addition to annotations for topics such as political influence, the California energy crisis, or government actions.

[3] `http://bailando.berkeley.edu/enron/enron_categories.txt`

However, the small scale of the annotated collection limits its utility for evaluation of information retrieval systems. Jabbari et al. took up the challenge at larger scale a couple of years later, annotating about 14,000 Enron email messages as one of six categories of business messages, or as one of three categories of personal messages [58]. Although we might reasonably treat their "close personal" category as sensitive (as, for example, Hillary Clinton did when removing personal emails before turning over her professional email while serving as Secretary of State to the State Department), the six business categories are considerably broader than is typical of topics in an information retrieval test collection (e.g., core business, or routine admin). One promising result in that paper was that a classifier trained to detect close personal messages achieved 80% F_1, suggesting that sensitivity classification may be a tractable problem. Jabbari et al. also reminded us of the issues at stake in this line of work with the title of their paper.

Perhaps the most ambitious effort to date involving sensitivity annotation for email has been the TREC 2010 Legal Track, which annotated of the Enron email collection for both relevance (to specific requests for "production" of documents on some topic germane to a lawsuit), and sensitivity (in that case, for the legal concept broadly referred to as "privilege" in which documents can be withheld from production because of, for example, attorney-client privilege or the attorney work-product doctrine) [26]. One limitation of the TREC Legal Track test collection, however, is that because the relevance detection and sensitivity detection tasks were modeled separately, different documents were annotated for relevance and for sensitivity. It is not straightforward, therefore, to use that collection to measure

the ability of a system to find relevant documents while simultaneously withholding sensitive documents, even if those sensitive documents are relevant. For that, we need a test collection of the type we introduce in this dissertation, in which every annotated document is annotated both for relevance and for sensitivity.

Annotating a widely distributed test collection for sensitivity is actually a somewhat contradictory task. Since our goal is to protect the sensitive information in the collection, then the last thing we would want to do is to highlight where to find that sensitive content. Doing so in a public test collection such as Enron thus raises some ethical concerns that do not seem to us to have yet been adequately commented upon. We have therefore chosen to annotate the Avocado Research Email Collection, which is distributed by the Linguistic Data Consortium on a restricted research license that includes content nondisclosure provisions [87]. This license effectively precludes crowdsourcing, so all annotations were performed locally. The Avocado Research Email Collection consists of emails and attachments taken from 279 accounts of a defunct information technology company referred to as "Avocado". The collection includes messages, attachments, contacts, and tasks, of which we use only the messages and the attachments (concatenating the text in each message and all of its attachments). There are in total of 938,035 messages and 325,506 attachments.

An email collection should be carefully selected so that it has possibly sensitive content. As a result, Avocado Research Email Collection was selected to represent the source of emails. The collection consists of emails and attachments taken from 279 accounts of a defunct information technology company referred to as "Avocado".

The collection is broken up into emails, attachments, contacts, tasks, etc. There are in total of 938,035 emails and 325,506 attachments. The collection is distributed by the Linguistic Data Consortium on a restricted research license that includes content nondisclosure provisions [87]. Unlike Enron email collection [64], Avocado email collection has attachments and did not go under various redactions due to requests from affected employees. There are only 77 emails removed from the Avocado email collection. Those emails have the sensitivity metadata explicitly set, by the sender, to either "private" or "personal". This observation indicates that it is not common that the sensitivity level is set when an email is created. As a result, there is a need for automatic classification to predict for sensitivity.

To further protect specific individuals, a social science team (1 professor and 1 master's student) from the iSchool department at UMD created representative personas for two fictional individuals, and we used those personas as a basis for sensitivity annotations [56]. The persona represents the sender if the email was sent from an Avocado employee, or the recipient if the email was sent from outside the company network. The sensitivity of an email was annotated based on the persona's expected decision whether to allow the email to appear in search results. Relevance judgments were created for 65 topics. This section describes the process by which the topics were designed and the relevance judgments performed. The annotations will be distributed by the Linguistic Data Consortium as an addendum to the Avocado Research Email Collection.

To summarize, our target test collection consists of the following components.

1. Avocado email collection is used as a source of documents, where an email represents a unit of retrieval.

2. A set of search topics. We have 65 topics that range from conversations on finance and company related issues to health and personal conversations.

3. Set of relevance judgments for each of the search topics. By following the pooling method, these emails are returned from the top results by a number of different IR systems, and by searchers using an interactive keyword search. We have 100 different emails, on average, assessed for relevance with respect to each topic. So there are 65000 relevance judgments in total.

4. Sensitivity labels are given for the union of pooled emails across all search topics. Each email will be assessed whether it is sensitive or not.

3.2.1 Topic Creation

To test systems on relevance, a number of topics had to be created. These topics would later be used to query the email test collection, where users would be able to examine the query results to see if individual emails were relevant to the topic. To test systems on sensitivity, some topics must find sensitive content. With these goals in mind, four people (1 PhD student and 3 undergraduates at UMD) created 137 topics that were designed to explore many parts of an individual's life, from business matters like promotions and shareholders, to current events like the Olympics and Columbine, to personal matters, like drug use and vacations, to attitudinal topics like selfish and tired.

While creating the topics, the creator searched the test collection to verify that at least some emails were possibly relevant to those topics; the emails noted as possibly relevant were recorded for future use. Searching was done using terms from the topics themselves, and also other related terms. For each topic the creator created title, description, and narrative fields, as is often done in TREC topics. The descriptions are a few sentences that give more context to the topic than just the topic name, and the narrations are explicit instructions on which types of documents are relevant to the topic, and which ones are not.

3.2.2 Collecting Pooled Documents

In practice, it is not feasible to judge each document in the collection for relevance with respect to a given search topic. As a result, we followed the pooling method, which aims at sampling all documents that are predicted to be relevant and judge only these sampled documents. The pool of documents returned by all the systems for a given query should provide a good representation of all documents relevant to that query in the corpus. The end result is a much smaller set of documents to annotate, with a much higher proportion of relevant documents.

So in addition to interactive search results, described in the section above, we have implemented different automatic search systems and ran the same set of topics. We collected the top 25 results from each system and added them to the interactive search results. As a result, we have 100 documents, on average, to be judged for relevance and sensitivity for each search topic. We built 18 different search systems

by varying the following specs.

- Search keywords: topic title, title+description, or title+description+narrative.
- Query expansion: yes or no.
- Retrieval weighting models: DPH (derived from the Divergence From Randomness (DFR)), Okapi BM25, or TF_IDF. All these models are implemented in Terrier search engine.

3.2.3 Personas

'Sensitivity' is an ambiguous concept [80], with both personal [69] and social components [82]. Ideally, individuals would code their own content as sensitive to account for both individual preferences and social norms. But approaching individual Avocado employees would be intrusive, and is not allowed by the license terms. To understand consistent components of sensitivity in professional emails, the social science team therefore conducted interviews with 10 archivists who had worked with email collections and 9 distinguished academics whose email collections are of potential interest to future scholars. Interviews focused on identifying types of content that creators deem sensitive. The team developed a set of types as follows.

- Legally-protected information, e.g. personally identifiable information (PII), banking information, health information, student records, information relating to pending litigation.
- Business secrets, e.g. trade and strategy secrets, internal security information,

contracts, nondisclosure agreements

- Reputational risk which represents information that, if revealed, might risk a donor's reputation or even people mentioned in the conversation.
- Memberships and beliefs, e.g. religious groups.
- Evidence of stigmatized activity, e.g. drug use.
- Using offensive language in work emails, e.g. battles.
- Gossiping, making unfiltered or very very frank remarks.
- Expressing emotional content in professional situations.

Next, the team translated the qualitative interview data into a form that coders could use to annotate the test collection by using personas, a concept drawn from the human-centered design literature [24, 95]. Personas are archetypal representations of users that include their goals, attitudes and other relevant design details. The team found that respondents differed on both the diversity of information they found sensitive (from very few kinds of information to quite a lot), and how careful they felt they had been in their professional email practices (from uninhibited to circumspect). The team grouped our respondents into three types: 1) the cautious writer: circumspect and very sensitive; 2) careful without cause: circumspect despite the fact that they found fewer types of information to be sensitive; and 3) the diarist: uninhibited in their email habits despite finding many kinds of content sensitive. Each category was informed by at least two interview respondents. Finally, the team created personas as composite characters for the two categories that

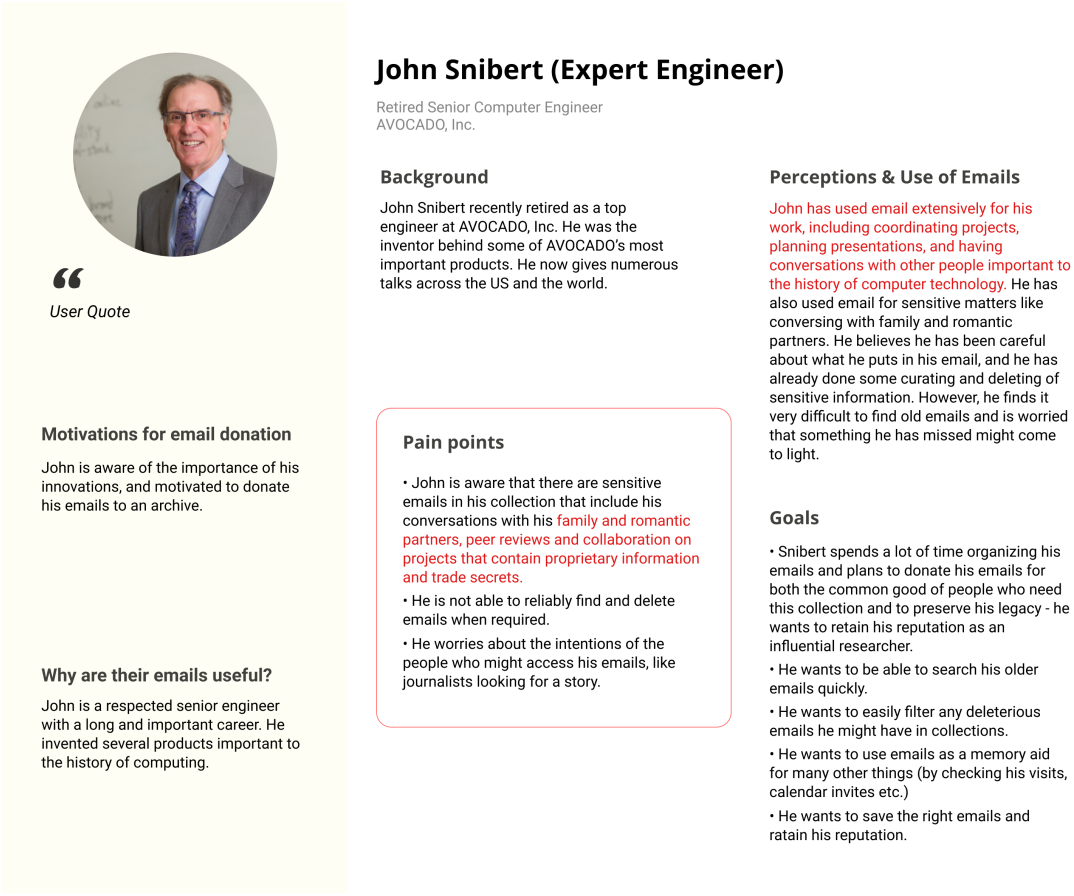


Figure 3.2: Persona of John Snibert

together reflected the broadest range of sensitivities, giving the characters names, and backstories loosely based on our interviews. The two personas are described in Figures 3.2 and 3.3.

John Snibert, as depicted in Figure 3.2, was the cautious writer: motivated to donate his emails to an archive because of their documentation of his career, and relatively assured of his care in writing emails over the years, but worried that he may have overlooked some of the many kinds of information about which he was sensitive. In contrast, Holly Palmer, as depicted in Figure 3.3, was the diarist: reluctant to donate her emails because she knows how much sensitive information they contain. The team gave each persona details about the character's background,

User Quote

Motivations for email donation

Holly is reluctant to donate her emails to an archive because she worries that, her emails will be taken out of context when accessed in archives. She might want to pass her emails on to her family. However, she does not feel her emails hold enough importance that it's worth the time and effort to curate them.

Why are their emails useful?

Holly has led several important research advances in the field of economics. Historians and economists would likely be interested in emails documenting her collaborations, research ideas, and research process.

Holly Palmer (Reluctant Professor)

Professor of Economics
Johns Hopkins University

Background

Palmer is a distinguished professor at Johns Hopkins University. She has won prestigious awards for her work in economics. Her papers and books are easily accessible on the internet but her communications and collaborations over the years are largely documented only in her email.

Perceptions & Use of Emails

She has used her work email to communicate with both personal and professional connections. She feels that her communication with them has been about work, home, logistics, gossips, and trade secrets (since she has consulted with Data and Tech companies for policy decisions). She feels that she has to pass on this collection to her family if no one else.

Pain points

- She is worried about the time and effort it will take to filter and delete conversations about her family and personal life.
- She is also mentioned travel and other receipts, and professional reviews of colleagues as information that she would not want shared.
- She is worried about potential harm to others (family and colleagues) if they discover unflattering things she has written about them.
- She is worried that her emails will be taken out of context.
- She does not understand why her emails would be useful to historians and scholars.

Goals

- Palmer is a busy professor, who does not have time or motivation to solve this problem - She wants a quick solution for her worst problem: unchecked social conversations.
- She wants to use the platform once a year for maintenance.
- She wants to understand why donation and archiving are important.
- She prefers to stay safe rather than save emails.

Figure 3.3: Persona of Holly Palmer

how they use email, and lastly, their particular concerns about sensitivity and the email types they consider sensitive. We adapted these personas to a corporate setting to match the nature of the Avocado collection, and then gave them to our annotators and asked them to infer sensitivity based on concerns expressed by the persona.

Our use of personas created a methodological challenge because the Avocado emails were authored by hundreds of individuals. Because creating hundreds of personas was untenable, and because the personas were written as composite people from a creator's point of view, we decided that the persona should represent the owner of any email that was being judged for sensitivity.

3.2.4 Collecting Annotation Results

Our target is to have a test collection of documents judged for relevance for at least 50 search topics. For half of the topics, we aim at judging their documents for sensitivity with regard to John's persona as in Figure 3.2, and the rest of documents (for the other half of topics) will be judged for sensitivity with regard to Holly's persona as in Figure 3.3.

During our experiments, we have two undergraduate UMD students who will be working as annotators. Each annotation task has 3 questions given a query document pair and one persona: 1) How relevant is the email to the specified query?, 2) Is the email sensitive according to the specified persona?, and 3) Is the email sensitive according to the annotator's personal opinion?. We developed an annotation

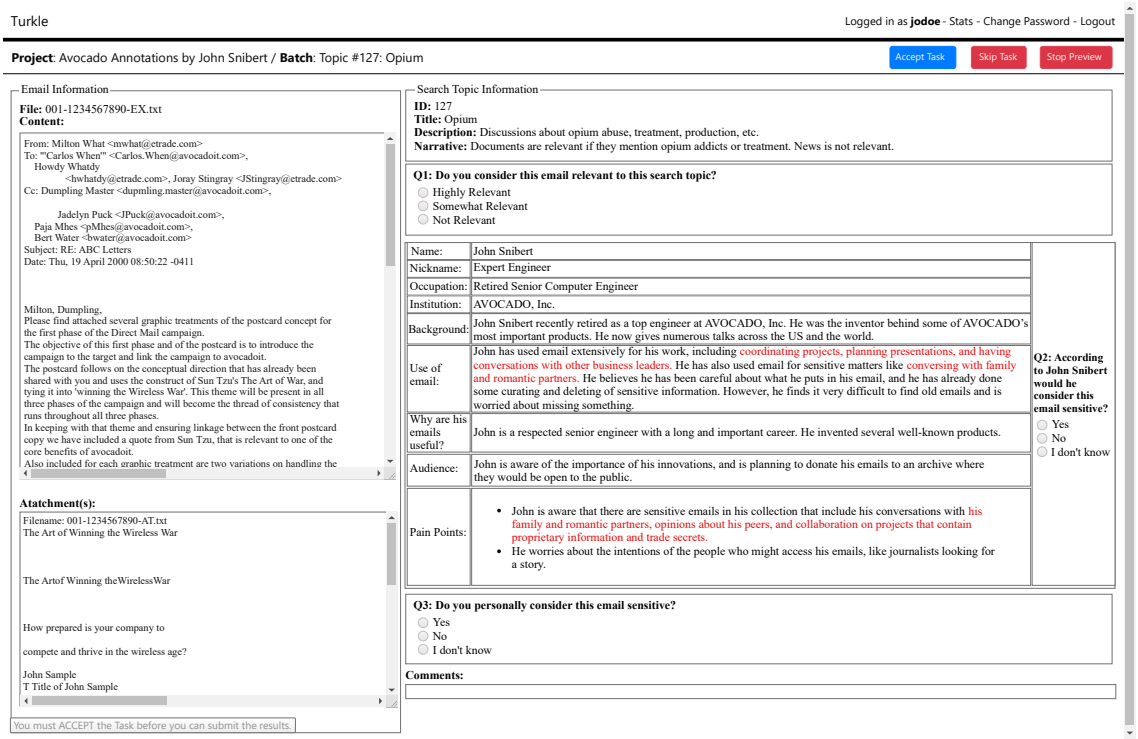


Figure 3.4: Sample Annotation Task (synthetic message and attachment created for public dissemination).

interface, as shown in Figure 3.4, by which our annotators used during training and actual annotations.

We started the training phase by launching two batches of tasks, where each batch represents the set of tasks related to a query's pooled documents. This phase has many benefits that will inspire us for the rest of the process. First, it shows the time taken in one annotation task, and hence we can plan how much time is needed for annotating 50 topics. Second, we can get feedback from annotators about the whole process if there are any aspects that need to be improved. Third, we can measure the inter-annotator agreement. Our plan is to train the two annotators on one persona for a set of topics. Then, once they reach a good agreement percentage. We will shift to the other persona (probably on the same set of topics used before)

until they reach a good agreement percentage. Then, we will let each annotator work on a different persona and different set of topics.

Round	**Topics**	**Rel.** κ	**Sens.** κ
1	#14: Bias, discrimination #133: Cubicles	0.22	0.27
2	#134: Drinking and hangover #51: Shareholder	0.23	0.15
3	#17: Porn #21: Lawsuit	0.76	0.66
4	#14: Bias, discrimination #134: Drinking and hangover	NA	0.53

Table 3.4: Annotator agreement, training phase, 532 observations. Rounds 1-3: John Snibert, Round 4: Holly Palmer.

Annotators were trained on system use and persona decision-making. There were two topics used for each of 4 training rounds; 3 for John Snibert and then 1 for Holly Palmer. Annotator agreement was measured per round using Cohen's kappa [5], as shown in Table 3.4. The first round tested the annotation system, and helped to acclimate the annotators to the process. After the second round, annotator disagreements on relevance and sensitivity were discussed. For example, one difference involved a joke, leading to a discussion on whether certain kinds of jokes might be sensitive. The third round tested the level of agreement between the two annotators on John Snibert's persona, again followed by a discussion of disagreements. The last round used the Holly Palmer persona with two recycled topics, both to acclimate one annotator to that persona and to test the level of agreement on that persona.

Kappa was computed as follows. We integrated a scale for our nominal variable, relevance, that measured highly relevant, somewhat relevant, and not relevant

documents. Table 3.4 shows little difference between rounds 1 and 2, but a large difference on round 3 after the discussion following round 2. Topic 17 is about Porn, which could indicate ease of agreements due to sensitive topics.

Excluding the 6 training topics, 65 topics were then randomly selected from the full set of topics and annotated, 35 for John Snibert by one annotator and 35 for Holly Palmer by the other annotator. The last 5 of the 35 topics in each set were duplicated in the other set, permitting recomputation of annotator agreement on relevance at the last stage of the annotation process. Table 3.5 shows that kappa on relevance at the end of the process was fairly high, except for Topic 135: Storage Space, which may have been more difficult to interpret consistently. Some reasons for low kappa coefficient could be due to lack of topic discussion between annotators, or human errors such as difference in knowledge, and topic narrative interpretation. On the other hand, one reason for a large kappa coefficient could be due to the clear directions of the topic narrative or the easiness of the topic, as indicated in Round 3. For instance, topic 113 was titled "Parents" with a narrative of "Documents are relevant if they contain discussions about the parents of those who work at Avocado or those who contact those who work at Avocado. All other emails about parents are irrelevant" where a variety of discussions about parents are relevant including meeting them, talking to them, or staying at their house.

Topics	Rel. κ
#13: Vacation	0.52
#84: Fax	0.70
#100: Chechnya	0.64
#113: Parents	0.71
#135: Storage Space	0.42

Table 3.5: Annotator agreement among common topics of both Avocado test collections, test phase, 585 observations.

3.2.5 Relevance Judgments

After collecting annotation results, there are 65 topics, each consisting of title, description, and narrative following the TREC topic format. There are 5 common search topics between the two personas. For each topic, a subset of the documents were judged for relevance. In total, there are 3567 and 3476 judged documents for Holly Palmer and John Snibert test collections respectively, each of which has a relevance judgment that indicates whether the document is highly, moderately, or not relevant to the query. Similarly to OHSUMED test collection, any document outside the pool of documents is considered not relevant. Because some documents were judged for relevance to more than one query, there are a total of 3314 and 3223 unique documents, in Holly Palmer and John Snibert test collections respectively, for which relevance to one or more queries has been judged. Table 3.6 shows the numbers of relevant documents (to any degree) and those documents that are not relevant.

Holly Palmer			
	Highly relevant	Moderately relevant	Not relevant
Sum	998	39	2,530
Avg.	30.24	3.25	72.29
Std.	28.08	3.29	47.83
Median	18	2	64
John Snibert			
	Highly relevant	Moderately relevant	Not relevant
Sum	999	37	2,440
Avg.	28.54	2.85	69.71
Std.	28.90	2.18	43.34
Median	26	2	60

Table 3.6: Relevance statistics about judged Avocado documents.

3.2.6 Sensitivity Judgments

After collecting sensitivity annotations, the two test collections have different distributions of sensitive content. Table 3.7 shows the number of sensitive documents and some of their statistics over search topics. It is clear to see that there are more sensitive documents with respect to John Snibert than to Holly Palmer. This may have occurred because John's discretion concerns were tailored for business rather than academic environments, and were therefore a better fit for the Avocado email collection. Many of Holly's concerns, such as the sensitive reviews of colleagues common in academia, would not have appeared as frequently in Avocado collection.

3.2.7 Empirical Analysis

Both the John Snibert and Holly Palmer personas have sensitivity and relevance judgments for 35 topics. The annotator for John Snibert had a median annotation time of 36 seconds; the annotator for Holly Palmer had a median an-

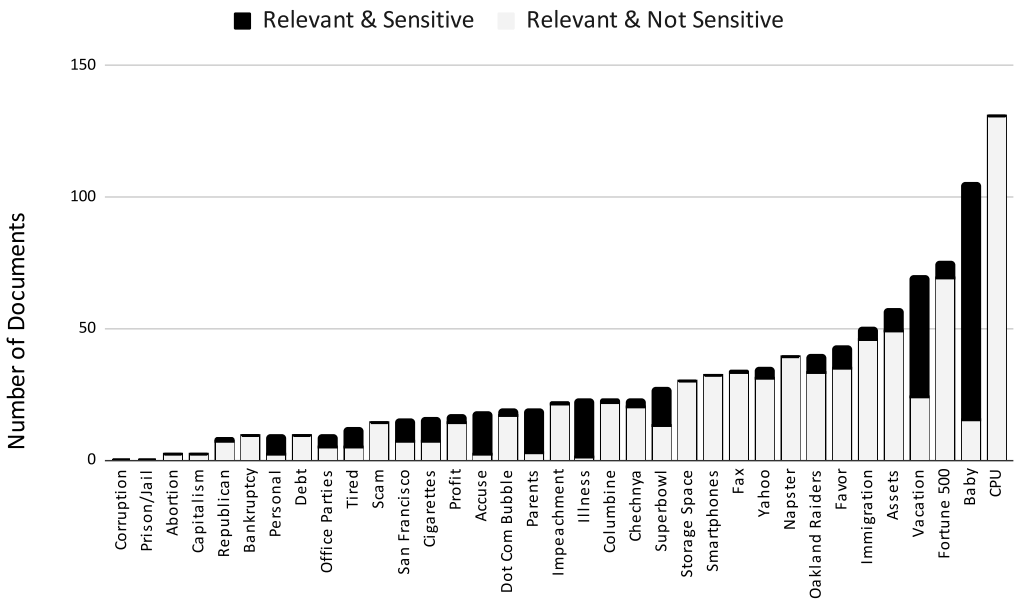
Holly Palmer		
	Sensitive	Not sensitive
Sum	579	2,988
Avg.	18.68	85.37
Std.	21.82	40.22
Median	10	78

John Snibert		
	Sensitive	Not sensitive
Sum	1618	1,858
Avg.	46.23	54.65
Std.	34.56	30.13
Median	40	51.5

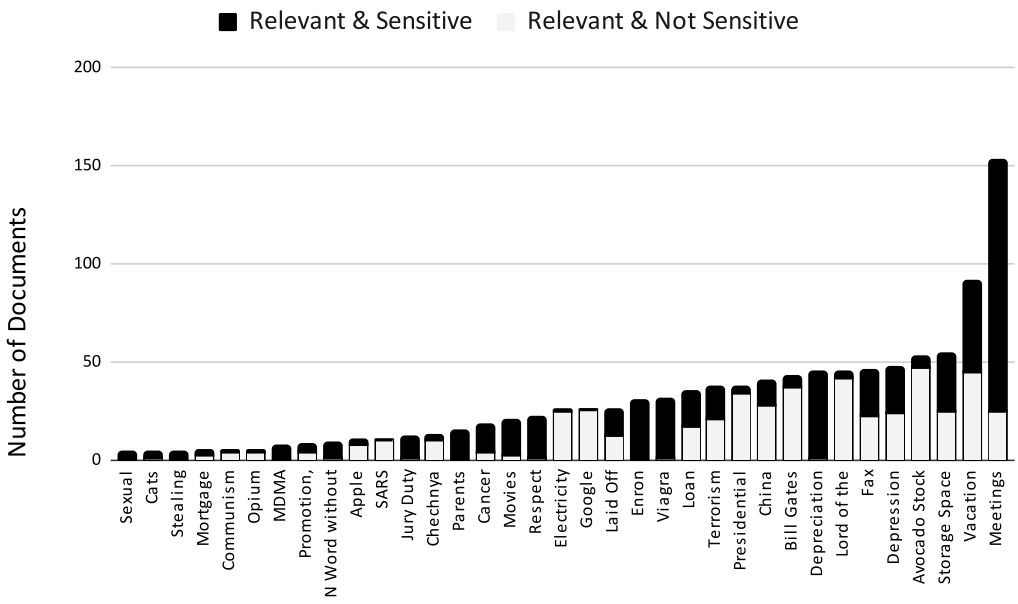
Table 3.7: Sensitivity statistics about judged Avocado documents.

notation time of 30 seconds. Over 35 topics, the John Snibert annotator found a mean of 29.8 relevant documents (to any degree) per topic and a mean of 46.2 sensitive documents per topic. Over a different set of 35 topics (5 of which were common with John Snibert), the Holly Palmer annotator also found a mean of 29.1 relevant documents (to any degree) per topic, but a mean of only 18.6 sensitive documents per topic. As Figure 3.5 shows, some documents have a high fraction of relevant documents that are also sensitive, whereas others do not. For example, it was expected to have many sensitive documents associated with the search topic "Meetings" in the John Snibert test collection, as these document might belong to trade secrets.

Like OHSUMED, every topic in both test collections has at least one relevant or sensitive document. Therefore, we did not remove search topic during our experiments. Table 3.8 shows relevance and sensitivity statistics for the Avocado test collections.



(a) Holly Palmer



(b) John Snibert

Figure 3.5: Number of relevant documents (to any degree) per topic, with documents that are both relevant and sensitive shown in black at the top of each bar. Holly Palmer (top), John Snibert (bottom).

Statistic	Holly Palmer	John Snibert
Mean Relevant	34.39%	33.06%
Median Relevant	25.20%	28.26%
Mean Sensitive	16.06%	46.86%
Median Sensitive	9.26%	45.90%

Table 3.8: Per query relevance and sensitivity statistics.

3.3 Relevance Features

To run Learning to Rank (LtR) algorithms, we need to extract features for each query-document pair. In OHSUMED, we adopt the features developed by the collection's authors which cover many classical features in IR, e.g. BM25 and language models [97]. Table 3.9 shows the relevance features extracted from query-document pairs in the OHSUMED test collection.

ID	**Feature Description**
1	sum TFs of query and title
2	sum log(TFs) of query and title
3	sum normalized TFs of query and title
4	sum log(normalized TFs) of query and title
5	sum IDFs of query and title
6	sum log(IDFs) of query and title
7	sum log(inverse prob. of term) of query and title
8	sum log(normalized TF_IDFs) of query and title
9	sum TF_IDFs of query and title
10	sum log(normalized TF * inverse prob. of term) of query and title
11	BM25 of query and title
12	log(BM25) of query and title
13	LMIR.DIR of query and title
14	LMIR.JM of query and title
15	LMIR.ABS of query and title
16-30	same as 1-15 but between query and abstract
31-45	same as 1-15 but between query and title+abstract

Table 3.9: LtR relevance features for the OHSUMED test collection.

For the two Avocado test collections, we have developed the set of features

ID	Feature Description
1-3	min,max,sum BM25 of title and subject
4-6	min,max,sum LMDIR of title and subject
7	normalized TF-IDF of title and subject
8	prob. sum of title and subject
9-11	min,max,sum DfrGl2 of title and subject
12-14	min,max,sum DfrInExpB2 of title and subject
15-17	min,max,sum Dph of title and subject
18	proximity of title and subject
19	TP score of title and subject
20	TPDist of title and subject
21	title length
22	unique term count of title
23	coverage ratio of title and subject
24	matching term count of title and subject
25	simplified clarity of title and subject
26-28	min,max,sum TFs of title and subject
29-31	min,max,sum TF-IDFs of title and subject
32-34	min,max,sum normalized TFs of title and subject
35-37	min,max,sum IDFs of title and subject
38-40	min,max,sum ICTFs of title and subject
41-80	same as 1-40 between title+desc and subject
81-160	same as previous between query and body
161-240	same as previous between query and attachments
241-320	same as previous between query and whole email

Table 3.10: LtR relevance features for the Avocado test collections.

shown in Table 3.10, which includes features we expect to be associated with relevance.

3.4 Sensitivity Features

In addition to relevance features in the OHSUMED test collection, we added the probability of document being sensitive (and its complement) as part of the vector representation, as shown in Table 3.11.

Based on our preliminary analysis, explained in Section 5.1, we added features

ID	Feature Description
46	probability of document being sensitive
47	1 - probability of document being sensitive

Table 3.11: LtR sensitivity features for the OHSUMED test collection.

ID	Feature Description
321-323	word length of subject, body, and attachments
324	1 if recipients are external; 0 otherwise
325	1 if recipients are internal; 0 otherwise
326	1 if sender is external; 0 otherwise
327	1 if sender is internal; 0 otherwise
328	1 if email is a reply; 0 otherwise
329	Number of attachments
330	1 if email has attachment(s); 0 otherwise
331	Sender's in-degree centrality
332	Sender's out-degree centrality
333	Sender's betweenness centrality
334	Sender's pagerank score
335-337	min,max,mean of recipients' in-degree centralities
338-340	min,max,mean of recipients' out-degree centralities
341-343	min,max,mean of recipients' betweenness centralities
344-347	min,max,mean of recipients' pagerank scores
348	probability of email being sensitive
349	1 - probability of email being sensitive

Table 3.12: LtR sensitivity features for the Avocado test collections.

extracted from email's content and metadata as shown in Table 3.12, which is a subset of features used in sensitivity classification.

3.5 Implementation

We have used several software toolkits throughout the process of creating the test collections.

1. Apache Solr[4] is used for building a keyword search engine on the Avocado email dataset. This tool played an important role in the process of creating

[4] `https://lucene.apache.org/solr/`

search topics and saving relevant documents during interactive search process.

2. Terrier[5] is used for building different search systems on the Avocado email dataset.

3. Turkle[6] which clones Amazon's Mechanical Turk service running in our local private environment. This tool helped us in creating annotation tasks and recording workers' results.

4. Flanker library[7] is used for parsing emails in raw format.

5. Networkx library[8] is used for constructing the email exchange network and computing network centralities.

6. Pyserini is used for building an index for the Avocado email collection and extracting different LtR features.[9]

3.6 Chapter Summary

In this chapter, we have created three test collections to support experimentation with a search and protection engine, and we have illustrated the use of that collection both for building a ranking model by constructing features that are possibly good indicators for both relevance and sensitivity. The first test collection is OHSUMED, and we used two MeSH labels to represent the sensitive content. Also,

[5] http://terrier.org/
[6] https://github.com/hltcoe/turkle
[7] https://github.com/mailgun/flanker
[8] https://networkx.github.io/
[9] https://github.com/castorini/pyserini

we have created two test collections based on the Avocado research email collection. The test collection has two personas that have different views of sensitivity. The John Snibert persona was motivated to donate his emails to an archive because of their documentation of his career; he was careful in his used in email, but still worried that he may have overlooked some of the many kinds of information about which he was sensitive. The Holly Palmer persona, by contrast, had originally been reluctant to donate her emails because she knows how much sensitive information they contain.

Chapter 4: Cost Sensitive Performance Measures[1]

In order to compare between different approaches for search and protection engines as will be explained later in Section 5.3, using an evaluation measure based on relevance only is not sufficient, as we expect to have lower relevance scores for those approaches that penalize sensitive documents. When searching among sensitive content, we must consider both the gain that results from showing a relevant document and the cost that accrues for showing a sensitive document. We introduce new measures that balance between relevance and sensitivity.

We want to evaluate the quality of a ranked result with respect to relevance and sensitivity. These two dimensions are conflicting; optimizing towards relevance possibly shows sensitive information, and optimizing towards sensitivity only may suffer from information loss. We list our potentially desired properties for evaluation measures that balance between sensitivity and relevance. During our experiments, we induced that our target is to maximize relevance of documents while hiding sensitive ones. When a new evaluation measure is developed, we need to raise to questions: 1) are there other evaluation measures that better capture the complexity of the target?, and 2) given an evaluation measure, how well can we build ranking

[1]Some parts of this chapter were taken from a paper in preparation by Sayed and Oard [105]

models optimized towards?

4.1 Measure#1: Simple Ternary Measure (TERN)

We start describing the proposed measures by TERN, which is simple to compute and understand. For a given query, It can take one of three possible values as follows.

$$TERN@k = \begin{cases} -M, & \text{if at least one sensitive document is shown} \\ 1, & \text{if no sensitive and at least one relevant document is shown} \\ 0, & \text{otherwise} \end{cases} \tag{4.1}$$

where M is a penalty for showing sensitive documents. According to the above equation, the TERN measure has the range [-M, 1]. So for a given query, if the score is negative, this indicates there is at least one sensitive document in the top k results. If the score is positive, this indicates there are relevant documents placed in the top k results. If the score is 0, this indicates that top k results are neither relevant nor sensitive.

4.2 Measure#2: SENS Measure

In contrast to TERN, the SENS evaluation measure is designed to behave like some standard relevance measure as long as there is no sensitive document in the result list. SENS could be defined using any standard relevance measure (nDCG, MAP, ERR, etc.); we define it using nDCG in the example below. However, if there

is at least one sensitive document, the evaluation measure should get the minimum value, regardless of the position of the sensitive result. It can be defined as follows.

$$SENS@k = \begin{cases} -M, & \text{if at least one sensitive document is shown} \\ nDCG@k, & \text{otherwise} \end{cases} \tag{4.2}$$

where M is as defined in TERN measure. It is worth noting that nDCG requires computing the ideal ranking, which should be based on non-sensitive documents only, in which case there should be at least k non-sensitive documents.

4.3 Measure#3: Cost Sensitive Discounted Cumulative Gain (CS-DCG)

This new measure is an extension based on 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.

Cost-Sensitive Discounted Cumulative Gain (CS-DCG) incorporates additional cost based on the document's sensitivity. In the simplest model, the cost for showing a sensitive document would be independent of where the document is in the ranked list. So CS-DCG up until rank position k can be defined as:

$$CS\text{-}DCG@k = \sum_{i=1}^{k} (g_i * d_i - c_i) \tag{4.3}$$

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 0.

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 [110]. 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. As with DCG, these bounds can be computed by the greedy rule-based algorithm shown in Algorithm 1 and 2 for the case in which $\max(c_i) > \max(g_i)$ and assuming binary sensitivity labels (e.g., sensitive or not). You can refer to Section 4.4 for the general case where there are graded relevance and graded sensitivity labels.

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 by putting less relevant documents on top of higher relevant ones.
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 by putting less relevant documents on top of higher relevant ones.
end if

Upper bounds are computed similarly as shown in Algorithm 2. An example running these algorithms for finding the best and worst ranking given a set of documents is shown in Figure 4.1.

Algorithm 2: Computing a query-specific upper bound on CS-DCG@k

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

Once we get the upper (best) and lower (worst) bounds for CS-DCG for a certain query q, we can normalize it using min-max normalization as follows.

$$nCS\text{-}DCG = \frac{CS\text{-}DCG - CS\text{-}DCG_{worst}}{CS\text{-}DCG_{best} - CS\text{-}DCG_{worst}} \tag{4.4}$$

So nCS-DCG could be equal to 1 when the documents are ranked indistinguishably from the best possible ranking, and 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 almost never anywhere in the ranked list. Note that we must cut the ranked list off at some point (hence

Figure 4.1: Example showing the best and worst rankings for a given set of documents with graded relevance (highly, somewhat, and not relevant) and binary sensitivity labels (sensitive or not).

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.

4.4 Measure#4: Differential Cost Sensitive Discounted Cumulative Gain (γCS-DCG)

In CS-DCG, the cost for showing a sensitive document is independent of where the document is in the ranked list. We propose a more general version of this measure by incorporating a discount based on how many sensitive documents appeared before the current one. Therefore, the proposed evaluation measure, named γCS-DCG, can be defined as follows.

$$\gamma CS\text{-}DCG@k = \sum_{i=1}^{k}(g_i * d_i - c_i * \gamma^{s_i}) \tag{4.5}$$

where γ is a geometric decay factor in the range [0, 1] for the cost of showing a sensitive document and s_i is the number of sensitive documents found in the ranked list before rank i. When γ is equal to 0, the measure penalizes only the first sensitive result. When γ is equal to 1, the measure yields to CS-DCG. Any value between these boundaries yields to different degrees of discounting the penalty for sensitive documents.

Unlike normalizing CS-DCG scores, it is hard to compute the best and worst rankings achieving the highest and lowest γCS-DCG scores respectively as documents have different penalties based on the number of preceding sensitive ones which makes the problem hard and our algorithms 1 and 2 do not solve such cases. As a result, we adopt a simple greedy approach to compute best and worst rankings. To compute the best ranking, we start from the first rank and select the document that maximizes the γCS-DCG score till the current rank position, and then do the same for subsequent positions until reaching rank cutoff (k). To compute the worst ranking, we follow the same procedure, but in each rank we select the document which minimizes the γCS-DCG score. This greedy approach does not guarantee optimal results, but these sub-optimal results should be acceptable in practice. For example, if we apply this greedy procedure on the example shown in Figure 4.1 while computing the worst ranking, the only somewhat relevant and sensitive document will be placed at the 4^{th} rank where it should be placed at the 10^{th} rank for the

optimal worst ranking.

Like CS-DCG, after computing best and worst rankings, one way of normalizing the scores is to use max-min normalization which produces nγCS-DCG that falls in the range [0,1]. It can be defined as follows.[2]

$$n\gamma CS\text{-}DCG = \frac{\gamma CS\text{-}DCG - \gamma CS\text{-}DCG_{worst}}{\gamma CS\text{-}DCG_{best} - \gamma CS\text{-}DCG_{worst}} \tag{4.6}$$

4.5 Evaluation Desiderata

Evaluation measures are models, and George Box reminds us that all models are wrong but some models are useful [12]. There thus can be no one "right" evaluation measure—the question is whether an evaluation measure has useful characteristics for some purpose. There are many things that we might wish of our evaluation measures, including that they offer insight into what we want to know, that they be easily explained or already widely understood, and that they be single-valued so that they can serve as a basis for optimization.

In this section, our aim is to build an evaluation measure that reflects how desirable a ranked list of documents is with respect to both the relevance of these documents to some query and also the sensitivity of these documents (irrespective of a query). By relying on axiomatic analysis [4, 19, 78], we list some formal properties that reflect which quality factors are captured by measures. Our plan is to formalize our desired properties for evaluation measures that balance between relevance and

[2]For a given query, if the γCS-DCG score lies outside this range, then the score is changed to the closest boundary's score.

sensitivity. Then we plan to propose several evaluation measures and check what properties are satisfied in each measure.

We adopt a simple model where any relevant (or sensitive) document has a relevance (or sensitivity) contribution, which is independent of other documents, affects the score of evaluation measure. We introduce the following definitions: $r(d)$ $\in$ [0, 1] represents the graded relevance of a document d, and $s(d) \in$ [0, 1] represents the graded sensitivity of a document d. We can assume there is a ranking cutoff (k), whereafter results are hidden, otherwise sensitive documents will be shown in any ranked result list. We use A to denote a ranking and $score(A)$ to denote the evaluation measure score for ranking A. Similarly to [4], we use $A_{d_i \leftrightarrow d_j}$ to denote the ranking A after replacing documents at positions i and j.

P1. Bounded, $\exists a, b \in R$: $score(A) \in [a, b]$. As a result, scores are on the same scale, which is a necessary condition to compare search systems across different information needs. If a measure is not bounded, it suffers from bias towards certain information needs that have many relevant or sensitive documents.

P2. Reference point, $\exists s, score(A) = s$ such that it is equivalent as if there are no relevant or sensitive documents in the result list, or possibly the gains, from relevant documents, and the costs, from sensitive documents, cancelled each other. When a measure has this property, it becomes easier to understand and interpret both relevance and sensitivity contributions.

P3. Convergence: given that we have a ranking cuttoff @k, if we swap a more relevant (less sensitive) document outside the top k with a document inside

the top k that is less relevant (more sensitive). Then the score should strictly increase.

- **Relevance convergence**: d′(outside top k), d(inside top k), and $r(d') > r(d)$ and $s(d') = s(d)$, then

  $score(A_{d \leftrightarrow d'}) > score(A)$.

- **Sensitivity convergence**: d′(outside top k), d(inside top k), and $s(d') < s(d)$ and $r(d') = r(d)$, then

  $score(A_{d \leftrightarrow d'}) > score(A)$.

P4. Head-weightedness: correctly swapping the order of adjacent documents inside the top k has more effect at earlier rankings.

- **Relevance head-weightedness**: $r(d_i) = r(d_j) < r(d_{i+1}) = r(d_{j+1})$, $s(d_i) = s(d_{i+1})$, and $s(d_j) = s(d_{j+1})$, then

  $score(A_{d_i \leftrightarrow d_{i+1}}) > score(A_{d_j \leftrightarrow d_{j+1}})$ when $i < j$.

- **Sensitivity head-weightedness**: $s(d_i) = s(d_j) > s(d_{i+1}) = s(d_{j+1})$, $r(d_i) = r(d_{i+1})$, and $r(d_j) = r(d_{j+1})$, then

  $score(A_{d_i \leftrightarrow d_{i+1}}) > score(A_{d_j \leftrightarrow d_{j+1}})$ when $i < j$.

P5. Localization: score is computed only using the top k results. Perhaps, this property may have an impact on efficiency while computing scores. When a measure is not localized, extra processing may be needed to compute the score that is based on the whole set of documents and not just the top k results.

P6. Realizability: score for any information need can actually reach the bounds as long as there is at least one relevant or sensitive document.

4.6 Evaluation Measures Analysis

In this section, we conduct an axiomatic evaluation of the proposed evaluation measures: TERN, SENS, CS-DCG, nCS-DCG, γCS-DCG, and nγCS-DCG. We study these measures by testing them for the desired properties, explained in Section 4.5. Table 4.1 shows that there is no one single measure that satisfies all the desired properties. For example, there is no measure that satisfies **P1**, **P2**, and **P6**. We did not put CS-DCG and nCS-DCG measures in Table 4.1 as they behave similarly to γCS-DCG and nγ CS-DCG respectively.[3]

It is clear that all measures are bounded, except CS-DCG and γCS-DCG. Unlike TERN and SENS, the lower bound of nCS-DCG and nγCS-DCG is 0, where it can be negative in the case of TERN or SENS measures.

One interesting consequence of the design of γCS-DCG is that properties **P1** and **P2** are in tension. In its unnormalized form, γCS-DCG satisfies property **P2** but not **P1**. For example, when the score is equal to 0, γCS-DCG indicates that the score is as good as no relevant or sensitive documents in the result list or their contributions cancel each other. We can use min-max normalization, producing nγCS-DCG, to satisfy property **P1**, but doing so then violates **P2**. It is this tension with interpretation of γCS-DCG that results in the TERN and SENS measures

[3]Actually, CS-DCG and nCS-DCG can be considered as special cases of their γ variants when γ=1.

being more easily interpreted.

Regarding convergence (**P3**), γ CS-DCG and nγ CS-DCG measures are convergent. The reason is that these measures reward (or penalize) a document according to its degree of relevance (or sensitivity). On the other hand, both TERN and SENS measures are not convergent, as it not necessarily true that the score will improve when one sensitive result becomes hidden or one new relevant result appears among the top k results. For example, if there are three sensitive results appeared in the ranked list, then removing one or two sensitive results will not improve the TERN score (it will still be -M). However, the score will improve when the ranking model succeeds in removing all 3 sensitive results, in which case the TERN score becomes non-negative.

Regarding head-weightedness (**P4**), γ CS-DCG and nγ CS-DCG measures are not head-weighted with respect to sensitivity as they penalize documents based on the number of preceding sensitive documents and not based on the rank. However, both measures are head-weighted with respect to relevance as they discount a document based on its rank, so correct swappings in earlier ranks have a bigger impact that in lower ranks. On the other hand, both SENS and TERN measures are not head-weighted as the score becomes negative if there is at least one sensitive result among the top results regardless of its rank.

Since, SENS and nγCS-DCG measure require knowledge of all relevant and/or sensitive documents for a given search topic, they are not localized (**P5**). However, the computation of TERN and γ CS-DCG depends on only the top k results. Because γ CS-DCG is not bounded, it is not realizable. Furthermore, TERN and

Property	TERN	SENS	γCS-DCG	nγCS-DCG
P1. Bounded	✓	✓	✗	✓
P2. Reference Point	✓	✓	✓	✗
P3a. Relevance Convergent	✗	✗	✓	✓
P3b. Sensitivity Convergent	✗	✗	✓	✓
P4a. Relevance Head-weighted	✗	✗	✓	✓
P4b. Sensitivity Head-weighted	✗	✗	✗	✗
P5. Local	✓	✗	✓	✗
P6. Realizable	✗	✗	✗	✓

Table 4.1: Comparison between all proposed evaluation measures for relevance and sensitivity with respect to the desired properties for search among sensitive content.

SENS measures are not realizable, as they can not be negative if a certain search topic has no sensitive documents. On the other hand, nγCS-DCG can reach both lower and upper bounds for any search topic as long as there is at least one relevant or sensitive document, and hence it is realizable.

4.7 Chapter Summary

In this chapter, we ask how would we measure the quality of ranked lists with respect to relevance and sensitivity? Answering that led us to propose several evaluation measures for search among sensitive content. Additionally, we have shown how some standard evaluation measures can be extended to incorporate sensitivity in addition to relevance. We developed a set of potentially desirable properties for the task of search among sensitive content. We analyzed the proposed evaluation measures in terms of whether they satisfy each of the properties. Analysis shows that there is no measure that satisfies all desired properties. And hence, we found that some measures are good for evaluating ranked lists, but they contain quantization noise which is not suitable when used for training ranking models (Section 6.1).

Chapter 5: Proposed Approaches[1]

When searching among sensitive content, search and protection engines should possess the following characteristics.

1. They still retrieve relevant content in responding to information needs.

2. They protect sensitive content which may put the content owner, or possibly the searcher, under the risk of disclosure.

3. They are fast, which will enable effective access to information without waiting for a long time between search request and its response. As a result, these engines support interactive query reformulation by the user who has the information need.

4. They are affordable, which will save manual review costs incurred by hiring reviewers. Reviewers are often paid on a per-hour basis, which can also make project budgets hard to predict.

To satisfy the aforementioned characteristics, we propose to use two technologies for the target retrieval task: 1) Learning to rank (LtR), and 2) Automatic sensitivity classification. Unlike conventional retrieval functions, e.g. BM25, LtR

[1]Some parts of this chapter were taken from a publication by Sayed and Oard [106] and a paper in preparation by Sayed, Mallekav, and Oard [104]

has transformed the ranking problem into a learning problem in which case each query-document pair 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 [68]: 1) Pointwise, 2) Pairwise, and 3) Listwise approaches. In section 5.2, we describe several LtR algorithms that span all LtR groups.

Often it is not feasible to give sensitivity labels for all the content that we have, and hence we need an automatic sensitivity classifier which predicts a document's sensitivity. In section 5.1, we describe the process of building different classifiers in terms of classification models, feature selection, and train/test split.

In Section 5.3, we generally propose different baselines for combining a sensitivity classifier with a ranking model for the target retrieval task. Then in Section 5.3.2, we propose an approach that leverages evaluation-driven Learning to Rank (LtR) techniques. Our work focuses on the listwise approaches, as they try to directly optimize an objective function that can be matched to the evaluation measure, as presented in Chapter 4.

Finally, in Section 5.4 shows the efficacy of the proposed approach in comparison with the baselines. These results were validated using all our test collections, presented in Chapter 3, and using different evaluation measures, presented in Chapter 4. Also, we show it is not a good practice to use the same measure for evaluating and optimizing search and protection engines. This observation comes as a result of not having one measure that satisfies all the desired properties as presented in

Section 4.5.

5.1 Sensitivity Classification Approaches

In this section, we describe the process of building sensitivity classification models for the OHSUMED and Avocado test collections. Also, we describe the process of feature engineering and train/test split.

5.1.1 Classifying OHSUMED Documents

To approximate the degree of accuracy that might be seen in an actual application for search among sensitive content, we used 2 MeSH labels to represent the sensitive content which are C12 (Male Urogenital Diseases) and C13 (Female Urogenital Diseases and Pregnancy Complications), as discussed in Section 3.1. We used the 14,430 documents that were judged for relevance as our test set, and the remaining 334,136 documents in the OHSUMED collection for training. We build different classification models using Logistic Regression (LR) and Support Vector Machines (SVM). One advantage of LR is that predicted sensitivity probabilities can be directly obtained without calibration. Each document is represented by a TF-IDF vector for the words found in the title and abstract. For hyper-parameter tuning, we run grid search with 5-fold cross validation on the training data and then pick the set of values that maximizes F_1. Table 5.1 lists the ranges of hyper-parameters for LR and SVM.

From the fact that our classes are unbalanced and that a false negative (mis-

Model	Parameter	Parameter Space
LR	use_idf	{False, True}
	stemmer	{None, Porter}
	C	{0.01, 0.1, 1, 5, 10}
	probability	{0.01, 0.02, ... 0.99, 1}
SVM	kernel	{rbf, linear}
	γ(rbf only)	{1e0, 1e-1, 1e-2, 1e-3, 1e-4}
	C	{0.01, 0.1, 1, 5, 10}

Table 5.1: Hyper-parameter ranges for LR and SVM classifiers. Note that $\gamma$is a kernel coefficient, which has a different meaning from γin γCS-DCG evaluation measure.

Model	**Precision**	**Recall**	**F_1**	**F_2**	**Accuracy**	**BAC**
LR	78.10	**72.44**	**75.16**	**73.50**	94.14	**84.80**
SVM	**86.35**	63.72	73.33	67.25	**94.32**	81.16

Table 5.2: Sensitivity classification results for OHSUMED (using test hold-out).

takenly 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_2 or even F_4, which gives considerably more weight to recall than precision. However, we decided to optimize for F_1 because later in Section 5.5, we show that improving F_1 measure is important to build more effective search and protection engines under specific design choices.

As shown in Table 5.2, we measure our results using several classification measures. Due to the nature of sensitivity classification, the error of a false negative (classifying a sensitive document as non-sensitive) may be more harmful than a false positive (classifying a non-sensitive document as sensitive). As a result, we report F_2 in addition to F_1, as F_2 gives more weight to recall over precision. We also report Balanced Accuracy (BAC) to mitigate the effect of class imbalance (note that in the OHSUMED test collection, ~12% of judged documents are sensitive).

5.1.2 Classifying Avocado Messages

The Avocado test collection, described in Section 3.2, consists of a set of search topics or information needs. For each search topic, a number of pooled messages is judged for relevance and sensitivity. To construct our test collection for message sensitivity classification, we group all pooled messages along with their sensitivity decisions, and create a test collection for each persona. We perform some filtering steps on these messages and their labels. First, we remove all messages that are judged for relevance with undetermined sensitivity label, which happens when the annotator can not decide about sensitivity. As a result, there are 4 and 35 removed messages from Holly Palmer and John Snibert test collections, respectively. Second, if a message was judged for relevance with respect to different search topics. There could be a possibility that the same message is given different sensitivity labels due to annotator's inconsistency. For such a case, we always consider this message as sensitive if it is annotated as sensitive in one of the judgments. As a result, there are 24 and 57 messages with conflicts that are considered to be sensitive. So after preprocessing, each message represents one sample in the dataset and the label is the annotator's opinion of the persona's decision about the message is sensitive or not. A summary of the number of sensitive documents for each persona is given in Table 5.3.

We next study how to classify a message. Different features can be useful for message classification task (e.g., message headers, message body, and message exchange network) [2,58]. We do not have a large number of messages for training. So

Persona	#sensitive	#not sensitive	%sensitive
Holly Palmer	563	2,751	16.99%
John Snibert	1,531	1,692	47.50%

Table 5.3: Number of sensitive/not sensitive documents for each persona, Avocado.

we propose to use statistical machine learning techniques. However, in Section 5.5, we fine-tune pre-trained neural network, e.g. BERT [38], for the classification task.

5.1.2.1 Characterizing Message Sensitivity

In this section, we do preliminary analysis of different factors that can help in recognizing sensitive messages. Results from this section guide us as to which features we should extract to build classification models to predict message sensitivity. Here, the strategy we adopt is to study different factors to check whether different feature values can change the prevalence of sensitive documents or not. As presented in Table 5.3, the base percentages of sensitive documents are ∼17% and ∼48% for the Holly Palmer and John Snibert test collections respectively.

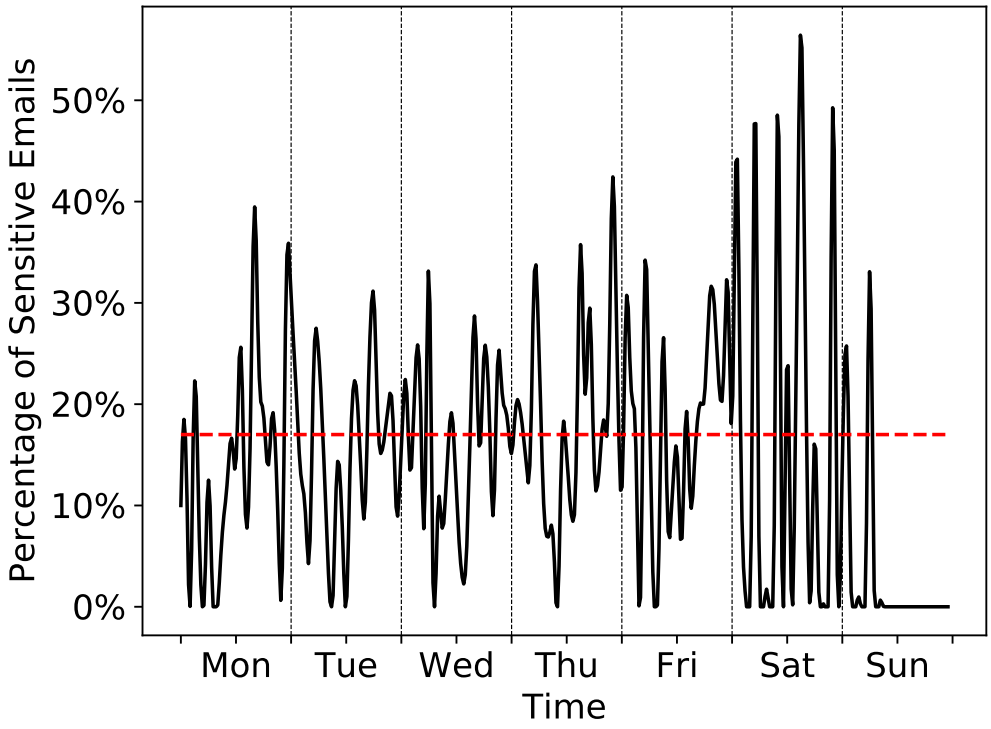
Time-based Factors. We first study the impact of time on message sensitivity. We partition the time of day by hour, i.e. {0-1, 1-2, ..., 23-24}, and then we aggregate all judged messages in each time range. We do not normalize time in message's metadata (e.g., convert time to UTC timezone), as we want to study the impact of time of day based on message owner's original timezone.

Figures 5.1(b) and 5.1(a) show both the number of sensitive and not sensitive messages broken by time of a day and day of a week for John Snibert and Holly Palmer respectively. It is shown that the number of sensitive messages increases

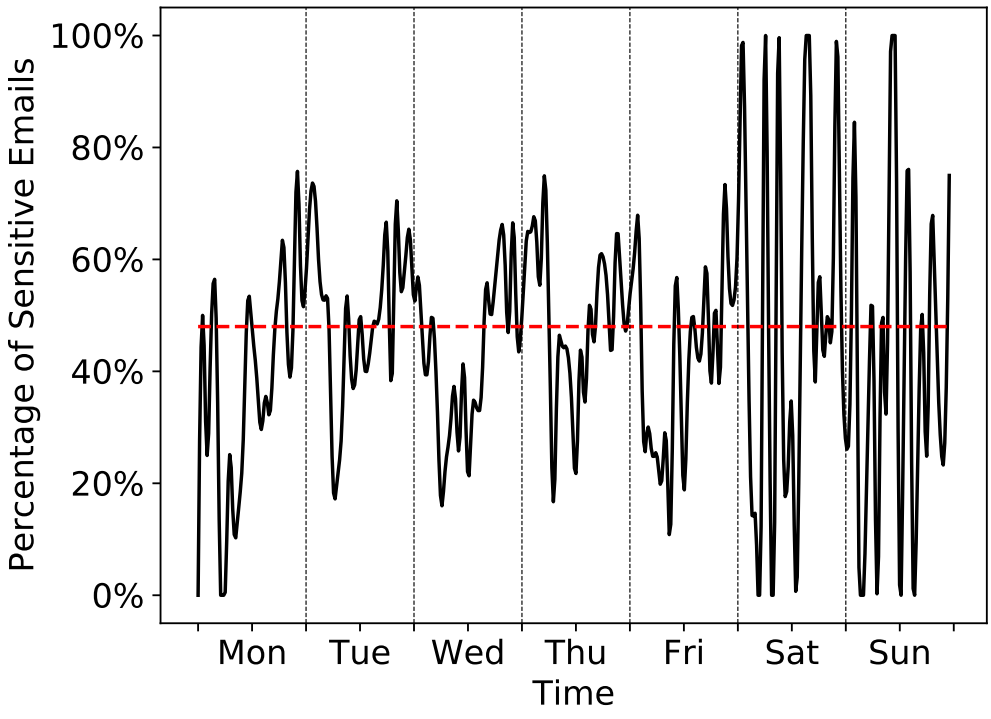
later in the day, i.e. after noon or in the evening. For example, on Tuesdays, John Snibert has even more sensitive messages than not sensitive ones. For Holly Palmer, we found many messages sent in the evening were related to family conversations.

Address Type Factors. We study the impact of address type (internal or external) on message sensitivity. We consider an email address as internal if the domain of the email address is "@avocadoit.com" which is used to replace the name of the company. Otherwise, the email address is considered as external. We aggregate messages by address type and compute the percentage of sensitive messages over all messages. First, we analyze the address type of the message sender. We aggregate messages sent from each address type and compute the percentage of sensitive messages over all judged messages. Then, we repeat the same analysis on the recipient(s). The only difference in the latter case is that we consider recipients as internal if any of them has an internal address. Otherwise, the recipients are considered as external.

Table 5.4 shows that the fraction of sensitive messages is higher when the message is sent from an internal address. This is because the message is most likely related to the company and have sensitive information (65.2% for John and 30% for Holly) such as information about company projects, documents, or finance. However, most of the messages received from an external address are coming from newsletters, which have a lower rate of sensitive messages. The rate of sensitive messages increase when one of the recipients is inside the company. Most of the messages sent to external addresses are sent to mailing lists, and hence there are few sensitive messages among them (30.8% for John and 14.9% for Holly).



(a) Holly Palmer



(b) John Snibert

Figure 5.1: Effect of time during the week on message sensitivity. Red line represents the base percentage of sensitive documents.

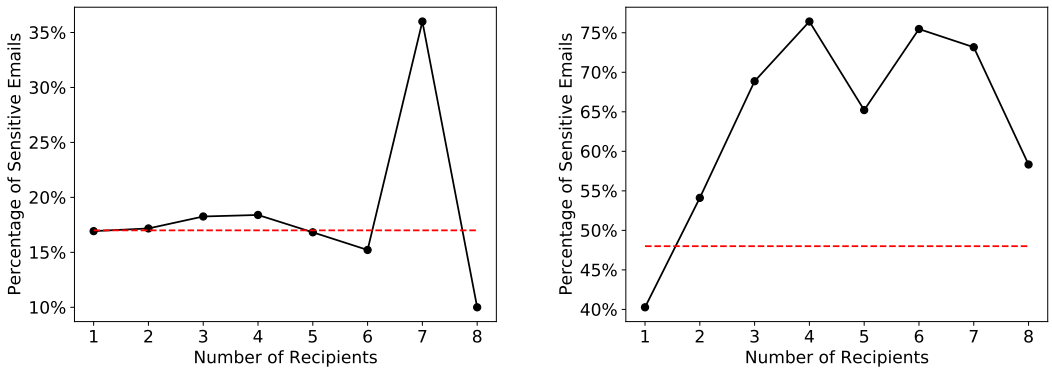
Persona	Sender Type		Recipient(s) Type	
	Internal	External	Internal	External
Holly Palmer	30.0%	10.0%	17.9%	14.9%
John Snibert	65.2%	30.0%	53.2%	30.8%

Table 5.4: Effect of email address type on message sensitivity.

Number of Message Recipient(s). We study the impact of the number of message recipients(s) on message sensitivity. We aggregate messages having the same number of recipients and compute the percentage of sensitive messages over all judged messages. We do not remove the sender email address if found in the recipients list, i.e. message is sent to self. This is to avoid cases when messages might have empty list recipients, e.g. message is only sent to the sender. Most messages are sent to a maximum of 8 recipients. One caveat is that we treat the email address for a mailing list as one recipient and not based on the number of people joined in that list.

Figures 5.2(b) and 5.2(a) show the fraction of sensitive messages with respect to the number of recipients for John Snibert and Holly Palmer, respectively. In the John Snibert test collection, we found more sensitive messages when the number of recipients is (>2). In these messages, we found many of them are related to company arrangements which were assessed as trade secrets.

job, department, country, and level **Roles Involved**. We study the impact of the sender's job level as categorized by Byun and Kirsch [20] on message's sensitivity. Also, the authors extracted the job title, department, and country. There are five job levels: L5: CEO, L4: Founder, VP, L3: Director / Sr Mgr, L2: Manager, L1: ETC. We added another job level (L0) for any email address that does not belong

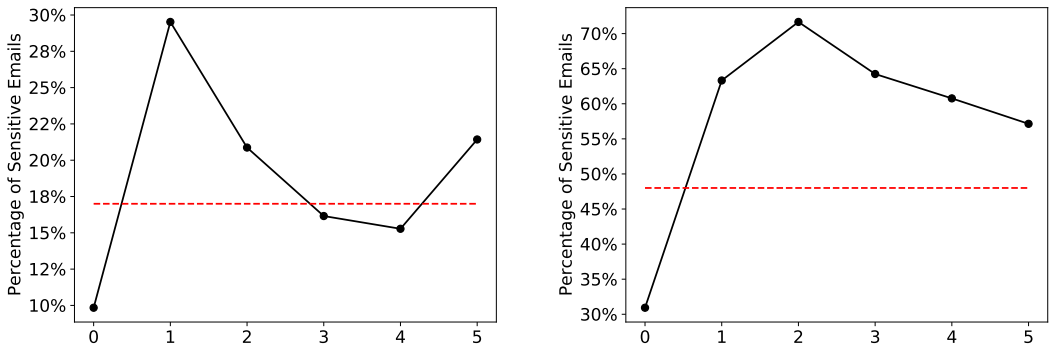


(a) Holly Palmer (b) John Snibert

Figure 5.2: Effect of number of recipients on message sensitivity. Red line represents the base percentage of sensitive documents.

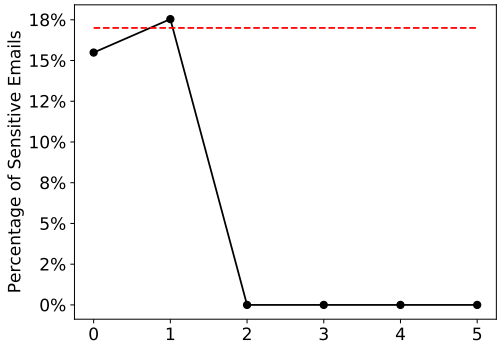
to the company. In cases when a message has multiple recipients, we compute these features only for the recipients with the lowest and highest job levels. As shown in Figure 5.3(b), when the sender's job level is $\geqslant 1$ the prevalence of sensitive content is increased as seen in the John Snibert test collection. In the Holly Palmer test collection, the fraction of sensitive content is high when the sender's job level is 1. We found out the job levels of the recipients did not help in discriminating sensitive content.

Message Metadata. These features represent the priority and importance of the message set by the sender, and is intended for the attention of the recipient. Priority can be either "nonurgent" (∼32% in John's and ∼29% in Holly's), "normal" (∼66% in John's and ∼68% in Holly's), or "urgent" (∼2% in both datasets). Importance can be either "low" (a handful of cases), "normal" (∼97% in both test collections), or "high" (∼3% in both test collections). Our annotators did not check this metadata during the review process as this metadata are not part of the message's raw text, but they can be found in a separate metadata file. Even though

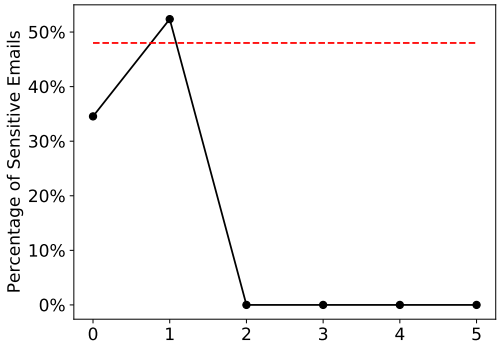


(a) Sender's Job Level - Holly Palmer

(b) Sender's Job Level - John Snibert



(c) Highest Recipient - Holly Palmer



(d) Highest Recipient - John Snibert

Figure 5.3: Effect of roles ranks on message sensitivity. Red line represents the base percentage of sensitive documents.

Persona	Priority		
	nonurgent	**normal**	**urgent**
Holly Palmer	7.9%	21.0%	14.5%
John Snibert	31.7%	54.8%	82.2%
Persona	**Importance**		
	low	**normal**	**high**
Holly Palmer	0%	17.2%	13.4%
John Snibert	0%	47.2%	76.7%

Table 5.5: Effect of message metadata (priority and importance) on message sensitivity.

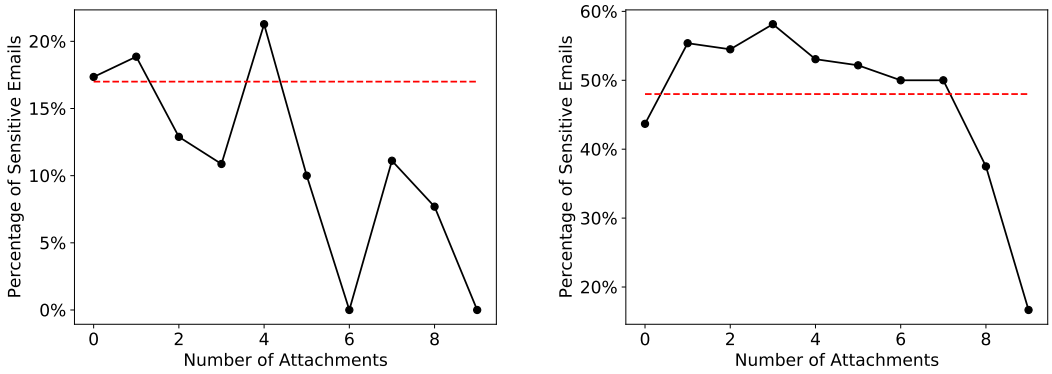
Persona	Has Attachments	No Attachments
Holly Palmer	15.37%	17.74%
John Snibert	56.93%	42.60%

Table 5.6: Effect of having attachments on message sensitivity.

these features take the default values for most of the messages in both test collections, they can still help in discriminating sensitive messages. Table 5.5 shows that when the priority is urgent or importance is high, the fraction of sensitive messages is high in the John Snibert test collection. Also, when the message's priority is nonurgent, the fraction of sensitive messages is low in both test collections.

Message Attachments. We study the effect of the presence of attachments on message sensitivity. Table 5.6 shows that the fraction of sensitive messages is high in the John Snibert test collection when a message has attachments. Figure 5.4 shows the fraction of sensitive messages by the number of attachments.

Message Reply. We study the effect of a message being a reply on message sensitivity. Table 5.7 shows that there is higher rate of sensitive messages when the message is a reply. We found some of these messages are between company's employees. On the other hand, newsletters do not require replies and hence they contribute a big portion of the non-sensitive messages.



(a) Holly Palmer (b) John Snibert

Figure 5.4: Effect of number of attachments on message sensitivity. Red line represents the base percentage of sensitive documents.

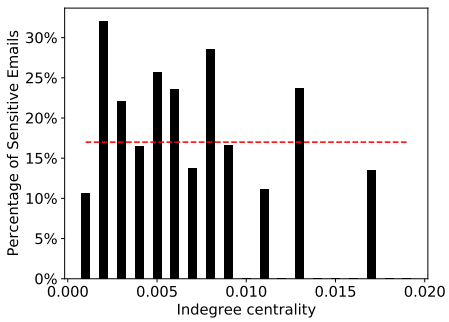
Persona	**Is a Reply**	**Not a Reply**
Holly Palmer	29.9%	14.2%
John Snibert	72.0%	42.2%

Table 5.7: Effect of being a reply on message sensitivity.

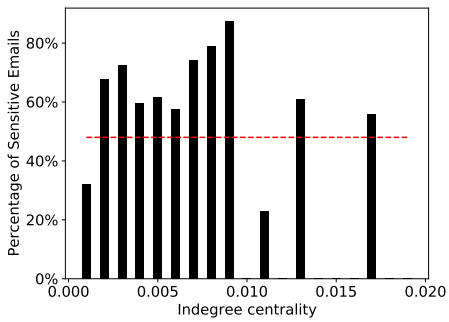
Message Exchange Network. We analyze the message exchange network based on all messages in the collection. We construct a directed graph that consists of nodes, each of which represents an email address, and directed edges, each of which exists if there is at least one message sent from the source node (i.e., email address) to the destination node. First, we remove all duplicate messages found in the collection. A message is considered as duplicate if there is another message has the same message ID and the same subject line. For example, a message is sent from employee A to employee B. Now, this message has two versions: 1) one version in the sent folder of employee A, and 2) another one in the inbox folder of employee B. For such cases, we keep only the sender's version of a message and make it the canonical version in a duplicate set. Second, for each message, we extract the sender email address from the From field, and all recipients found in the To, Cc, and Bcc

fields. Then, for each recipient, we add a directed edge between the source node that represents the sender to the recipient's node.

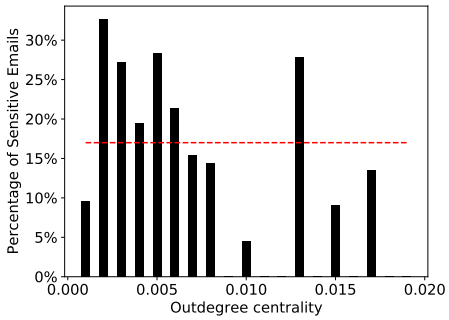
Network features are generated by leveraging the message exchange network of the whole Avocado email collection. We want to capture the importance of nodes that represent the sender and recipient(s). For each node, we extract that node's scores for four network statistics (in-degree centrality, out-degree centrality, betweenness centrality, and pagerank). Degree centrality is a measure of the number of edges a particular node has in the network. In the case of directed graphs, we break it into two measures based on in-degree and out-degree edges, respectively. Betweenness centrality for a node is a measure that reflects the fraction of the shortest paths, between any pair of other nodes, that pass through the node. The nodes with high betweenness act as gateways that play a significant role in the communication/information flow within the network. Pagerank denotes to the importance of a node based on the neighbors. In cases when a message has multiple recipients, we compute the min, max, and mean scores for each of the network statistics for the recipients. Figure 5.5 shows the fraction of sensitive messages based on the sender's node scores. As can be seen in Figure 5.5(b), the fraction of sensitive messages is high when the sender's in or out-degree centralities are in the range [0.001,0.009] in the John Snibert test collection.



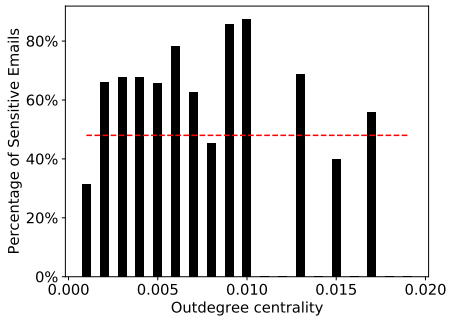
(a) In degree centrality - Holly Palmer



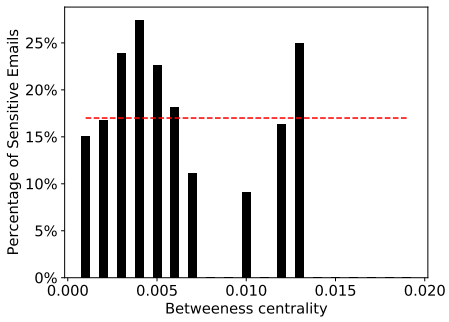
(b) In degree centrality - John Snibert



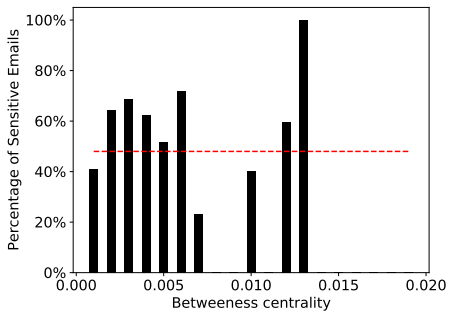
(c) Out degree centrality - Holly Palmer



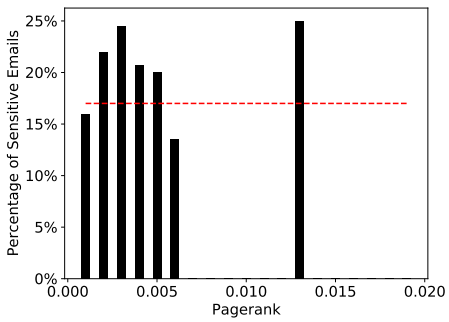
(d) Out degree centrality - John Snibert



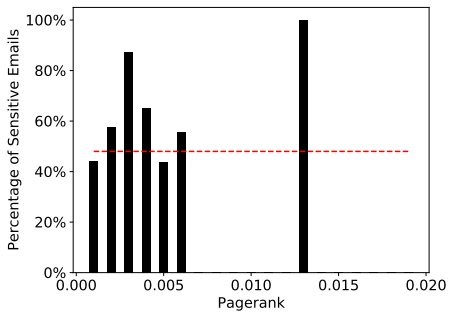
(e) Betweenness centrality - Holly Palmer



(f) Betweenness centrality - John Snibert



(g) PageRank - Holly Palmer



(h) PageRank - John Snibert

Figure 5.5: Effect of sender's node centrality on message sensitivity. Red line represents the base percentage of sensitive documents.

5.1.2.2 Predicting Message Sensitivity

Because messages contain metadata in addition to text, and because some of that metadata defines temporal or graph structures that could be useful for predicting sensitivity, we extract a rich set of features as shown in Table 5.8. Note that this is a richer set of features than is directly available to our LtR systems, so the sensitivity classifier fills a feature aggregation role when its output is used as one basis (among many) for LtR as shown in Table 3.12.

For each test collection, we build 2 different classification models via Logistic Regression and Support Vector Machines. We tune the hyper-parameters, as presented in Table 5.1, using grid search with 5-fold cross validation on the training data and then pick the value that maximizes F_1. Results, as shown in Table 5.9, show that LR and SVM have comparable results, but we used LR for integration with a ranking model as LR's probabilities are normally well calibrated.

5.2 Ranking Approaches

For testing our baselines and our approach, we used different algorithms to build LtR models [68]. We selected a representative algorithm from the pointwise and pairwise approaches. Additionally, we selected two algorithms from the listwise approach to show that our proposed approach works with different listwise ranking algorithms. 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 is a pointwise approach that learns a linear ranking function

Feature	Description
subjectTFIDF	TF-IDF weights of the terms in the subject field.
bodyTFIDF	TF-IDF weights for the terms in the message's body.
attachmentsTFIDF	TF-IDF weights for the terms in attachment(s).
dayOfWeek	7 binary features for the day of the week, e.g. Monday.
timeOfDay	4 binary features for the time of day (0-6, ..., 18-24).
isWeekend	1 binary feature if day is a weekend and 0 otherwise.
isSenderExternal	1 binary feature if sender is external and 0 otherwise.
isReceiverExternal	1 binary feature if all recipients are external and 0 otherwise.
numRecipients	total number of recipients in "To", "Cc", and "Bcc" fields.
subjectLen	number of words in the subject field.
bodyLen	number of words in the message's body
attachmentsLen	number of words in attachment(s).
senderInDegCent	in-degree centrality score for message's sender.
senderOutDegCent	out-degree centrality score for message's sender.
senderBetweennessCent	betweeness centrality score for message's sender.
senderPageRank	pagerank score for message's sender.
receiverInDegCent	3 features for min/max/mean of in-degree scores for recipient(s).
receiverOutDegCent	3 features for min/max/mean of out-degree scores for recipient(s).
receiverBetweennessCent	3 features for min/max/mean of betweeness scores for recipient(s).
receiverPageRank	3 features for min/max/mean of pagerank scores for recipient(s).
senderJob	1 categorical feature to represent the job title of message's sender.
senderDepartment	1 categorical feature to represent the department of message's sender.
senderRank	1 categorical feature to represent the job rank of message's sender.
senderCountry	1 categorical feature to represent the country of message's sender.
receiverJob	2 categorical features for min and max job titles of message's recipient(s).
receiverDepartment	2 categorical features for min and max department of message's recipient(s).
receiverRank	2 categorical features for min and max ranks of message's recipient(s).
receiverCountry	2 categorical features for min and max country of message's recipient(s).
importance	3 binary features for message's importance (low, medium, high).
priority	3 binary features for message's priority (nonurgent, normal, urgent).
sensitivity	2 binary features for message's sensitivity (none, company confidential).
hasAttachment	1 binary feature indicating whether the message has attachments.
numAttachments	1 integer feature indicating the number of attachments in the message.

Table 5.8: Description of features used for training and predicting message's sensitivity

Persona	Model	Precision	Recall	F_1	F_2	Accuracy	BAC
Holly Palmer	LR	**78.56**	62.66	69.64	65.26	**90.77**	79.59
	SVM	74.90	**66.04**	**70.18**	**67.62**	90.52	**80.79**
John Snibert	LR	81.92	80.87	81.36	81.06	82.44	82.39
	SVM	**82.13**	**81.96**	**82.02**	**81.98**	**82.97**	**82.93**

Table 5.9: Sensitivity classifier results (upper Holly Palmer, lower John Snibert) using 5-fold cross validation.

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, and the ranking problem can be seen as a regression problem.

$$L(F(x), y) = \sum_{i}^{m} (f(x_i) - y_i))^2 \tag{5.1}$$

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

LambdaMART [121] is a pairwise approach which is based on Multiple Additive Regression Trees (MART) [46] 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.

$$l(f; x_i, x_j, y_{ij}) = -\overline{P}_{ij} \log P_{ij}(f) - (1 - \overline{P}_{ij}) \log (1 - P_{ij}(f)) \tag{5.2}$$

where 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 some IR measure (e.g., Δ NDCG) as in LambdaRank [17]. In our experiments, we have a held-out set of search topics that serve as a validation set, which is used to determine the ensemble of trees with the best score on the evaluation measure.

AdaRank [124] 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 rankers. 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.

$$L(F(x), y) = \sum_{i}^{m} \exp\{-E(\pi(q_i, d_i, f)), y_i\} \tag{5.3}$$

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

Coordinate Ascent [75] is another listwise approach which is iteratively optimizing a multivariate objective function by performing 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.

5.3 Integration of Sensitivity Classifiers and Rankers

When searching among content that is not sensitive, it suffices to learn an effective ranking function. For searching 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 modify a search engine results so that they take the predicted sensitivity of each document into account. Basically, our setup is composed of the following components.

1. A ranking 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. The ranking model could be as simple as a retrieval function (e.g., BM25) or a model trained on ranking documents (e.g., LtR).

2. A sensitivity classifier that takes a document as input, and outputs the document's probability of sensitivity.

3. A sensitivity filter that excludes documents if they are predicted to be sensitive.

5.3.1 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, a ranking model is trained using all documents without considering their sensitivities, as shown in Figure 5.6(a). During test 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, during training time, documents are fed to a filter along with their sensitivity labels, as shown in Figure 5.6(b). Then the filter omits any document that is truly sensitive. After that, the filtered documents are used to train the ranking model. During test time, documents are first filtered based on their predicted sensitivities. Then, they are fed to the resulting ranking 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 Baseline 1, the ranking model is trained using all documents. During test 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 predicted as sensitive are filtered out and the remaining top k documents are returned, as shown in Figure 5.6(c). This baseline has more training documents than Baseline 2.

Baseline 4: Demoting relevance scores for sensitive documents. Inspired by Dong et al. [39, 40], we demote the relevance scores of a query-document pair to the minimum grade (relevance score = 0) when the document is truly sensitive. Then the modified query-document pairs are used to train the ranking model. This is to train the ranking model so that it outputs low relevance scores for documents that are predicted to be sensitive. During test time, documents are fed to the resulting model along with a test query/topic, and then the model outputs a ranked list.

5.3.2 Jointly Modeling Relevance and Sensitivity

Some LtR algorithms allow us to produce models that optimize customizable loss functions. Those algorithms are called list-wise approaches. A common approach is to base the loss function on the evaluation measure (e.g., seeking to minimize 1-nDCG). Since we aim at balancing between relevance and sensitivity, we propose to use any of our evaluation measures as part of the loss function (e.g., 1 - nCS-DCG) to be minimized. In addition to modifying the loss function, we inject

the probability of a document being sensitive (and its complement) as part of its vector representation while building the listwise LtR model, as shown in Figure 5.6(e). At test time, all documents, augmented with their probabilities of sensitivity, are fed to the resulting model along with a test query/topic, and then the model outputs a ranked list. We refer to that approach as "Joint". An additional optional step is to apply a post-filter on the results to further hide any result that is predicted to be sensitive. We refer to that extension as "Joint+Post-filter" (plotted in dotted lines in Figure 5.6(e)).

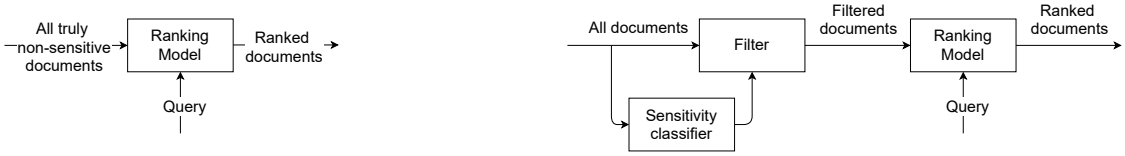
5.4 Results

By using the test collections described in Chapter 3, we evaluate the effectiveness of all our baselines and our proposed approach based on our evaluation measures, described in Chapter 4.

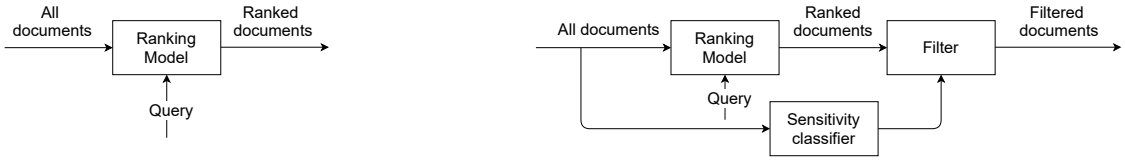
5.4.1 Experiment Design

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 true sensitivity labels in the test collections (accuracy = 100%). Then we investigate the performance of our baselines and our proposed approach using nDCG and TERN measures with a ranking cutoff k, after which results are not evaluated. 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

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

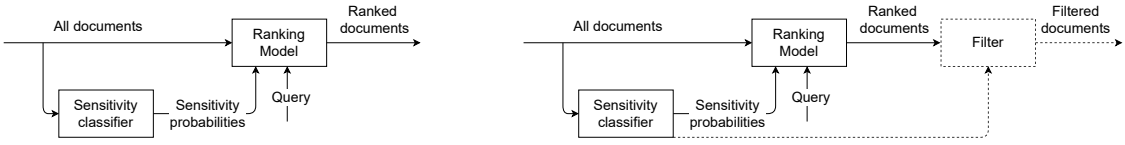


(b) Baseline 2: Pre-filtering then Relevance Only.



(c) Baseline 3: Relevance Only then Post-filtering.

(d) Baseline 4: Demoting scores for sensitive documents.



(e) Jointly modeling relevance and sensitivity with optional Post-filtering.

Figure 5.6: Alternative approaches for search among sensitive content. For each approach, we show how it works during training (left) and test (right) times.

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

Table 5.10: Gain matrix for documents depending on the relevance level

is used for test. So each fold contains ∼21 and 7 search topics in the OHSUMED and Avocado test collections, respectively.

Then, we experiment using an imperfect classifier that might be seen in practice. We use the LR classifier achieving $F_1 = 73.5$, 69.6, and 81.3 for the OHSUMED, Holly Palmer, and John Snibert test collections, respectively as shown in Tables 5.2 and 5.9. We therefore investigate the performance of our baselines and our proposed approach using nDCG, TERN, SENS, and nCS-DCG measures with a ranking cutoff k

We numerically interpret the relevance levels and compute the discount using the default approach of Ranklib as shown in equation (4.3).

$$g_i = 2^{rel} - 1, d_i = \frac{1}{\log(i+2)} \tag{5.4}$$

where i is the rank and rel is relevance score. In both OHSUMED and Avocado test collections, 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 5.10.

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

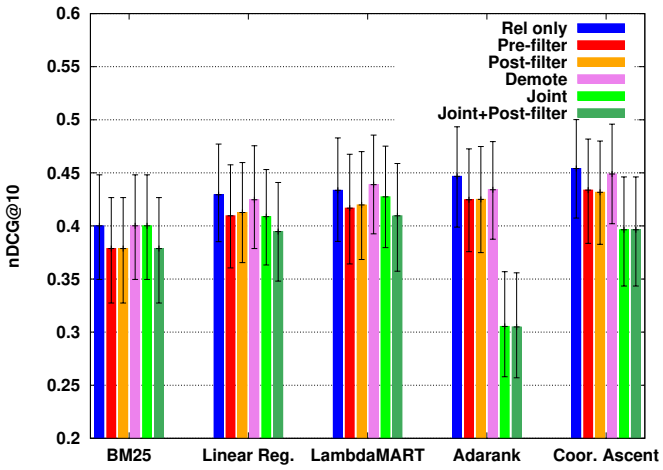
	Sensitive	Not sensitive
Retrieved	C_s	0
Not retrieved	0	0

Table 5.11: Cost matrix for documents depending on the sensitivity level

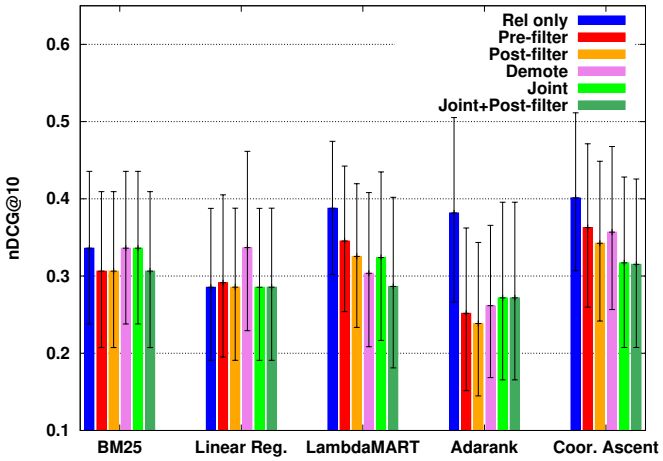
such cases). We therefore apply different sensitivity penalties (M = 0, 1, 3) for both TERN and SENS measure to correspond to little, moderate, and severe penalties for showing sensitive documents. We create these different settings to show the effectiveness of the proposed approaches under different test scenarios. For CS-DCG and nCS-DCG, we 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 Tables 3.1 and 3.6 show, on average a query has more than 10 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 5.11. Due to the design of the test collections, 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.

5.4.2 Oracle Upper Bounds

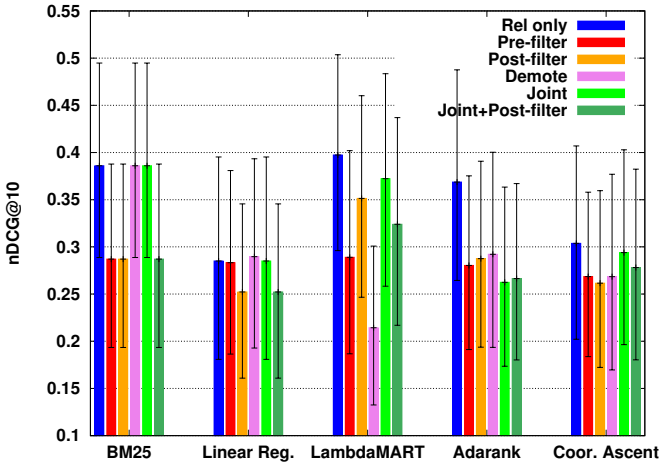
Under the ideal conditions when we have a perfect sensitivity classifier (accuracy = 100%), we expect that no sensitive documents would appear in any ranked result list. Indeed, that's what we see.



(a) OHSUMED

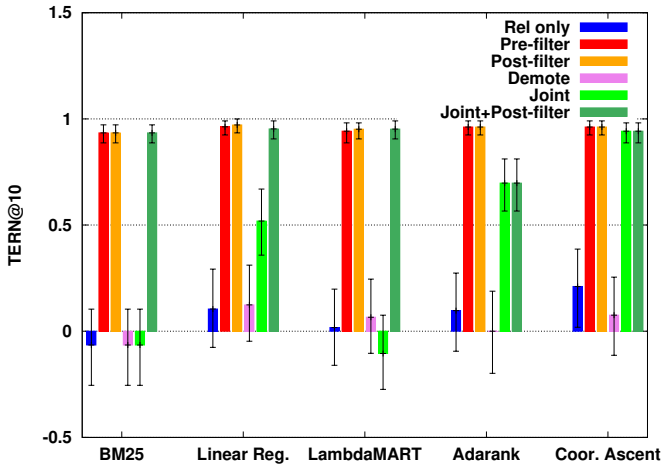


(b) Holly Palmer

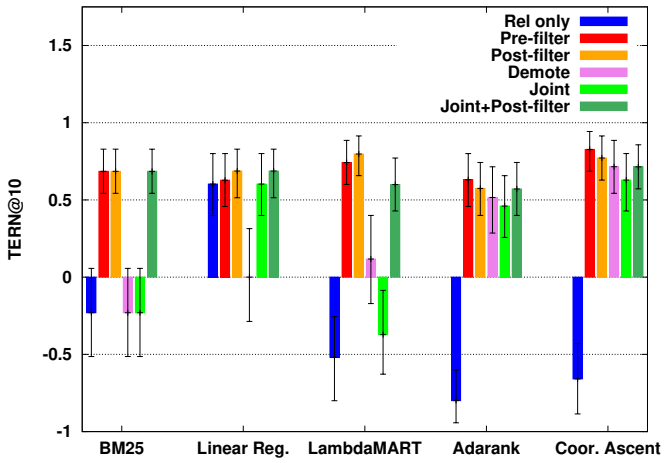


(c) John Snibert

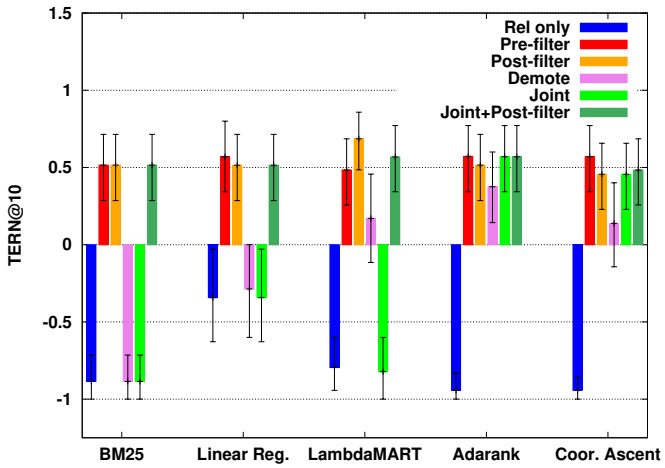
Figure 5.7: nDCG@10 of different approaches with a perfect sensitivity classifier for each test collection. Error bars are the 95% confidence interval.



(a) OHSUMED



(b) Holly Palmer



(c) John Snibert

Figure 5.8: TERN@10 (M=1) of different approaches with a perfect sensitivity classifier for each test collection. Error bars are the 95% confidence interval.

We evaluate our baselines and our proposed system using nDCG@10 and TERN@10 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 5.7 shows, that's what we see.

Moreover, as we would expect, Figure 5.8 shows that pre-filtering or post-filtering outperforms other approaches. This is because with a perfect classifier, there are 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).

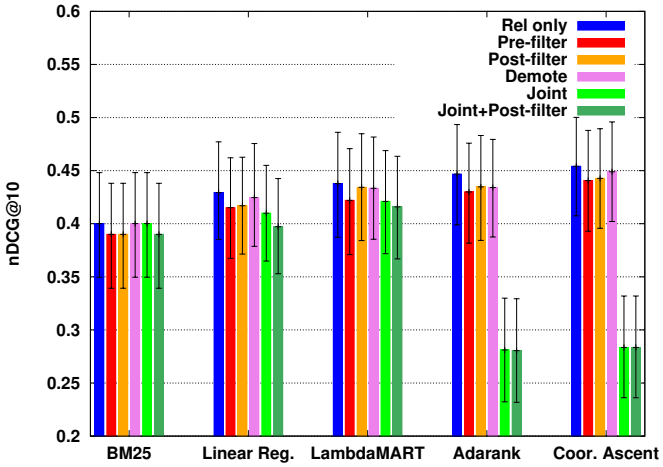
5.4.3 Integration with Imperfect Sensitivity Classifier

In this part of our experiments, we investigate how different approaches perform if the sensitivity classifier results have some misclassifications, as would be expected in practice. Obviously, as shown in Figure 5.9, we expect lower nDCG@10 scores for those approaches which rely on filtering or penalizing to discourage showing sensitive documents, since some sensitive documents could be relevant. We also note that training a ranking model on TERN@10 scores should decrease the resulting nDCG@10 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@10 in every case.

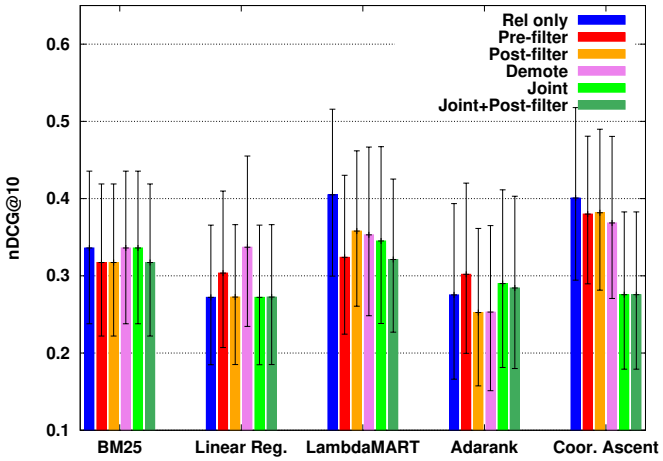
Turning now to evaluation using our new evaluation measure (TERN@10), we see a quite different picture. As shown in Figure 5.10, we get lower scores for baseline 1 as expected, since it does not take document sensitivity into account in any way. Additionally, prefiltering and postfiltering no longer have the best scores (as they did when evaluated using TERN@10 in Figure 5.8) because they rely on an imperfect classifier. 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 5.8 and 5.10 shows, TERN@10 results drop for all approaches as misclassification rates increase. Because the cost of showing a sensitive document is relatively high (M=1), false negatives are more important than false positives. Unsurprisingly, it is therefore the false negatives that are responsible for this drop in absolute scores.

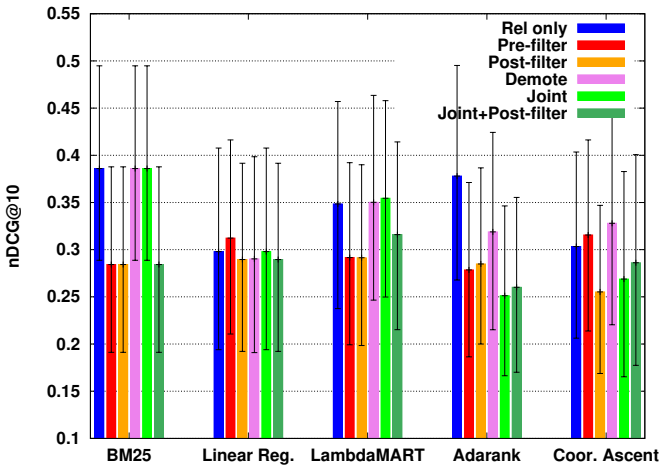
As shown in Figure 5.10, listwise approaches trained with TERN have a higher TERN@10 scores than other baselines. For AdaRank, our approach is marginally better than the baselines in the John Snibert test collection, although none of the differences are statistically significant. For Coordinate Ascent, it consistently outperforms other baselines in all test collections. Our results are statistically significantly better than our baselines by a two-tailed paired sign test ($p < 0.05$) in both Holly Palmer and John Snibert test collections.



(a) OHSUMED

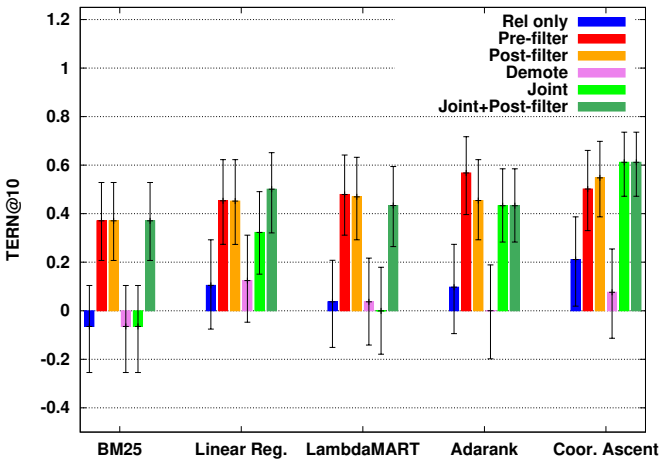


(b) Holly Palmer

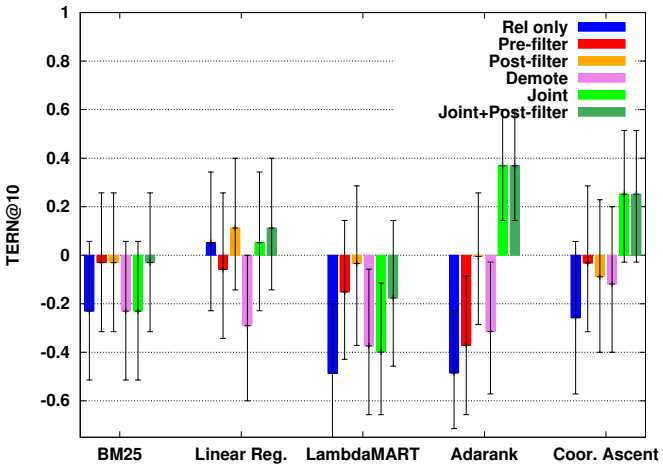


(c) John Snibert

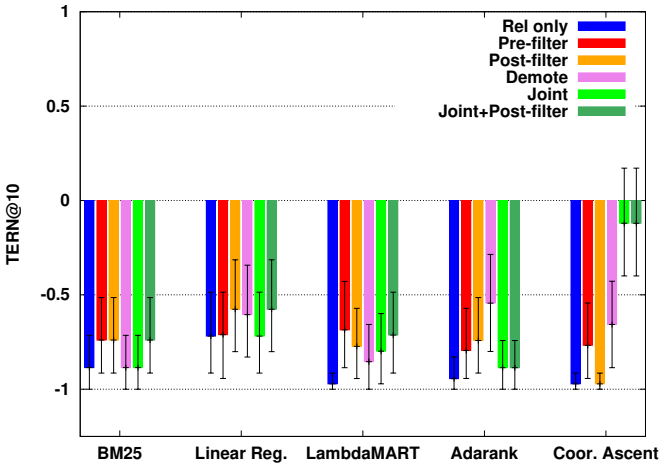
Figure 5.9: nDCG@10 of different approaches with an LR sensitivity classifier achieving $F_1 = 73.5$, 69.6, and 81.3 in the OHSUMED, Holly Palmer, and John Snibert test collections, respectively. Error bars are the 95% confidence interval.



(a) OHSUMED



(b) Holly Palmer



(c) John Snibert

Figure 5.10: TERN@10 (M=1) of different approaches with an LR sensitivity classifier achieving $F_1 = 73.5$, 69.6, and 81.3 in the OHSUMED, Holly Palmer, and John Snibert test collections, respectively. Error bars are the 95% confidence interval.

5.4.4 SENS Comparison

In the previous experiments, we used the TERN measure to evaluate the effectiveness of different systems. In this set of experiments, we use the SENSE measure as another evaluation measure to verify the effectiveness of the joint approach. In addition to non-trained retrieval function (BM25), we use the Coordinate Ascent algorithm, as it outperforms AdaRank in several test collections, and it does not require the optimization measure to be bounded (i.e., [-1,1] as in AdaRank). The Coordinate Ascent algorithm is used for building listwise LtR ranking models optimized towards a relevance only measure (nDCG@10) and models optimized towards the SENS@10 scores that jointly model relevance and sensitivity. For each case, the ranking model's output is treated in two ways: 1) no post-filtering, or 2) post-filtering documents which are predicted to be sensitive. By following this structure, we adopt baselines #1, #3, joint, and joint+post-filter approaches, which showed promising results in the previous experiments (Section 5.4.3).

After that, we compare the effectiveness of the proposed approaches using the SENS evaluation measure with ranking cutoff (@10). We apply different sensitivity penalties (M = 0, 1, 3) to correspond to little, moderate, and severe penalties for showing sensitive documents. We create these different settings to show the effectiveness of the proposed approaches under different test scenarios, as shown in Table 5.12.

Two main observations stand out from this analysis. First, We note that approaches relying on listwise LtR optimizing towards SENS@10 score the highest in

Approach	OHSUMED		
	M=0	M=1	M=3
(1a) BM25	0.188▾	-0.331▾	-1.368▾
(1b) BM25 + Filter	0.267▾	-0.025▾	-0.610▾
(2a) LtR (nDCG@10)	0.284▾	-0.096	-0.869
(2b) LtR (nDCG@10) + Filter	**0.365**	0.148▾	-0.286▾
(3a) LtR Joint (SENS@10, M=0)	0.346	0.139▾	-0.276▾
(3b) LtR Joint (SENS@10, M=0) + Filter	0.346	0.148▾	-0.248▾
(4a) LtR Joint (SENS@10, M=1)	0.312▾	0.133	-0.225
(4b) LtR Joint (SENS@10, M=1) + Filter	0.312▾	0.133	-0.225
(5a) LtR Joint (SENS@10, M=3)	0.297▾	**0.183**	**-0.043**
(5b) LtR Joint (SENS@10, M=3) + Filter	0.297▾	**0.183**	**-0.043**

Approach	Holly Palmer		
	M=0	M=1	M=3
(1a) BM25	0.162▾	-0.381▾	-1.467▾
(1b) BM25 + Filter	0.200	-0.228▾	-1.086▾
(2a) LtR (nDCG@10)	0.171▾	-0.429▾	-1.629▾
(2b) LtR (nDCG@10) + Filter	0.199	-0.315▾	-1.344▾
(3a) LtR Joint (SENS@10, M=0)	0.219	-0.124	-0.809
(3b) LtR Joint (SENS@10, M=0) + Filter	0.219	-0.067	-0.638
(4a) LtR Joint (SENS@10, M=1)	**0.263**	**0.121**	**-0.165**
(4b) LtR Joint (SENS@10, M=1) + Filter	**0.263**	**0.121**	**-0.165**
(5a) LtR Joint (SENS@10, M=3)	0.260	0.060	-0.340
(5b) LtR Joint (SENS@10, M=3) + Filter	0.260	0.060	-0.340

Approach	John Snibert		
	M=0	M=1	M=3
(1a) BM25	0.029▾	-0.914▾	-2.800▾
(1b) BM25 + Filter	0.047▾	-0.781▾	-2.439▾
(2a) LtR (nDCG@10)	0.000▾	-0.971▾	-2.914▾
(2b) LtR (nDCG@10) + Filter	0.000▾	-0.918▾	-2.804▾
(3a) LtR Joint (SENS@10, M=0)	0.078▾	-0.522▾	-1.722▾
(3b) LtR Joint (SENS@10, M=0) + Filter	0.078▾	-0.522▾	-1.722▾
(4a) LtR Joint (SENS@10, M=1)	0.137	-0.377	-1.406
(4b) LtR Joint (SENS@10, M=1) + Filter	0.146	-0.340	-1.311
(5a) LtR Joint (SENS@10, M=3)	0.175	-0.311	-1.282
(5b) LtR Joint (SENS@10, M=3) + Filter	**0.184**	**-0.273**	**-1.187**

Table 5.12: SENS@10 of the proposed approaches. Each row block represents the performance of one ranking model when a post-filter is not ('a') or is ('b') used. ▾ denotes a significant difference by a two-tailed Wilcoxon signed rank test [49] ($p < 0.05$) when compared with the best performing approach in the same column (marked in bold).

different test scenarios, and they statistically significantly beat the baselines that rely on filtering, e.g. approaches 1b and 2b. This observation holds except only for one case (OHSUMED with penalty M = 0; in that case, approach 2b which applies a filter after a listwise LtR model optimized towards nDCG@10 has the highest SENS@10 score). This comports with the results from the previous section, showing that listwise LtR models can be optimized towards the optimization measure, outperforming models that rely on filtering. In these experiments, by contrast, we used the SENS@10 for both optimization and evaluation.

Second, as expected, any approach that does not rely on filtering (with suffix 'a') scores less than or equal to the corresponding approach where results are filtered (with suffix 'b'). This is because the former approach does not take sensitivity into account. However, it is interesting to notice that filtering does not score higher when it is deployed after a listwise LtR model optimized towards the SENS@10 measure. For example, both approaches 4a and 4b score equally on the Holly Palmer test collection, which illustrates the flexibility of listwise LtR in learning how to jointly optimize for relevance and sensitivity.[2] This observation does not hold when M=1 (during evaluation) in the OHSUMED test collection, when we compare between approaches 3a and 3b (where M=0 during optimization). This is because when the SENS@10 score is 0, it is hard for the ranker to determine which action is needed to improve the score: 1) add more documents that are predicted to be relevant, or 2) hide some documents predicted to be sensitive. When M > 0, it is clearer which

[2]We inspected the output ranked lists from these approaches, and they are exactly the same. In this case, filtering does not hide any additional sensitive results.

action to learn (e.g., to hide some documents when SENS score = -M). To summarize this observation, applying a post-filter to our proposed approach (Figure 5.6(e)) does not degrade the effectiveness measured by SENS@10, but in most cases it has no effect.

5.4.5 nCS-DCG Comparison

In this section, we show the effectiveness of the joint approach by evaluating the results using nCS-DCG@10 (C_s=12). This is to provide a complementary view and verify the superiority of our proposed approach. As seen in Table 5.13, the joint approach, whether a post-filter is used or not (approaches 6a and 6b), outperforms other straightforward approaches that rely on filtering. We noticed that the difference is statistically significant using a two-tailed paired t-test [36] ($p < 0.05$) in all test collections, except only for one case where approach 6a was marginally better than approach 2b in the OHSUMED test collection.

Also, we noticed that applying a filter on the results of the joint approach helped in hiding additional sensitive results, and hence the score increased in the John Snibert test collection. In other test collections, the post-filter had no effect on the result lists.

Approach	OHSUMED nCS-DCG@10(C_s=12)
(1a) BM25	0.729▾
(1b) BM25 + Filter	0.801▾
(2a) LtR (nDCG@10)	0.773▾
(2b) LtR (nDCG@10) + Filter	0.8284
(6a) LtR Joint (nCS-DCG@10, C_s=12)	**0.850**
(6b) LtR Joint (nCS-DCG@10, C_s=12) + Filter	**0.850**
Approach	Holly Palmer nCS-DCG@10(C_s=12)
(1a) BM25	0.747▾
(1b) BM25 + Filter	0.825▾
(2a) LtR (nDCG@10)	0.682▾
(2b) LtR (nDCG@10) + Filter	0.811▾
(6a) LtR Joint (nCS-DCG@10, C_s=12)	**0.892**
(6b) LtR Joint (nCS-DCG@10, C_s=12) + Filter	**0.892**
Approach	John Snibert nCS-DCG@10(C_s=12)
(1a) BM25	0.540▾
(1b) BM25 + Filter	0.742▾
(2a) LtR (nDCG@10)	0.488▾
(2b) LtR (nDCG@10) + Filter	0.740▾
(6a) LtR Joint (nCS-DCG@10, C_s=12)	0.862
(6b) LtR Joint (nCS-DCG@10, C_s=12) + Filter	**0.870**

Table 5.13: nCS-DCG@10 (C_s=12) of the proposed approaches. Each row block represents the performance of one ranking model when a post-filter is not ('a') or is ('b') used. ▾ denotes a significant difference by a two-tailed t-test [36] ($p < 0.05$) when compared with the best performing approach in the same column (marked in bold).

5.5 Effect of Sensitivity Classification on Search and Protection Engines

In our initial results, we built a Logistic Regression classifier for each of the test collections. Then, we showed the effectiveness of the proposed approach at a given level of misclassification error. The key question we raise in this section is which approach is best under different classifier accuracies. We build three models for classifying sensitivity: Logistic Regression, DistilBERT, and an **or** combination of the two. Then, we measure the effectiveness of these sensitivity classification models when integrated inside search and protection engines.

5.5.1 Selection of Sensitivity Classification Models

Our logistic regression classifiers were built with sklearn's Logistic Regression library [43]. The logistic regression model was trained on the union of the title and abstract for each document in the OHSUMED test collection, and on the union of the subject, body, and attachments for the Avocado collection. Our neural classifier was built with huggingface's DistilBERT, a pre-trained classification model trained on a large collection of English data in a self-supervised fashion [102]. DistilBERT is a distilled version of BERT large, and we choose it because it runs 60% faster than BERT large [37] while still retaining over 95% of its effectiveness. For the OHSUMED collection, fine-tuning of DistilBERT for this classification task was performed using the training set. Many email messages have more text in the union

of their subject, body and attachments than DistilBERT's 512-token limit, so for Avocado we divided the text of each item into 500-token passages with a 220-token stride. For fine-tuning each of the 5 Avocado classifiers for this task on the 5 training folds, we considered a passage to be sensitive if the document from which that passage had been extracted was marked as sensitive; for testing, we considered a document to be sensitive if any passage in that document was classified as sensitive, and the probability of sensitivity for a document to be the maximum sensitivity probability for any passage in that document.

DistilBERT probabilities can benefit from calibration, so we used the validation sets to perform calibration as follows. We first binned the sensitivity probability estimates of the classifier on the validation set into 10 uniform partitions (0-10%, 10%-20%, ..., 90%-100%). The true fraction of truly sensitive documents in each partition was computed using ground truth validation set annotations. We then found the best fit line that defined an affine function to transform system estimates to ground truth values with minimum Mean Square Error (MSE) over those 10 points. At test time, this affine function was then used to transform every DistilBERT sensitivity probability estimate to better approximate the true sensitivity probability. To assign a binary (yes/no) value for sensitivity to each test document, we learned one threshold for each classifier on the probability estimates by using a grid search in the range [0, 1] with step size 0.01, and then selected the threshold with the highest F_1 on the validation set.

Our third model, a disjunctive combination of our logistic regression and DistilBERT models, used the **or** function between the decisions of the DistilBERT and

logistic regression models. For example, if Logistic Regression identified an Avocado email message as sensitive but DistilBERT classified it as not sensitive, the combined model would declare it as sensitive. For our extrinsic evaluation below we require not a decision but a probability; in that case we calculate the combined probability using an independence assumption (i.e., the probability that a document is not sensitive is the product of the logistic regression and calibrated DistilBERT probabilities that the document is not sensitive).

5.5.2 Train/Validation/Test Splits

While building sensitivity classification models, we used the set of OHSUMED documents judged for relevance for test, while keeping the rest for training. The training data is further partitioned into training and validation by an 85/15 split. For the Avocado collection, we first segregated the sensitivity judgments for the Holly Palmer persona and for the John Snibert persona. For each persona, we then built and tested classifiers using five-fold cross-validation, iteratively training on some four folds and testing on the fifth. We took 20% of the data from each training fold and created a validation subfold for each training fold.

While building search and protection engines, we followed the same train/test split as describe in Section 5.4.1. Basically, search topics are split into 5 folds, three of which are used for training, one is used for validation, and one for test.

5.5.3 Intrinsic Evaluation

Table 5.14 reports four intrinsic measures of classification effectiveness: precision, recall, F_1, and F_2. Our experiment showed that the DistilBERT classifier has the best F_1 score on the OHSUMED test collection, logistic regression has the best F_1 for John Snibert, and the combined model has the best F_1 for Holly Palmer. The reason for DistilBERT excelling on OHSUMED for F_1 is likely related to the number of training samples (>250k). Neural methods tend to perform better with more data, and the Avocado collection only contains around 2,000 training samples. Another possible factor is that DistilBERT was fine-tuned on Avocado with weak supervision by having to segment each email message and assign a weak label to each segment. So if a message is labeled as sensitive, then all its segments are labeled as sensitive even if there are some of them are not sensitive. We adopt this approach as it is often used during fine-tuning BERT models for relevance [34]. This approach may have led to worse results.

The combined model performed the best by F_2 for all three test collections. F_2 emphasizes recall, and as expected, the combined model yielded the best recall in every case. From the results, it can be seen that choosing the best classifier relies heavily on the test collection for maximizing F_1, but the combined classifier is the best for maximizing F_2 scores.

Classifier	OHSUMED Precision↑	Recall↑	F_1↑	F_2↑	Accuracy↑
(a) LR	76.72	73.29	74.96	73.95	94.01
(b) DistilBERT	**82.75**	80.08	**81.39**	80.60	**95.52**^{a,c}
(c) Combined	74.61	**83.81**	78.94	**81.8**	94.53^{a}
Classifier	Avocado: Holly Palmer Precision↑	Recall↑	F_1↑	F_2↑	Accuracy↑
(a) LR	**72.29**	69.98	71.12	70.43	**90.34**^{b,c}
(b) DistilBERT	66.20	67.85	67.02	67.52	88.65
(c) Combined	64.15	**80.11**	**71.25**	**76.31**	89.02
Classifier	Avocado: John Snibert Precision↑	Recall↑	F_1↑	F_2↑	Accuracy↑
(a) LR	**80.53**	84.85	**82.63**	83.95	**83.06**^{b,c}
(b) DistilBERT	72.87	87.00	79.31	83.75	78.44
(c) Combined	70.86	**93.73**	80.71	**88.05**	78.72

Table 5.14: Intrinsic sensitivity classification results (percent). Superscripts indicate statistically significant improvement over that system by McNemar's test [74] ($p < 0.05$) for comparing accuracy of two models.

5.5.4 Extrinsic Evaluation

In this section, we study the effect of sensitivity classification on search among sensitive content. In Section 5.3, we proposed several approaches for combining a ranking model and an automatic sensitivity classifier. Among these approaches, we have selected two for comparison here. First, The post-filter approach works by applying the sensitivity classifier on the ranking model's output as a filter, so that any result that is predicted to be sensitive is removed the result list. We build ranking models using the Coordinate Ascent ranking algorithm [75] optimizing for normalized Discounted Cumulative Gain (nDCG@10). For evaluation, we used TERN@10 with penalty M=1.

Second, the joint approach works by having the ranking model optimized for

a measure that balances between relevance and sensitivity. This can be achieved by leveraging listwise learning to rank (LtR) techniques. In our experiments, we used the Coordinate Ascent listwise LtR algorithm, which outperforms other alternatives on these test collections as shown in Section 5.4. We used TERN@10 with penalty M=1 for both training and evaluating models in this approach.

As shown in Table 5.15, we compare the post-filter and joint approaches using the classification models described in Section 5.5.3. Additionally, we add a perfect oracle classifier to show the effect of sensitivity classification on the TERN measure. Several observations stand out of this experiment. First, there is a significant gap in effectiveness when a perfect oracle classifier is substituted for our classification models. This indicates that sensitivity classification plays an important role in the task.

Second, investigating the effectiveness of the post-filter approach using different classifiers, we see that that on these test collections the TERN@10 score is positively correlated with F_2. Specifically, the combined classifier outperforms other classifiers by both F_2 and TERN@10. We justify this observation because the combined classifier has a higher recall than other models, and hence it is able to detect more sensitive documents. As a result, the combined classifier is more effective, when deployed inside a search and protection engines, in filtering out documents that are predicted to be sensitive. Even though the combined classifier is less precise than LR and DistilBERT (shown in Table 5.14), it achieved the best TERN@10 score in the post-filter approach.

Third, we see that for all three of our classifiers the joint model yields bet-

ter results than the postfilter model, which is consistent with our results when we compare different approaches using nCS-DCG@10 and SENSE@10 measures as presented in Sections 5.4.3 and 5.4.4, respectively. Furthermore, in two test collections, the classifier the joint model does best with by TERN@10 is the one that it did best with by F_1. Moreover, with the Holly Palmer test collection where LR has the highest TERN@10 score, the F_1 score (F_1=71.12) is very nearly the best. Lastly, we see that if an oracle classifier were available, nothing could beat the post-filter approach, since it never reveals a sensitive document, and it never unnecessarily withholds a document that is not truly sensitive. This result conforms to the results presented Section 5.4.2 where different approaches were compared using TERN@10. So although the joint model is our best current design for a search and protection engine, that conclusion does depend on the effectiveness of the sensitivity classifier.

Classifier	OHSUMED		Holly Palmer		John Snibert	
	Post-filter	Joint	Post-filter	Joint	Post-filter	Joint
(a) LR	0.528	0.613	-0.229▾	**0.343**	-0.714	**0.086**
(b) DistilBERT	0.585	**0.717**	0.057	-0.029▾	-0.657	-0.029
(c) Combined	**0.642**	0.708	**0.143**	0.2	**-0.543**	-0.057
Oracle	0.943	0.934	0.629	0.629	0.457	0.457

Table 5.15: TERN@10 (M=1). Results are based on 5-fold cross validation on search topics. There are 106, 35, and 35 topics for the OHSUMED, Holly Palmer, and John Snibert test collections respectively. Except for the oracle classifier, the highest value in each column is marked in bold. ▾ denotes a significant difference by a two-tailed paired sign test ($p < 0.05$) when compared with the highest value.

5.6 Implementation

We used several ranking algorithms that are implemented in the Ranklib library, which is part of the Lemur project.[3] We added classes for our new evaluation measures (TERN, SENS, γCS-DCG, and γnCS-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`.

5.7 Chapter Summary

In this chapter, we show that at some levels of classifier accuracy that might be seen in practice (e.g., $F_1 = 73.5$, 69.6, and 81.3 for the OHSUMED, Holly Palmer, and John Snibert test collections, respectively), 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 showed that when the classifier accuracy is high, filtering has the best performance among all 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 our evaluation measures 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 the baselines.

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

Chapter 6: Extensions[1]

In the previous chapter, we compared between different approaches for search and protection engines. Results showed that the joint approach achieves the best results according to our evaluation measures. In this chapter, we present four extensions for enhancing the results presented so far. Since LtR transforms the ranking problem into a learning problem, we study the effect of changing two components of the learning problem: 1) optimization function, and 2) representation. First, we study the effect of changing the optimization function the ranking model optimizes for. As previously described, the joint approach works by setting the optimization function to the evaluation measure, e.g. 1 - nCS-DCG. In Section 6.1, we separate the evaluation measure, and use different measures to optimize for. The key insight is that some measures are good for evaluation, but not for optimizing ranking models. In Section 6.2, we study the effect of changing the representation of query-document pairs. This is achieved by using transformer-based rankers instead of using hand-crafted features.

Our last two extensions can be grouped, as they can be seen as post-processing steps after a search and protection system produces a result list. First, we introduce

[1]Some parts of this chapter were taken from a paper in preparation [105] and a publication [106] by Sayed and Oard

an extension that relies on clustering results, and swapping documents with others that are less sensitive but in the same cluster(Section 6.3). Finally, we study the effect of varying the search depth, and present two cutoff selection policies that learn how to obtain a cutoff where the score is maximized. (Section 6.4.2).

6.1 Optimization vs. Evaluation Measures

For the results in Tables 5.12 we used the SENS measure for both training ranking models and evaluating them. In this section, we focus on building ranking models that are optimized using measures different from the one used for evaluation.

From the evaluation perspective, as seen in Chapter 4, TERN and SENS are better than nγCS-DCG in terms of understandability, and in particular the ability to infer the relevance and sensitivity contributions. For instance, we can compare two systems either by using the mean score across queries, or by using the score per query for further investigation. Figure 6.1 compares between approach 2b which is optimized towards nCS-DCG coupled with a sensitivity post-filter against the best approach for the Holly Palmer test collection (approach 4a). It is obvious that approach 2b has more topics showing sensitive results (i.e., SENS@10 = -1). For example, with the Holly Palmer test collection, approach 2b gets a negative score for 18 topics, while there are only 5 topics with negative scores when approach 4a is used. The TERN measure shares with SENS the same properties, except that positive scores are all 1. As a result, the TERN measure can be considered as a simple statistic for the number of queries with positive, zero, and negative scores.

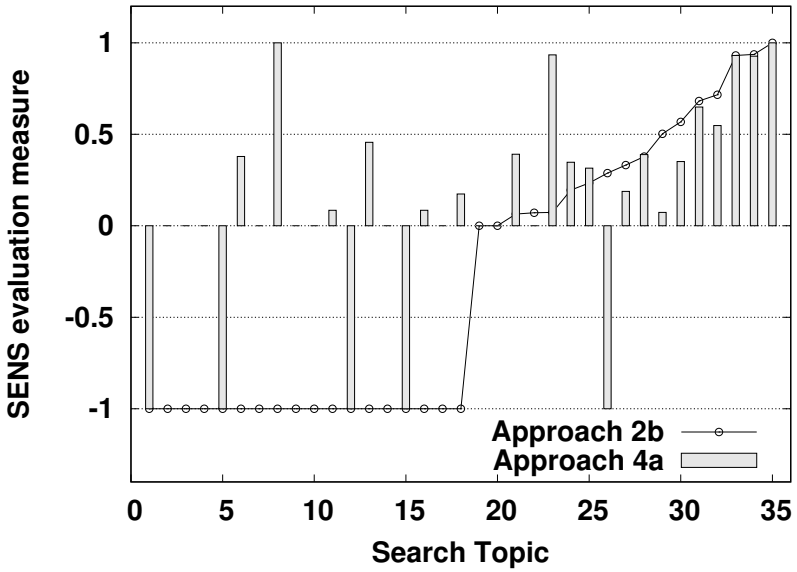


Figure 6.1: SENS of the best performing approach (4a, plotted as bars) for Holly Palmer against a baseline based on filtering (2b, plotted as a line). SENS penalty M = 1. Topics are sorted by baseline scores.

Optimization Measure	OHSUMED		
	M=0	M=1	M=3
(m1) TERN@10 (M=0)	0.764	0.623	0.340
(m2) TERN@10 (M=1)	0.736	0.613	0.368
(m3) TERN@10 (M=3)	0.755	0.651	0.443
(m4) SENS@10 (M=0)	0.755	0.547	0.132
(m5) SENS@10 (M=1)	0.736	0.557	0.198
(m6) SENS@10 (M=3)	0.745	0.632	0.406
(m7) CS-DCG@10 (c=12)	0.745	0.613	0.349
(m8) nγCS-DCG@10 (c=12, γ=1)	0.764	0.632	0.368
(m9) nγCS-DCG@10 (c=12, γ=0.5)	0.755	0.585	0.245
(m10) nγCS-DCG@10 (c=12, γ=0.1)	0.708	0.472	0.000
(m11) nγCS-DCG@10 (c=12, γ=0)	0.736	0.519	0.085
(m12) nγCS-DCG@10 (c=100, γ=1)	**0.792**	**0.679**	0.453
(m13) nγCS-DCG@10 (c=100, γ=0.5)	0.726	0.566	0.245
(m14) nγCS-DCG@10 (c=100, γ=0.1)	0.755	0.585	0.245
(m15) nγCS-DCG@10 (c=100, γ=0)	0.689	0.509	0.151
(m16) nγCS-DCG@10 (c=1000, γ=1)	0.736	0.632	0.425
(m17) nγCS-DCG@10 (c=1000, γ=0.5)	0.774	0.670	**0.462**
(m18) nγCS-DCG@10 (c=1000, γ=0.1)	0.783	0.670	0.443
(m19) nγCS-DCG@10 (c=1000, γ=0)	0.726	0.566	0.245

Table 6.1: TERN@10 for LtR Joint approach optimized towards different measures with different penalties. Highest value per column is bold.

Optimization Measure	Holly Palmer		
	M=0	M=1	M=3
(m1) TERN@10 (M=0)	0.486	0.257	-0.200
(m2) TERN@10 (M=1)	0.486	0.257	-0.200
(m3) TERN@10 (M=3)	0.514	0.400	0.171
(m4) SENS@10 (M=0)	0.429	0.086	-0.600
(m5) SENS@10 (M=1)	0.543	0.400	0.114
(m6) SENS@10 (M=3)	0.514	0.314	-0.086
(m7) CS-DCG@10 (c=12)	0.486	0.314	-0.029
(m8) nγCS-DCG@10 (c=12, γ=1)	**0.600**	**0.514**	**0.343**
(m9) nγCS-DCG@10 (c=12, γ=0.5)	0.486	0.314	-0.029
(m10) nγCS-DCG@10 (c=12, γ=0.1)	0.514	0.286	-0.171
(m11) nγCS-DCG@10 (c=12, γ=0)	0.514	0.286	-0.171
(m12) nγCS-DCG@10 (c=100, γ=1)	0.543	0.429	0.200
(m13) nγCS-DCG@10 (c=100, γ=0.5)	0.429	0.171	-0.343
(m14) nγCS-DCG@10 (c=100, γ=0.1)	0.571	0.429	0.143
(m15) nγCS-DCG@10 (c=100, γ=0)	0.457	0.229	-0.229
(m16) nγCS-DCG@10 (c=1000, γ=1)	0.543	0.429	0.200
(m17) nγCS-DCG@10 (c=1000, γ=0.5)	0.571	0.400	0.057
(m18) nγCS-DCG@10 (c=1000, γ=0.1)	0.486	0.229	-0.286
(m19) nγCS-DCG@10 (c=1000, γ=0)	0.543	0.429	0.200

Table 6.2: TERN@10 for LtR Joint approach optimized towards different measures with different penalties. Highest value per column is bold.

Optimization Measure	John Snibert		
	M=0	M=1	M=3
(m1) TERN@10 (M=0)	**0.371**	**0.000**	**-0.743**
(m2) TERN@10 (M=1)	0.314	-0.114	-0.971
(m3) TERN@10 (M=3)	0.257	-0.171	-1.029
(m4) SENS@10 (M=0)	0.200	-0.400	-1.600
(m5) SENS@10 (M=1)	0.229	-0.286	-1.314
(m6) SENS@10 (M=3)	0.286	-0.200	-1.171
(m7) CS-DCG@10 (c=12)	0.229	-0.229	-1.143
(m8) nγCS-DCG@10 (c=12, γ=1)	0.314	-0.057	-0.800
(m9) nγCS-DCG@10 (c=12, γ=0.5)	0.200	-0.314	-1.343
(m10) nγCS-DCG@10 (c=12, γ=0.1)	0.229	-0.314	-1.400
(m11) nγCS-DCG@10 (c=12, γ=0)	0.286	-0.257	-1.343
(m12) nγCS-DCG@10 (c=100, γ=1)	0.229	-0.229	-1.143
(m13) nγCS-DCG@10 (c=100, γ=0.5)	0.314	-0.086	-0.886
(m14) nγCS-DCG@10 (c=100, γ=0.1)	0.200	-0.314	-1.343
(m15) nγCS-DCG@10 (c=100, γ=0)	0.314	-0.114	-0.971
(m16) nγCS-DCG@10 (c=1000, γ=1)	0.229	-0.200	-1.057
(m17) nγCS-DCG@10 (c=1000, γ=0.5)	0.229	-0.200	-1.057
(m18) nγCS-DCG@10 (c=1000, γ=0.1)	0.257	-0.171	-1.029
(m19) nγCS-DCG@10 (c=1000, γ=0)	0.314	-0.114	-0.971

Table 6.3: TERN@10 for LtR Joint approach optimized towards different measures with different penalties. Highest value per column is bold.

From the optimization perspective, we build different models, where each is optimized towards a different measure. After that, we compare the effectiveness of the these models using the TERN measure with different sensitivity penalties (M = 0, 1, 3) and search cutoff equals to 10, as shown in Tables 6.1, 6.2, and 6.3. Similar patterns are observed when we use the SENS measure for evaluation. Results show that the optimization measure does not need to be the same as the evaluation measure. For example, models optimized for nγCS-DCG@10 outperform other models for the OHSUMED and John Snibert test collections. This is because TERN and SENS take widely spaced discrete values and thus introduce additional quantization noise. This observation does not hold for the Holly Palmer test collection, however, but we can see that model m13 still earns second place after model m1, which has the highest score. CS-DCG yields poor performance when used for optimization, probably because it is not bounded and hence will be biased towards queries with more relevant or sensitive results.

6.2 Transformer-Based Rankers

In our research, while learning to rank, each query-document pair is represented by a multi-dimensional feature vector, and each dimension of the vector is a feature that we expect to be information with regard to how relevant the document is with respect to the query. For the OHSUMED corpus, there are in total 45 features extracted from the fields of title and abstract, as shown in Table 3.9. For the Avocado test collections, there are in total 320 features extracted from the

fields of email's subject, body, and attachments as shown in Table 3.10. Examples include many of classical features used in IR, such as BM25 and language models scores. These features are often extracted by first building an index, and then extracting retrieval features for query-document pairs. Unfortunately, the diversity of features are limited to the capabilities the indexing tool offers. Perhaps, more features could be added by manual implementation, but that could be time-consuming and error-prone [48].

In this section, we study using transformer-based models for generating relevance features that could be useful for improving effectiveness. In particular, we study the use of monoBERT (an adapted version of BERT for text relevance) for generating relevance features for query-document pairs. As a result, we have a different set of relevance features, replacing the features in Tables 3.9 and 3.10. It is worth noting we still append the sensitivity features to our representation, mentioned in Tables 3.11 and 3.12.

Our objective is to study the effect of using transformer-based features on the effectiveness of search and protection engines. Our plan is to build models similar to those described in Table 5.12 but using transformer-based features. Then, we compare the effectiveness of the resulting models with models relying on handcrafted features, described in Section 5.4.4, using the SENS measure.

6.2.1 MonoBERT/MonoT5

We used the monoBERT model developed by Nogueira et al. [83] for ranking. MonoBERT is an adaption of BERT [37] to relevance classification. The input sequence for monoBERT is structured as follows.

[CLS] q [SEP] d [SEP]

where q represents the query tokens, and d represents the document tokens, [CLS] marks the beginning of the sequence, and [SEP] marks the end of an input (query or document). This model is fine-tuned to produce a probability of relevance by using the MS MARCO passage ranking test collection [29]. The 'mono' term refers to the fact that model is considered as pointwise learning to rank [66].

Additionally, we used the monoT5 model by Nogueira et al. [84] for ranking. The input sequence for monoT5 is structured as follows.

Query: q Document: d Relevant:

where this model is fine-tuned to produce the tokens "true" or "false" depending on whether the document is relevant to the query of not. In both the Holly Palmer and John Snibert test collections, documents were longer than the length limitation of BERT. As a result, documents are segmented into passages using a 10-sentence sliding window with a stride of 5 sentences. Then, similarly to [34], the

final document score is determined by taking the maximum passage score.

These two models could be used for ranking documents with respect to a given query. These models can be seen as examples to pointwise learning to rank models. As a result, filtering baselines can be used with these ranking models to hide any documents. In our experiments, we apply the post-filter approach by filtering out documents, that are predicted to be sensitive from the output of these models. We used PyGaggle while building monoBERT and monoT5 models.[2]

6.2.2 Listwise LtR on monoBERT Features

In this model, we extend monoBERT by extracting the embeddings of the [CLS] token, and then a learning to rank model is trained using these embeddings instead of the hand-crafted relevance features shown in Table 3.10. For text content that is longer than monoBERT's context length limit, we use the representation of the [CLS] token for the passage achieving the maximum relevance score. To this we concatenate features from Table 3.12 as potential indicators for sensitivity.

For building search and protection engines, we can leverage listwise learning to rank algorithms to jointly optimize for relevance and sensitivity. Additionally, filtering approaches can be applied as well. In our experiments, we train models optimizing towards a relevance measure (e.g., nDCG@10), and a measure that balances between relevance and sensitivity (e.g. SENS@10) with different penalties (M). For all the models, we optionally apply the post-filter approach to remove any document from the result list when the document is predicted to be sensitive.

[2] `https://github.com/castorini/pygaggle/`

Approach	OHSUMED		
	M=0	M=1	M=3
Best using hand-crafted features (Table 5.12)	0.365▾	0.183▾	-0.043▾
(7a) MonoBERT	0.312▾	-0.112▾	-0.961▾
(7b) MonoBERT + Filter	0.400	0.155	-0.336
(8a) MonoT5	0.290▾	-0.182▾	-1.125▾
(8b) MonoT5 + Filter	0.393	0.119	-0.428
(9a) MonoBERT + LtR (nDCG@10)	0.266▾	-0.244▾	-1.263▾
(9b) MonoBERT + LtR (nDCG@10) + Filter	0.377	0.094	-0.472
(10a) MonoBERT + LtR Joint (M=0)	0.409	0.192	-0.242
(10b) MonoBERT + LtR Joint (M=0) + Filter	**0.425**	0.236	-0.141▾
(11a) MonoBERT + LtR Joint (M=1)	0.399▾	**0.258**	-0.025▾
(11b) MonoBERT + LtR Joint (M=1) + Filter	0.399▾	**0.258**	-0.025▾
(12a) MonoBERT + LtR Joint (M=3)	0.346▾	0.232▾	**0.006**
(12b) MonoBERT + LtR Joint (M=3) + Filter	0.346▾	0.232▾	**0.006**

Table 6.4: SENS@10 of the proposed models on the OHSUMED test collection. Each row block represents the performance of one ranking model when a filter is not ('a') or is ('b') used. ▾ denotes a significant difference by a two-tailed Wilcoxon signed rank test [49] ($p < 0.05$) when compared with the best performing approach in the same column (marked in bold).

6.2.3 Results

Following the same experiment design as Section 5.4.4, we build models that rely on transformer-based features. In all test collections, we note that ranking models that are based on transformer features (e.g. monoBERT) perform better than models that are based on hand-crafted features, as shown in Tables 6.4, 6.5, and 6.6. For example, for the John Snibert test collection, approach 11a outperforms all approaches that rely on hand-crafted features (e.g., approaches 2a to 5b) in Table 5.12. This observation does not hold only for one case in the Holly Palmer test collection when the sensitivity penalty M=3 where approaches 12a and 12b score slightly lower than approaches 4a and 4b (Table 5.12). This observation suggests that there is no

Approach	Holly Palmer		
	M=0	M=1	M=3
Best using hand-crafted features (Table 5.12)	0.263	0.121	**-0.165**
(7a) MonoBERT	0.334	-0.095	-0.952
(7b) MonoBERT + Filter	**0.357**	-0.015	-0.758
(8a) MonoT5	0.265	-0.249▾	-1.278▾
(8b) MonoT5 + Filter	0.315	-0.114▾	-0.971▾
(9a) MonoBERT + LtR (nDCG@10)	0.283	-0.146▾	-1.003▾
(9b) MonoBERT + LtR (nDCG@10) + Filter	0.333	0.076	-0.438
(10a) MonoBERT + LtR Joint (M=0)	0.297	-0.074	-0.817
(10b) MonoBERT + LtR Joint (M=0) + Filter	0.297	-0.074	-0.817
(11a) MonoBERT + LtR Joint (M=1)	0.272▾	-0.014	-0.585
(11b) MonoBERT + LtR Joint (M=1) + Filter	0.272▾	-0.014	-0.585
(12a) MonoBERT + LtR Joint (M=3)	0.325	**0.125**	-0.275
(12b) MonoBERT + LtR Joint (M=3) + Filter	0.325	**0.125**	-0.275

Table 6.5: SENS@10 of the proposed models on the Holly Palmer test collection. Each row block represents the performance of one ranking model when a filter is not ('a') or is ('b') used. ▾ denotes a significant difference by a two-tailed Wilcoxon signed rank test [49] ($p < 0.05$) when compared with the best performing approach in the same column (marked in bold).

need to build an index for a collection of documents and manually select features to be extracted based on query-document pairs. Instead, a monoBERT model can be used as a feature generator by extracting the embeddings of the [CLS] token.

6.3 Cluster-Based Replacement

Looking at the performance of any approach in terms of TERN@10 (e.g., as shown in Figure 5.10), it is just one number, and that may not be enough to describe the performance across each search topic. For example, when applying Coordinate Ascent for the OHSUMED test collection, we count the number of queries with negative TERN@10 scores, and there are 41, 25, 23, 48, 13, and 13 queries for no filter, pre-filter, post-filter, demote, joint, joint+post-filter approaches, respectively.

Approach	John Snibert		
	M=0	M=1	M=3
Best using hand-crafted features (Table 5.12)	0.184	-0.273	-1.187
(7a) MonoBERT	0.093	-0.793▾	-2.565▾
(7b) MonoBERT + Filter	0.093	-0.736▾	-2.393▾
(8a) MonoT5	0.085	-0.772▾	-2.486▾
(8b) MonoT5 + Filter	0.128	-0.643▾	-2.186▾
(9a) MonoBERT + LtR (nDCG)	0.064▾	-0.822▾	-2.593▾
(9b) MonoBERT + LtR (nDCG) + Filter	0.104	-0.696▾	-2.296▾
(10a) MonoBERT + LtR Joint (M=0)	0.096	-0.618▾	-2.047▾
(10b) MonoBERT + LtR Joint (M=0) + Filter	0.123	-0.506▾	-1.763▾
(11a) MonoBERT + LtR Joint (M=1)	**0.155**	**-0.245**	**-1.045**
(11b) MonoBERT + LtR Joint (M=1) + Filter	**0.155**	**-0.245**	**-1.045**
(12a) MonoBERT + LtR Joint (M=3)	0.126	-0.446	-1.589
(12b) MonoBERT + LtR Joint (M=3) + Filter	0.126	-0.446	-1.589

Table 6.6: SENS@10 of the proposed models on the John Snibert test collection. Each row block represents the performance of one ranking model when a filter is not ('a') or is ('b') used. ▾ denotes a significant difference by a two-tailed Wilcoxon signed rank test [49] ($p < 0.05$) when compared with the best performing approach in the same column (marked in bold).

It is obvious that the joint approach, or when it is coupled with a post-filter, outperforms other approaches as it has the least number of queries that have at least one sensitive result among the top 10 ranks. However, there remains the question of whether we can further reduce the number of queries with negative TERN@10 scores.

One simple trick that we can use to limit the risk of false negatives (falsely predicting a sensitive document as not sensitive) 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

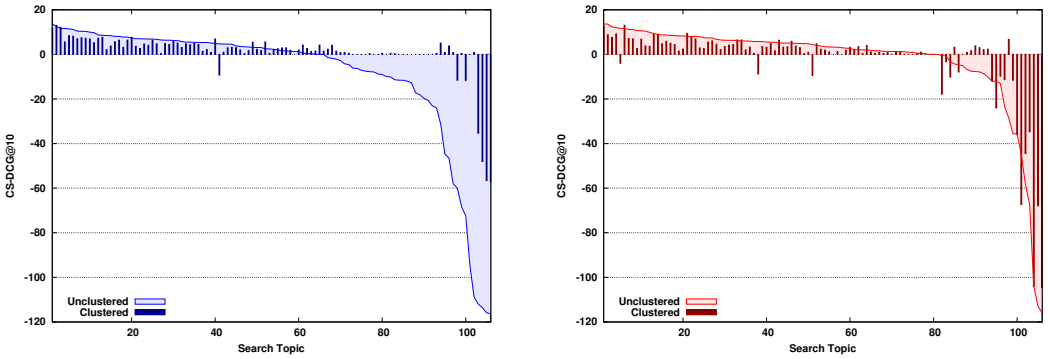
would get a sensitivity-averse analogue to diversity ranking.

We can apply this idea to any ranked list. Given the result list of a query, we first cluster all 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, where k is the rank cutoff. 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 was not at least in the same cluster as some document in the ranked list.

We used CLUTO software for clustering documents.[3] CLUTO 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).

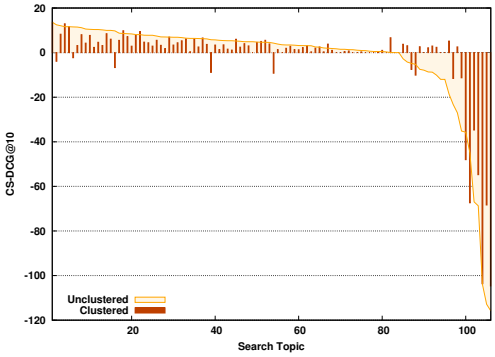
As Figure 6.2 shows, this cluster-based replacement strategy helps to limit the risk of getting a negative CS-DCG@10 score, and it achieves that result for

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

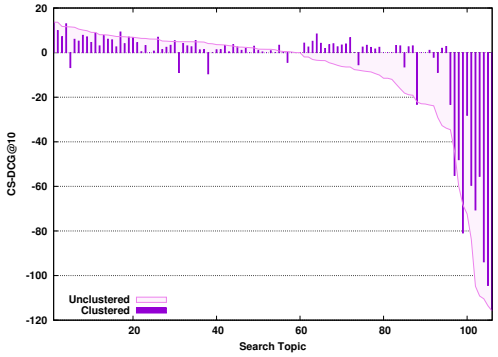


(a) Relevance only

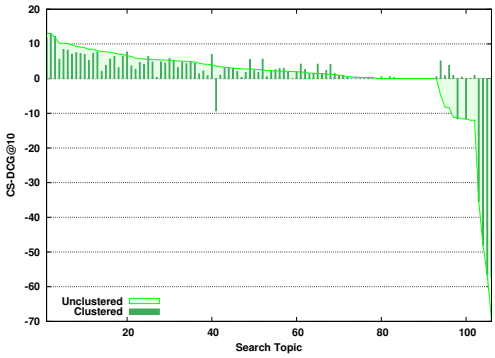
(b) Pre-filter



(c) Post-filter



(d) Demote



(e) Joint

Figure 6.2: CS-DCG@10 for Coordinate Ascent on the OHSUMED test collection with a sensitivity classifier F_1 of 73.5 with (bar) and without (filled area) cluster-based replacement.

every one of our approaches. We selected CS-DCG@10 when showing performance across each query to show the benefits of using cluster-based replacement strategy in reducing the number of sensitive results. In each case, the (generally) lower line that dips sharply at the right shows the CS-DCG@10 scores achieved by each query without cluster-baesd replacement, sorted from best to worst. The superiority of tuning Coordinate Ascent with a TERN@10 loss function is clearly indicated in this figure, since fewer queries yield poor CS-DCG@10 results. As the (generally) upper bars show, it is the hardest queries, those with the lowest CS-DCG@10, that see the greatest improvement from cluster-based replacement. As Table 6.7 shows, improvements in TERN@10 are evident as well for all approaches and test collections.

As shown in Table 6.7, we separately count the number of queries with positive, negative, or zero CS-DCG@10 score. Two main observations stand out from this analysis. First, It is obvious to notice that the number of queries with negative CS-DCG@10 scores decreased when the cluster-based replacement policy is applied. Second, there are more queries with zero score when the cluster-based replacement policy is applied. One possible scenario that contributes to that is when the replacement policy successfully removes all sensitive results of a hard query, and as a result possible relevant results are also replaced by not relevant documents, in which case there is no relevant or sensitive document among the top results. In that case, showing a ranked list with zero utility is better than showing a result list that has sensitive information. In Section 6.4.2, we try to mitigate this issue by showing fewer results (or even nothing) if the query is predicted to be hard.

OHSUMED				
Ranking Approach	unclustered		clustered	
	+/0/-	TERN@10	+/0/-	TERN@10
Relevance only	63/2/41	0.208	73/14/19	**0.509**
Pre-filter	78/3/25	0.5	76/11/19	**0.538**
Post-filter	81/2/23	0.547	75/15/16	**0.557**
Demote	56/2/48	0.075	72/14/20	**0.491**
Joint	78/15/13	0.613	81/18/7	**0.698**
Joint+Post-filter	78/15/13	0.613	81/18/7	**0.698**

Holly Palmer				
Ranking Approach	unclustered		clustered	
	+/0/-	TERN@10	+/0/-	TERN@10
Relevance only	12/2/21	-0.257	12/4/19	**-0.200**
Pre-filter	15/4/16	-0.029	15/4/16	-0.029
Post-filter	15/2/18	-0.086	16/4/15	**0.029**
Demote	13/5/17	-0.114	15/7/13	**0.057**
Joint	17/10/8	0.257	19/11/5	**0.400**
Joint+Post-filter	17/10/8	0.257	19/11/5	**0.400**

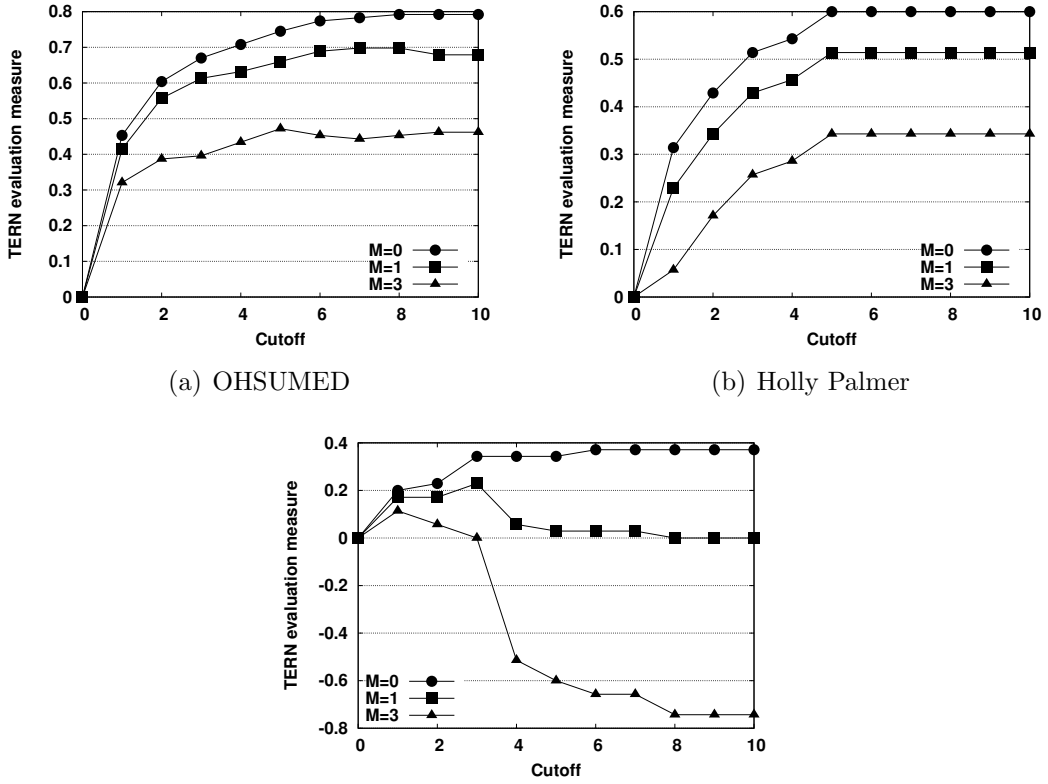
John Snibert				
Ranking Approach	unclustered		clustered	
	+/0/-	TERN@10	+/0/-	TERN@10
Relevance only	0/1/34	-0.971	1/2/32	**-0.886**
Pre-filter	3/2/30	**-0.771**	3/1/31	-0.800
Post-filter	0/1/34	-0.971	4/3/28	**-0.686**
Demote	5/2/28	-0.657	4/4/27	-0.657
Joint	11/9/15	-0.114	11/9/15	-0.114
Joint+Post-filter	11/9/15	-0.114	11/9/15	-0.114

Table 6.7: TERN@10 (M=1) comparison between different ranking approaches when Coordinate Ascent algorithm is used for each of our test collections. The joint approach is optimized towards TERN@10 (M=1). Number of search topics with positive/0/negative TERN@10 scores (+/0/-) is also displayed. For each pair, TERN@10 is averaged across five cross-validation folds when cluster-based replacement is not (left) or is (right) used. Symbol (*) denotes statistically significant difference according to a two-tailed paired sign-test ($p < 0.05$) over the corresponding clustered/unclustered score in the same approach.

To get a better idea of 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 TERN@10. We found that the number of sensitive documents added is less than the number of sensitive documents being removed. That is why there is an improvement when this replacement policy is used in the *Relevance Only* and *Demoting* approaches, as shown in Table 6.7. In approaches where filtering is used, and in the joint approach, there are fewer sensitive documents used for replacement, and the TERN@10 score increases as well.

6.4 Query-Specific Cutoffs

Looking at Tables 6.1 to 6.3, when a system has a positive TERN score, especially when $M > 0$, we can infer that this system successfully returns relevant content and protects sensitive results, on average. On the other hand, for the John Snibert test collection as shown in Table 6.3, all systems produce negative scores when the sensitivity penalty $M = 3$. This observation comes as a result to the higher number of sensitive documents found in that collection, as shown in Table 3.8. Here, the question we want to study is the effect of varying the search depth on the system score. Our intuition is that if a certain topic, or a test collection is known to have many sensitive results, it might be a good practice to run a strict system that does not show many results to lower the risk of showing sensitive results. When the search depth is decreased, the probability of encountering a relevant or sensitive



(a) OHSUMED

(b) Holly Palmer

(c) John Snibert

Figure 6.3: TERN of the best performing approach for each test collection under different cutoffs. Each cutoff is applied for all topics.

document is decreased, and vice versa. So it is appropriate to adjust the cutoff to maximize the value of the target evaluation measure.

In all the previous experiments, we fix the search depth to be 10. In this section, we investigate the performance of the best performing model, from Tables Table 6.1, 6.2, and 6.3, for each of the test collections separately, under different search cutoffs. We select models m12 for OHUSMED (M=0, M=1), m17 for OHSUMED (M=3), m8 for Holly Palmer under all penalties, and m1 for John Snibert under all penalties. As shown in Figure 6.3, we vary the search depth from 0 to 10 and report the value of the average TERN measure across topics. As the search depth

increases, except for the Holly Palmer test collection or when M = 0, we note that the TERN measure goes up until a certain point and then decreases after that. Our aim is to detect the cutoff which maximizes the value of the TERN measure. To do this, we describe two proposed cutoff policies, and then we present the results.

6.4.1 Policy #1: Learning a Single Cutoff

One way to fine-tune the search depth is to exhaustively try all possible cutoffs and select the one that maximizes the value of the TERN measure on a held-out set of topics. We call this policy "static cutoff selection" because the search depth is predetermined and does not vary by topic. In our experiments, we have a validation set of topics for each of our test collections. We therefore exhaustively try all cutoffs in the range [0,10] and pick the cutoff that achieves the best TERN score on this validation set.

6.4.2 Policy #2: Query Specific Cutoff

We propose another policy for cutoff selection that depends on the topic under test. The idea is that a small cutoff should be applied if a topic is known to have many sensitive results, and hence is hard to do well at. On the other hand, if a topic has few sensitive results, then a large enough cutoff should be applied so that at least one relevant result may appear in the top results. Finding at least one relevant document is important in interactive search applications, because otherwise there could be little basis on which a user could reformulate their query. The sensitivity of

documents is not known at test time, we need to estimate sensitivity with the help of the sensitivity classifier we have built. Specifically, we estimate the expected number of documents that are predicted to be sensitive. We call this policy "dynamic cutoff selection", as the best cutoff is determined for each topic separately and different topics might have different cutoffs.

In our experiments, we compute the percentage of documents associated with a specific topic that are predicted to be sensitive, and define a monotonically decreasing function to compute the cutoff as follows.

$$cutoff = \lfloor 10 * (1 - \#sens/\#docs) \rfloor \tag{6.1}$$

where $\#sens$ is the number of documents predicted to be sensitive, and $\#docs$ is the total number of documents to be (re-)ranked. The resulting cutoff has the range [0,10]. We set the maximum cutoff to 10 in this setting, which is equal to the original fixed cutoff in our previous experiments. But, it too could be learned, or set to any other fixed value.

6.4.3 Results

As Table 6.8 shows, these cutoff selection policies help in improving TERN scores when compared against TERN@10. Several observations stand out of this analysis. First, we note that both proposed policies either improve or equalize the effectiveness when a fixed cutoff at position 10. This observation does not hold in the case of a static cutoff policy when applied on the John Snibert test collection and

Cutoff Policy	OHSUMED		
	M=0	M=1	M=3
Cutoff = 10	0.792	0.679	0.462
Static cutoff	0.802	0.717	**0.481**
Dynamic cutoff	**0.821**	**0.736**	**0.481**
Oracle	0.821	0.821	0.811

Cutoff Policy	Holly Palmer		
	M=0	M=1	M=3
Cutoff = 10	0.6	0.514	0.343
Static cutoff	0.6	0.514	0.343
Dynamic cutoff	0.6	0.514	0.343
Oracle	0.6	0.6	0.6

Cutoff Policy	John Snibert		
	M=0	M=1	M=3
Cutoff = 10	0.371	0.0	-0.743
Static cutoff	0.371	-0.114	-0.114
Dynamic cutoff	0.371	**0.229**	**0.0**
Oracle	0.429	0.429	0.429

Table 6.8: TERN of the best performing approach for each test collection under different cutoff selection policies. We did not find any significant difference according to a two-tailed paired sign test ($p < 0.05$) over the corresponding TERN@10 in the same column. The highest value other than the oracle cutoff in each column for each test collection is marked in bold.

sensitivity penalty M = 1. This exception happens because the static cutoff policy fails in some search topics (1, 3, 5, and 7) by setting small cutoff at position 1 which unnecessarily hides relevant documents, as shown in Figure 6.4. Second, we find these policies can adapt to test collections with different sensitivity distributions. For example, in the Holly Palmer test collection, which does not have many sensitive documents, we can see that applying cutoff selection policies does not decrease the effectiveness of our models. This is because the score reaches the highest value and saturates after cutoff=5, as seen in Figure 6.3(b), in which case any policy that selects cutoff $\geqslant 5$ will achieve the same score.

Third, we note that the dynamic cutoff selection policy seems promising be-

cause our simple implementation (Equation 6.1) outperforms learned fixed cutoffs despite no reported significant differences. This observation opens the door towards implementing more advanced formulae to map from a topic's estimated difficulty to a good optimal cutoff in order to better close the gap between current proposed policies and the oracle policy.

To better visualize how the resulting cutoffs from applying the proposed policies compared with the optimal cutoffs, Figure 6.4 shows the computed cutoffs per search topic for the John Snibert test collection. Optimal cutoff is defined as the largest cutoff where the TERN score is maximized and after which the score drops. This definition breaks ties because the maximum TERN score can be achieved at multiple cutoffs. No cutoff that is higher than the optimal cutoff can achieve the maximum TERN score, but smaller cutoffs can. As can be seen from the Figure 6.4, optimal cutoffs are closer to query specific cutoffs than to static cutoffs on average across queries. Hence, the dynamic cutoff policy achieves better effectiveness than the static policy on these test collections and sensitivity penalties.

6.5 Chapter Summary

In this chapter, we investigate enhancing the results presented in the previous chapter in different ways. First, results show that some measures model the task with higher fidelity but are not suitable for training. For example, we find that TERN measure could be a good fit for evaluating ranked lists, but it is not the best fit to be used for optimization. Actually, we find that training using nγCS-DCG

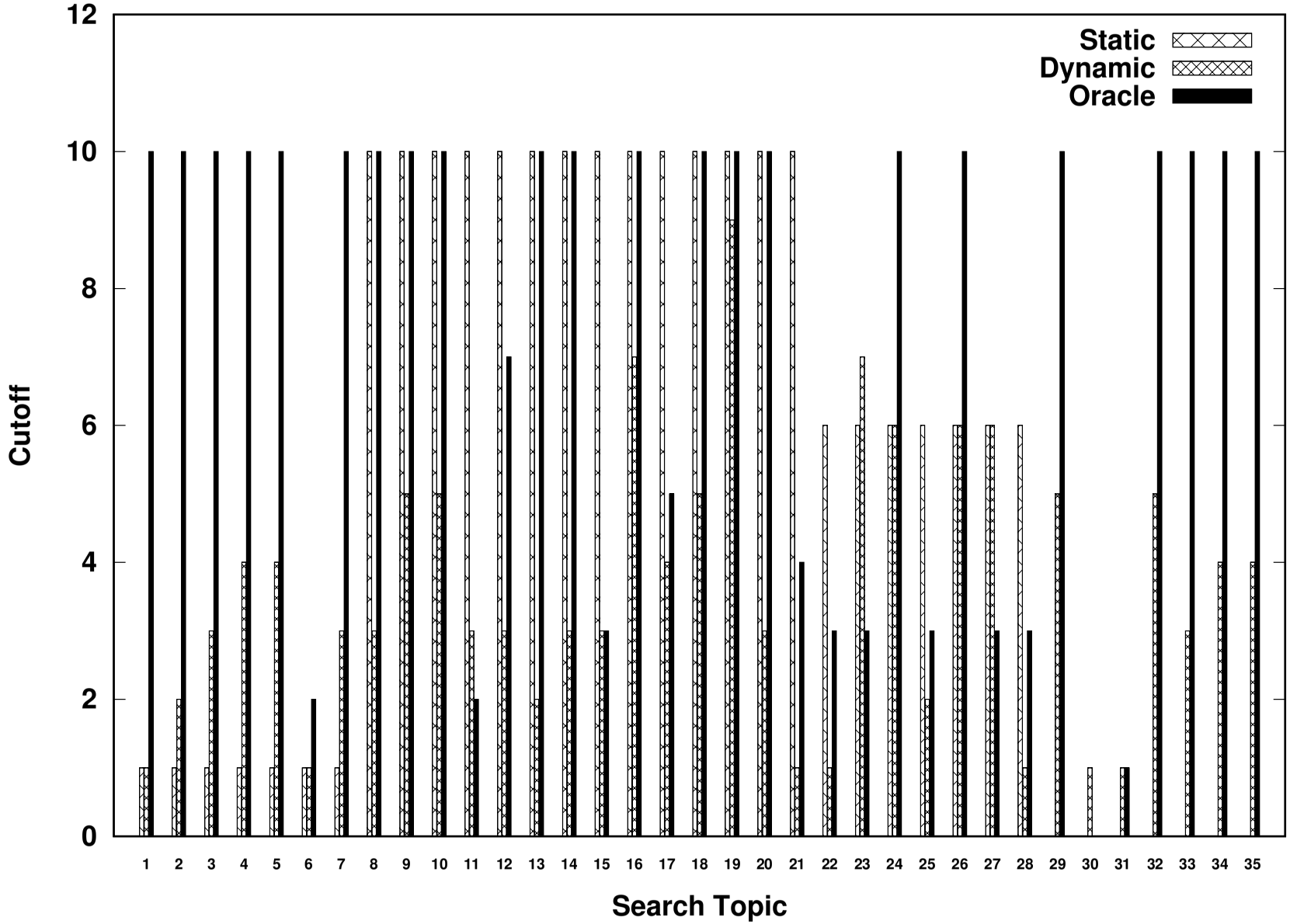


Figure 6.4: Cutoffs of the proposed policies in addition to the optimal cutoffs computed for the search topics in the John Snibert test collection.

yields to better models. Second, we show that changing the representation of query-document pairs by the help of transformer-based rankers improves the effectiveness over when hand-crated features are used.

Then, we 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. We also have been able to show that a careful selection to the ranking cutoff can limit the risk of showing sensitive results while still displaying some relevant results.

Chapter 7: Conclusion

Search among sensitive content is a novel retrieval task that aims at retrieving relevant content while protecting sensitive content. This problem of deciding what information can be shown in response to a request arises in many settings, including protection of attorney-client privilege [47], protection of government interests [71], and protection of personal privacy [56]. The key to our approach is to put the cart before the horse, first knowing what content the users actually wish to see and then making sensitivity decisions about that content. The work presented in this dissertation advances the state of the art in the design and evaluation of information retrieval systems by structuring and instantiating novel evaluation measures that will drive the creation of systems that can effectively search among secrets. This dissertation provides a proof of concept for this general idea (Contribution **S1**) by studying some of its key components. Our completed work represents initial steps towards building effective search and protection engines.

In Chapter 3, we have created two test collections based on the Avocado email research collection (Contribution **T1**). Additionally, we repurposed the OHSUMED test collection by using two MeSH labels to represent the sensitive content (Contribution **T2**). These test collections support experimentation with a search and

protection engine, and we have illustrated the use of these test collections both for training a sensitivity classifier and for building search and protection engines.

In Chapter 4, we have developed a set of potentially desirable properties for the task of search among sensitive content (Contribution **E1**). Additionally, we have shown how some standard evaluation measures can be extended to incorporate sensitivity in addition to relevance. We have proposed four measures: 1) TERN, 2) SENS, 3) CS-DCG (and its normalized form nCS-DCG), and 4) γCS-DCG (and its normalized form nγCS-DCG). These measures can be used for evaluating the effectiveness of search and protection engines (Contributions **E2**, **E3**, and **E4**), but they differ in how to compute the gains for showing relevant results and costs for showing sensitive results. We analyzed these measures in terms whether they satisfy each of several important properties (Contribution **E5**). Our analysis has shown that the TERN and SENS measures have advantages for evaluating ranked lists. This is because these measures contain a reference point around which it is easy to interpret the relevance and sensitivity contributions. However, the TERN and SENS measures exhibit quantization noise, which is potentially problematic when they are used for training ranking models. On the other hand, we found that nCS-DCG did not correlate well with the number of queries that have at least one sensitive result, as shown in Section 6.3. However, nCS-DCG or nγCS-DCG can be used for training ranking models, as they correlate positively with inserting relevant results and hiding sensitive results.

After describing test collections and evaluation measures, we studied how to construct a ranked retrieval system that knows what to find and what not to find. In

Chapter 5, we have shown we can construct search and protection engines by having three main components: 1) a ranking model, which could be as simple as BM25 or more advanced such as an LtR model, 2) a sensitivity classifier, and 3) a sensitivity filter. Then, based on these components, four broad classes of approaches have been tried (Contribution **S1**). The simplest approach is prefiltering: decide what can not be searched, and then simply do not index that content. When that approach excludes too much, as it often does, an alternative is postfiltering: first perform the search, and then run some classification process on the results of that search so that sensitive content can be withheld. This approach is common, for example, when searching for evidence to exchange among the parties to a lawsuit [91], and when seeking to foster transparent government using processes such as Freedom of Information Act requests in the United States [8]. Another alternative would be training ranking models using demoted relevance labels for truly sensitive documents so that models output low relevance scores for sensitive documents at test time. As a result, sensitive documents do not appear among the top results. Finally, we propose integration of search and protection by developing search engines that balance the risk of showing sensitive content with the benefit of showing relevant content. This can be achieved by leveraging the flexibility of listwise LtR models in optimizing towards an evaluation measure (Contribution **S2**).

Then, we went on to show that at some levels of classifier accuracy that might be seen in practice (Contribution **S4**), 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) (Contribution **S3**). From the experiments,

we noticed substanial benefits from using the Coordinate Ascent listwise LtR algorithm. Although it is slower than AdaRank during training, it can be parallelized by putting random restarts in separate threads (although the library we used did not have this optimization feature). Furthermore, we have shown that improvements in classifier effectiveness, as measured by intrinsic measures such as F_1 and F_2 are predictive of better results when used as a part of an integrated search and protection engine [72] (Contribution **S8**).

The focus of Chapter 6 was to refine the results of the baselines and the joint approach. We have proposed four ways of improving the quality of results with respect to relevance and sensitivity. First, we have shown that it is often better to have the optimization measure be different from the evaluation measure. For example, we have shown that nCS-DCG is a better choice than TERN for optimization, as it correlates positively with actions that insert relevant documents and hide sensitive documents. Second, we have shown that transformer-based features are better at representing query-document pairs than relying on hand-crafted features, which might be challenging to extract. Our results have shown improvements in the SENS score when transformer-based features were used (Contribution **S5**).

Third, we have been able to show that a simple cluster-based replacement strategy can further improve an evaluation measure score by reducing the number of queries showing at least one sensitive result (Contribution **S6**). Fourth, we have been able to show that a careful selection of the ranking cutoff can limit the risk of showing sensitive results while still displaying some relevant results (Contribution **S7**).

7.1 Limitations

A number of important limitations should be kept in mind when interpreting the results reported in this dissertation. We note, in particular:

1. While using the OHSUMED test collection, we limited our choice to two MeSH labels: C12 and C13, to represent the sensitive content. The prevalence of documents marked with at least one of these labels is 8.4% among the full set of OHSUMED documents, and 12.2% among the documents that have been judged for relevance. Perhaps different choices to other MeSH labels could be made to test the efficacy of our proposed approaches in different settings.

2. For our email experiments, we have trained on sensitivity labels that are available only for relevant documents, but active learning might be used to extend the set of labeled documents in ways that could further improve classification accuracy.

3. Throughout our experiments, we assumed sensitivity judgments are binary. We might model multiple degrees of sensitivity that might have nonlinear cost functions.

4. Throughout our experiments, we assumed sensitivity judgments are correct. We might model some bounds of errors in these judgments, and then we might develop approaches that take labeling error into consideration.

5. We tested all our approaches using three test collections. More test collections

are needed to separate what works well in general from what works well on these specific test collections. Nonetheless, by moving from no collections to three, we have gained insights to support a robust evaluation infrastructure for the critical task of providing privacy-sensitive access to important text collections.

6. We adopt a simple model for sensitivity by assuming that sensitivity is determined by the document only. It does not depend on what the question is, who is asking, or the purpose of the use of the information [82].

7. Search topics for the avocado test collections were made by students. This could lead to an imperfect sample of information needs where searchers might have.

8. We did not evaluate our Avocado test collections. One way to do so is to check the stability of systems ranking when some judgments are removed from the test collection.

9. Automatic search systems that participated in creating pools for the search topics of the Avocado test collections were developed by one person (the author of this dissertation). Systems developed by different people may have been more diverse to retrieve as many relevant documents as possible.

10. During evaluation, we limited our choice to 3 different penalties for TERN and SENS measures (M = 0, 1, and 3). For nCS-DCG, we used only one penalty $C_s = 12$.

11. We proposed only one family of single-valued measures in which the user's browsing model is position-based [30]. In this model, the effect of retrieving a document is parameterized by the document's rank. We could introduce a second family of single-valued measures based on cascade models [23]. By following the cascade model, the user examines a ranked result list from top to bottom. At each rank, the user has a probability of stopping (being satisfied or disturbed) based on the relevance or sensitivity of the current document.

12. In Section 4.6, none of our proposed measures has satisfied all desired properties. So additional evaluation measures are needed to be developed that better satisfy these properties.

13. As seen in Section 6.4.2, there are hard queries that contain many sensitive results, while other queries are easy to answer. From this fact, we might conclude that an evaluation measure in which the hard queries receive more emphasis (e.g., a geometric mean) might yield different results [99].

14. Except for Section 5.5, we used Logistic Regression (LR) and Support Vector Machines (SVM) for sensitivity classification which are good in general for text classification. We could try other classification models, e.g. tree-based or neural models.

15. In our experiments, as mentioned in Section 5.1, we fine-tuned the hyper-parameters of our sensitivity classifiers optimizing towards F_1. We could optimize these classifiers for F_2 or even F_4 to give more weight to recall over

precision when predicting sensitivity.

16. In the OHSUMED test collection, we have trained a classifier for the categories we use as surrogates for sensitivity by using very large amounts of training examples; we could 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.

17. In Section 5.5, our classifiers used only word and word sequence features, but in applications such as email or social media classification additional features such as relationship graphs and temporal patterns might help to further improve classification accuracy.

18. In Section 6.4.2, we implemented a simple formula to estimate the query difficulty in Equation 6.1. The query difficulty is estimated by computing the ratio of the documents that are predicted to be sensitive. That computation lacks other signals that could be beneficial for estimating query difficulty (e.g., query terms and the order of the results when the query is run). Furthermore, the problem can be cast to a regression problem which aims to predict the best cutoff given a query and a set of documents to rank, and training samples could be extracted from a held-out set of search queries and their optimal cutoffs. Despite the simplicity of our current method, using dynamic cutoffs shows promising results when compared with static learned cutoffs or fixed cutoffs (@10). Building better estimators for query difficulty could lead to cutoffs that are even closer to the oracle cutoffs.

7.2 Future Work

In this section, we describe the next steps that can be done to the problem we target.

7.2.1 Training Using Query-Specific Cutoffs

Throughout our experiments, we build listwise LtR models optimizing towards an optimization measure, e.g. TERN, at a fixed cutoff @10. The measure used for optimization serves as the objective function that drives the learning algorithm to build effective ranking models that balance between relevance and sensitivity. In search among sensitive content, we can see that the optimization algorithm learns to perform two actions to maximize the optimization measure at cutoff @k: 1) inserting relevant documents among the top k ranks, and 2) hiding sensitive documents from the top k ranks.

During our experiments, we noticed that there are some queries that are hard to answer. This is because the query had many sensitive documents. As a result, most of the ranking models failed in hiding all sensitive documents from the top ranks that are shown to the searcher. If these hard queries are used for training, it might be hard for the learning algorithm to build an effective ranking model, as the optimization measure might not improve where there are still sensitive results among the top k ranks. For example, if a query has more than one sensitive result, successfully hiding one sensitive document will not improve the TERN@k score as it will still be -M. To mitigate this issue, we propose to reduce the cutoff for

these queries, and hence the optimization problem becomes easier for the learning algorithm.

Inspired by the idea of query-specific cutoffs, presented in Section 6.4.2, we would like to introduce query-specific cutoffs during the training, so that the ranking model can be optimized towards these cutoffs instead of a fixed cutoff (e.g., 10 as we presently do). This direction has a secondary objective which is mitigating the drawbacks of both SENS and TERN measures in which they take discrete values, and hence they are not suitable to be used for optimization (as shown in Section 4.6). For example, when a small cutoff is applied for a hard query during training, there is smaller chance that there are many sensitive results initially. As a result, during optimization, it is easier for the learning model to see improvements in the TERN and SENS scores (e.g., TERN does not stick being negative and becomes positive). However, it is not clear the effect of changing the search cutoff during training when nCS-DCG, or nγCS-DCG, is used for optimization. This is because changing cutoffs affects both the numerator (discounted gains and costs of documents in the top ranks) and denominator (ideal and worst ranking scores) for both nCS-DCG and nγCS-DCG. So experiments are needed to empirically verify the benefits of this direction across different test collections and different measures.

7.2.2 Directly Optimizing Towards Our Measures

How to directly optimize ranking measures is an interesting but challenging problem. This problem arises because ranking measures depend on ranks that are

usually computed by sorting documents by their scores. As a result, ranking measures are either flat or discontinuous everywhere; they are not differentiable with respect to the score outputted by the ranking function. There are two categories of solutions to tackle this problem. First, iterative training by using ranking measures is used to reweight query-document pairs. For example, AdaRank [124] applies boosting [107] to the ranking problem, and uses the value of the ranking measure of each query to compute a weight for it in the next training iteration. In our work, we adopt this approach by running the AdaRank [124] and Coordinate Ascent [75] ranking algorithms, which do not require the optimization measure to be differentiable.

The second approach is to replace the optimization measure with a differentiable surrogate in terms of the ranking parameters. Hence, gradient descent methods can be utilized to optimize towards the approximated surrogate measures. ApproxNDCG [96] is one example of approximating nDCG where the ranks are approximated by the scores outputted from the ranking function. The intuition behind this is that a document's rank is determined by the number of documents whose scores are higher than its score. Hence, ranks can be approximated by the scores of the ranking output. Because nDCG is a utility, a loss function can be defined as negative ApproxNDCG, and the loss can be minimized using gradient descent [13]. We would like to apply the same approximation framework on our measures, especially nCS-DCG and nγCS-DCG, as they are direct extensions to the standard nDCG that show better effectiveness than TERN and SENS when used during training.

Ultimately, we would like to build neural models that are optimized towards the resulting surrogate measures. Currently, we used monoBERT for inference only, without changing BERT parameters, and then run listwise LtR algorithms on top of the monoBERT's [CLS] token embeddings. We are interested in studying the performance if we further fine-tune the BERT parameters based on a cost function that is defined on a differentiable surrogate measure. TFR-BERT [53] could be an option to try, as it requires that the optimization measure to be differentiable [96, 119].

7.2.3 Archivist in the Loop

In Chapter 5, we proposed different approaches that hide sensitive content from being shown to the searcher. But we found that such fully automatic systems may suffer from mistakes, and thus they should not be working alone in answering queries in at least some applications. However, since people are far better than machines at drawing inferences from running text. We therefore propose an active learning strategy where an archivist intervenes to manually review content which the sensitivity classifier is most uncertain about.

Previously, we classify documents as sensitive (hidden from user) or non-sensitive (fine to show). We propose to extend it to ternary states by adding another state when the sensitivity classifier is uncertain about a document. In this case, the specified document needs to be reviewed by an archivist. Assuming the archivist is perfect in deciding the relevance and sensitivity of a document. If the document

is relevant and non-sensitive, the archivist sends it to the searcher as an additional result. In any case, the archivist's feedback will enable the sensitivity classifier to adapt to the sensitivities within the collection. We therefore plan to adopt an archivist-in-the-loop process in which on-demand review is focused on documents in which the searcher is interested, and the classifier is uncertain about.

As depicted in Figure 7.1, the search process begins when the searcher submits a query. Then the search engine, with help of the sensitivity classifier, produces a ranked list of documents that is split on two sublists: 1) searcher's list, and 2) archivist's list. The former represents a ranked list of documents which are predicted as relevant and nonsensitive, and can be shown to the searcher immediately. The latter sublist is a ranked list of documents which the sensitivity classifier is most uncertain about. Then the archivist has to review all assigned documents. The archivist then sends any relevant and non-sensitive documents back to the searcher, and all feedback is sent back to the search engine to retrain it to the sensitives found in the reviewed documents.

To measure the goodness of the proposed system, we propose a new evaluation measure with the following goals.

- As in normal search engines, we need to maximize the number of relevant documents that can be shown directly to the user.
- We need to minimize the amount of review done by the archivist, and hence we need to reduce the number of documents to be reviewed.
- We need to minimize the number of mistakes, e.g. showing sensitive documents

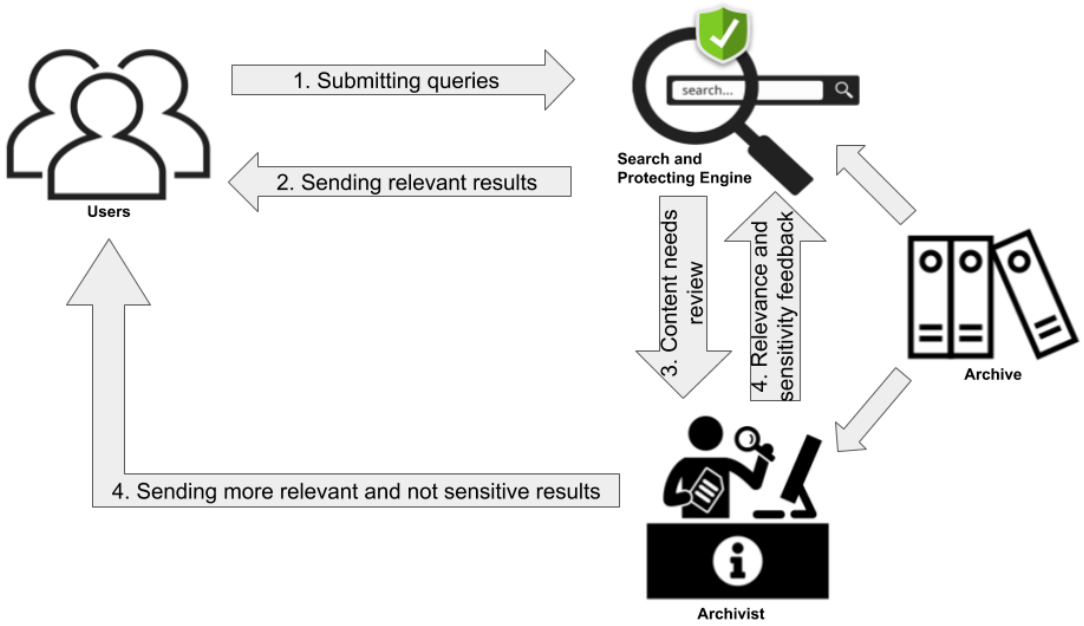


Figure 7.1: Workflow for search and protection engines with an archivist in the loop.

to the user.

For example, we might define the following measure.

$$O\text{-}CS\text{-}DCG@k = \sum_{i=1}^{k_s}(g_i * d_i - c_i * \gamma^{s_i}) + \sum_{i=1}^{k_a}(g'_i * d'_i - e) \tag{7.1}$$

where k_s and k_a represent the number of documents processed by an S&P engine and archivist, respectively, and $k_s + k_a = k$. e is the archivist's effort in reviewing a document. In this measure, we set different gains (g') and discounts (d') for the list processed by the archivist and the searcher. It can be seen that *O-CS-DCG* is composed of two terms: 1) the first term represents the original γCS-DCG, while 2) the second term represents the extra gain when the archivist sends additional relevant documents.

7.3 Implications

The range of applications to which effective and well-characterized techniques for search among sensitive content could be applied is substantial. Examples include: 1) archival access to email donated by scholars and 2) responsiveness and privilege review in e-discovery. These two settings are of substantial current interest, and clear deployment pathways exist for the techniques that we have developed. Applications to government transparency are also evident. Our research proposes to make the sensitivity determinations fully automatically, but the techniques that we have developed would also support the first stage of a process in which very highly-sensitive content (e.g., classified materials that must be reviewed for declassification) could be sent for manual review only if current users actually wished to see it, as described in Section 7.2.3. Such techniques could facilitate serving Freedom of Information Act requests, as well as review for declassification and public release of the growing backlog of documents requiring systematic review (e.g., after 25 years). Additionally, in the United States, both national [41] and state [92] interests have called for improved ways of prioritizing the declassification review task in order to intelligently manage the huge wave of documents requiring review. Among other urgent needs for the technology that we propose to develop, the "right to be forgotten" that has been recognized by the European Court of Justice [98] imposes requirements on search engine services to prevent serving content deemed sensitive by individuals. Such requests are already at a staggering volume, and still growing. For example, in the 18 months between May 2014 and November 2015, more than

340,000 people requested that more than 1.2 million URLs be removed from the index of Google search services that are widely used in Europe, and 42% of these requests were granted.

It may be, however, that the most important applications for search among sensitive content, will be the ones that emerge after the capability is in hand. At present, parents do not want their children to search their email, and children do not want their parents to search their chat logs, because no means exists to assure that the content there that should be private will remain so. Privacy concerns evoked when Google Glass, or similar devices still in the lab, become mainstream are not fundamentally rooted in what might be recorded, but rather in how those recordings might be used. These concerns will remain salient for at least as long as we deny ourselves the ability to search among sensitive content in ways that balance the interests of both the content creators and those who wish to find and use that content.

Bibliography

[1] Daniel Abril, Guillermo Navarro-Arribas, and Vicenç Torra. On the declassification of confidential documents. In *International Conference on Modeling Decisions for Artificial Intelligence*, pages 235–246. Springer, 2011.

[2] Sakhar Alkhereyf and Owen Rambow. Work hard, play hard: Email classification on the avocado and enron corpora. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 57–65, 2017.

[3] Enrique Amigó, Julio Gonzalo, and Felisa Verdejo. A general evaluation measure for document organization tasks. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 643–652, 2013.

[4] Enrique Amigó, Damiano Spina, and Jorge Carrillo-de Albornoz. An axiomatic analysis of diversity evaluation metrics: Introducing the rank-biased utility metric. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 625–634. ACM, 2018.

[5] Ron Artstein and Massimo Poesio. Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596, 2008.

[6] Javed A Aslam, Virgil Pavlu, and Emine Yilmaz. A statistical method for system evaluation using incomplete judgments. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 541–548. ACM, 2006.

[7] Javed A Aslam, Emine Yilmaz, and Virgiliu Pavlu. The maximum entropy method for analyzing retrieval measures. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 27–34. ACM, 2005.

[8] Jason R Baron, Mahmoud F Sayed, and Douglas W Oard. Providing more efficient access to government records: A use case involving application of machine learning to improve foia review for the deliberative process privilege. *arXiv preprint arXiv:2011.07203*, 2020.

[9] John H Beisner. Discovering a better way: The need for effective civil litigation reform. *Duke Law Journal*, pages 547–596, 2010.

[10] Ron Bekkerman. Automatic categorization of email into folders: Benchmark experiments on enron and sri corpora. *Technical Report, Dept. of Computer Science, Univ. of Massachusetts*, 2004.

[11] Jiang Bian, Xin Li, Fan Li, Zhaohui Zheng, and Hongyuan Zha. 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*, pages 131–140. ACM, 2010.

[12] George EP Box. Robustness in the strategy of scientific model building. In *Robustness in Statistics*, pages 201–236. Elsevier, 1979.

[13] Sebastian Bruch, Masrour Zoghi, Michael Bendersky, and Marc Najork. Revisiting approximate metric optimization in the age of deep neural networks. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1241–1244, 2019.

[14] Chris Buckley and Ellen M Voorhees. Retrieval evaluation with incomplete information. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 25–32. ACM, 2004.

[15] Chris Buckley and Ellen M Voorhees. Evaluating evaluation measure stability. In *ACM SIGIR Forum*, volume 51, pages 235–242. ACM, 2017.

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

[17] Christopher J Burges, Robert Ragno, and Quoc V Le. Learning to rank with nonsmooth cost functions. In *Advances in neural information processing systems*, pages 193–200, 2007.

[18] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[19] Luca Busin and Stefano Mizzaro. Axiometrics: An axiomatic approach to information retrieval effectiveness metrics. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, page 8. ACM, 2013.

[20] Heejung Byun and David A Kirsch. Organizational timing norms: Evidence from email time-to-responses. In *Academy of Management Proceedings*, volume 2015, page 17825. Academy of Management Briarcliff Manor, NY 10510, 2015.

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

[22] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[23] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 621–630, 2009.

[24] Alan Cooper. The origin of personas. *Innovation*, 23(1):26–29, 2004.

[25] Gordon V Cormack and Maura R Grossman. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 153–162. ACM, 2014.

[26] Gordon V Cormack, Maura R Grossman, Bruce Hedin, and Douglas W Oard. Overview of the trec 2010 legal track. In *Proc. 19th Text REtrieval Conference*, volume 1, 2010.

[27] David Cossock and Tong Zhang. Subset ranking using regression. In *International Conference on Computational Learning Theory*, pages 605–619. Springer, 2006.

[28] Kristof Coussement and Dirk Van den Poel. Improving customer complaint management by automatic email classification using linguistic style features as predictors. *Decision Support Systems*, 44(4):870–882, 2008.

[29] Nick Craswell, Bhaskar Mitra, Emine Yilmaz, Daniel Campos, and Jimmy Lin. Ms marco: Benchmarking ranking models in the large-data regime. *arXiv preprint arXiv:2105.04021*, 2021.

[30] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 87–94, 2008.

[31] Elisabeth Crawford, Judy Kay, and Eric McCreath. Automatic induction of rules for e-mail classification. 2001.

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

[33] Na Dai, Milad Shokouhi, and Brian D Davison. Multi-objective optimization in learning to rank. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 1241–1242. ACM, 2011.

[34] Zhuyun Dai and Jamie Callan. Deeper text understanding for ir with contextual neural language modeling. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 985–988, 2019.

[35] Anubrata Das and Matthew Lease. A conceptual framework for evaluating fairness in search. *arXiv preprint arXiv:1907.09328*, 2019.

[36] Joost CF De Winter. Using the student's t-test with extremely small sample sizes. *Practical Assessment, Research, and Evaluation*, 18(1):10, 2013.

[37] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.

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

[40] Anlei Dong, Ruiqiang Zhang, Pranam Kolari, Jing Bai, Fernando Diaz, Yi Chang, Zhaohui Zheng, and Hongyuan Zha. Time is of the essence: Improving recency ranking using twitter data. In *Proceedings of the 19th International Conference on the World Wide Web*, pages 331–340. ACM, 2010.

[41] Martin C Faga. *Improving Declassification: A Report to the President from the Public Interest Declassification Board*. DIANE Publishing, 2008.

[42] Marco Ferrante, Nicola Ferro, and Maria Maistro. Towards a formal framework for utility-oriented measurements of retrieval effectiveness. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, pages 21–30, 2015.

[43] Matthias Feurer, Katharina Eggensperger, Stefan Falkner, Marius Lindauer, and Frank Hutter. Auto-sklearn 2.0: The next generation. *arXiv preprint arXiv:2007.04074*, 2020.

[44] Herbert N Foerstel. *Freedom of Information and the Right to Know: The Origins and Applications of the Freedom of Information Act*. Greenwood Press Westport, CT, 1999.

[45] Yoav Freund, Raj Iyer, Robert E Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of machine learning research*, 4(Nov):933–969, 2003.

[46] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[47] Manfred Gabriel, Chris Paskach, and David Sharpe. The challenge and promise of predictive coding for privilege. In *ICAIL 2013 DESI V Workshop*, 2013.

[48] Luke Gallagher, Antonio Mallia, J Shane Culpepper, Torsten Suel, and B Barla Cambazoglu. Feature extraction for large-scale text collections. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3015–3022, 2020.

[49] Edmund A Gehan. A generalized wilcoxon test for comparing arbitrarily singly-censored samples. *Biometrika*, 52(1-2):203–224, 1965.

[50] David Graus, David Van Dijk, Manos Tsagkias, Wouter Weerkamp, and Maarten De Rijke. Recipient recommendation in enterprises using communication graphs and email content. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1079–1082. ACM, 2014.

[51] Maura R Grossman and Gordon V Cormack. Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Rich. JL & Tech.*, 17:1, 2010.

[52] Cloyd Dake Gull. Seven years of work on the organization of materials in the special library. *American Documentation*, 7(4):320–329, 1956.

[53] Shuguang Han, Xuanhui Wang, Mike Bendersky, and Marc Najork. Learning-to-rank with bert in tf-ranking. *arXiv preprint arXiv:2004.08476*, 2020.

[54] Donna Harman. Overview of the second text retrieval conference (trec-2). *Information Processing & Management*, 31(3):271–289, 1995.

[55] Marti A Hearst. Teaching applied natural language processing: Triumphs and tribulations. In *Proceedings of the Second ACL Workshop on Effective Tools and Methodologies for Teaching NLP and CL*, pages 1–8, 2005.

[56] Modassir Iqbal, Katie Shilton, Mahmoud F Sayed, Douglas Oard, Jonah Lynn Rivera, and William Cox. Search with discretion: Value sensitive design of training data for information retrieval. *Proceedings of the ACM on Human-Computer Interaction*, 5(CSCW1):1–20, 2021.

[57] Koichi Iwai, Kaoru Iida, Masanori Akiyoshi, and Norihisa Komoda. A classification method of inquiry e-mails for describing faq with self-configured class dictionary. In *Distributed Computing and Artificial Intelligence*, pages 35–43. Springer, 2010.

[58] Sanaz Jabbari, Ben Allison, David Guthrie, and Louise Guthrie. Towards the orwellian nightmare: Separation of business and personal emails. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 407–411. Association for Computational Linguistics, 2006.

[59] Emily Jamison and Iryna Gurevych. Headerless, quoteless, but not hopeless? using pairwise email classification to disentangle email threads. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 327–335, 2013.

[60] Kalervo Järvelin and Jaana Kekäläinen. 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*, pages 41–48. ACM, 2000.

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

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

[63] Thorsten Joachims. A support vector method for multivariate performance measures. In *Proceedings of the 22nd international conference on Machine learning*, pages 377–384. ACM, 2005.

[64] Bryan Klimt and Yiming Yang. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*, pages 217–226. Springer, 2004.

[65] Hang Li. Learning to rank for information retrieval and natural language processing. *Synthesis Lectures on Human Language Technologies*, 4(1):1–113, 2011.

[66] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. Pretrained transformers for text ranking: Bert and beyond. *arXiv preprint arXiv:2010.06467*, 2020.

[67] Christina Lioma, Jakob Grue Simonsen, and Birger Larsen. Evaluation measures for relevance and credibility in ranked lists. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 91–98. ACM, 2017.

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

[69] Kirsten Martin and Helen Nissenbaum. Measuring privacy: An empirical test using context to expose confounding variables. *Colum. Sci. & Tech. L. Rev.*, 18:176, 2016.

[70] Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 437–444. ACM, 2006.

[71] Graham McDonald, Craig Macdonald, and Iadh Ounis. Enhancing sensitivity classification with semantic features using word embeddings. In *European Conference on Information Retrieval*, pages 450–463. Springer, 2017.

[72] Graham Mcdonald, Craig Macdonald, and Iadh Ounis. How the accuracy and confidence of sensitivity classification affects digital sensitivity review. *ACM Transactions on Information Systems (TOIS)*, 39(1):1–34, 2020.

[73] Graham McDonald, Craig Macdonald, Iadh Ounis, and Timothy Gollins. Towards a classifier for digital sensitivity review. In *European Conference on Information Retrieval*, pages 500–506. Springer, 2014.

[74] Quinn McNemar. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika*, 12(2):153–157, 1947.

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

[76] Donald A Metzler, W Bruce Croft, and Andrew McCallum. Direct maximization of rank-based metrics for information retrieval. Technical report, Citeseer, 2005.

[77] Tanushree Mitra and Eric Gilbert. Analyzing gossip in workplace email. *ACM SIGWEB Newsletter Winter*, 5, 2013.

[78] Alistair Moffat. Seven numeric properties of effectiveness metrics. In *Asia Information Retrieval Symposium*, pages 1–12. Springer, 2013.

[79] Ghulam Mujtaba, Liyana Shuib, Ram Gopal Raj, Nahdia Majeed, and Mohammed Ali Al-Garadi. Email classification research trends: Review and open issues. *IEEE Access*, 5:9044–9064, 2017.

[80] Deirdre K Mulligan, Colin Koopman, and Nick Doty. Privacy is an essentially contested concept: a multi-dimensional analytic for mapping privacy. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2083):20160118, 2016.

[81] David F Nettleton and Daniel Abril. Document sanitization: Measuring search engine information loss and risk of disclosure for the wikileaks cables. In *International Conference on Privacy in Statistical Databases*, pages 308–321. Springer, 2012.

[82] Helen Nissenbaum. *Privacy in Context: Technology, Policy, and the Integrity of Social Life*. Stanford University Press, 2009.

[83] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[84] Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. *arXiv preprint arXiv:2003.06713*, 2020.

[85] Rodrigo Nogueira, Wei Yang, Kyunghyun Cho, and Jimmy Lin. Multi-stage document ranking with bert. *arXiv preprint arXiv:1910.14424*, 2019.

[86] Rabia Nuray and Fazli Can. Automatic ranking of retrieval systems in imperfect environments. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 379–380. ACM, 2003.

[87] Douglas Oard, William Webber, David Kirsch, and Sergey Golitsynskiy. Avocado research email collection. *Linguistic Data Consortium*, pages 29–32, 2015.

[88] Douglas W Oard, Jason R Baron, Bruce Hedin, David D Lewis, and Stephen Tomlinson. Evaluation of information retrieval for e-discovery. *Artificial Intelligence and Law*, 18(4):347–386, 2010.

[89] Douglas W Oard, Fabrizio Sebastiani, and Jyothi K Vinjumur. Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery. *ACM Transactions on Information Systems (TOIS)*, 37(1):11, 2018.

[90] Douglas W Oard, Jyothi Vinjumur, and Fabrizio Sebastiani. When is it rational to review for privilege? In *Proceedings of the Workshop on Using Advanced Data Analysis in eDiscovery*, volume 8, 2017.

[91] Douglas W Oard, William Webber, et al. Information retrieval for e-discovery. *Foundations and Trends® in Information Retrieval*, 7(2–3):99–237, 2013.

[92] Brett Orzechowski. *FOIL: The Law and the Future of Public Information in New York*. Syracuse University Press, 2018.

[93] Gabriella Pasi, Gloria Bordogna, and Robert Villa. A multi-criteria content-based filtering system. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 775–776. ACM, 2007.

[94] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2970–2978. ACM, 2019.

[95] John Pruitt and Tamara Adlin. *The Persona Lifecycle: Keeping People in Mind Throughout Product Design*. Elsevier, 2010.

[96] Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information retrieval*, 13(4):375–397, 2010.

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

[98] Neil Richards. *Intellectual Privacy: Rethinking Civil Liberties in the Digital Age*. Oxford University Press, USA, 2015.

[99] Stephen Robertson. On gmap: and other transformations. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, pages 78–83, 2006.

[100] Mehran Sahami, Susan Dumais, David Heckerman, and Eric Horvitz. A bayesian approach to filtering junk e-mail. In *Learning for Text Categorization: Papers from the 1998 workshop*, volume 62, pages 98–105. Madison, Wisconsin, 1998.

[101] Tetsuya Sakai and Zhaohao Zeng. Which diversity evaluation measures are "good"? In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 595–604, 2019.

[102] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[103] Mahmoud F Sayed, William Cox, Jonah Lynn Rivera, Caitlin Christian-Lamb, Modassir Iqbal, Douglas W Oard, and Katie Shilton. A test collection for relevance and sensitivity. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1605–1608, 2020.

[104] Mahmoud F Sayed, Nishanth Mallekav, and Douglas W Oard. Comparing intrinsic and extrinsic evaluation of sensitivity classification. In *preparation*.

[105] Mahmoud F Sayed and Douglas W Oard. Filtering is not enough: Towards building a search and protection engine. In *preparation*.

[106] Mahmoud F Sayed and Douglas W Oard. Jointly modeling relevance and sensitivity for search among sensitive content. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 615–624. ACM, 2019.

[107] Robert E Schapire. Explaining adaboost. In *Empirical Inference*, pages 37–52. Springer, 2013.

[108] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.

[109] Krysta M Svore, Maksims N Volkovs, and Christopher JC Burges. Learning to rank with multiple objective functions. In *Proceedings of the 20th international conference on World wide web*, pages 367–376. ACM, 2011.

[110] Zhiwen Tang and Grace Hui Yang. Investigating per topic upper bound for session search evaluation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 185–192. ACM, 2017.

[111] Michael Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: Optimizing non-smooth rank metrics. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 77–86. ACM, 2008.

[112] E Dale Thompson and Michelle L Kaarst-Brown. Sensitive information: A review and research agenda. *Journal of the American Society for Information Science and Technology*, 56(3):245–257, 2005.

[113] Jyothi K Vinjumur and Douglas W Oard. Finding the privileged few: Supporting privilege review for e-discovery. *Proceedings of the Association for Information Science and Technology*, 52(1):1–4, 2015.

[114] Ellen M Voorhees. On expanding query vectors with lexically related words. In *TREC*, pages 223–232, 1993.

[115] Ellen M Voorhees, Dawn M Tice, et al. The trec-8 question answering track evaluation. In *TREC*, volume 1999, page 82. Citeseer, 1999.

[116] Lidan Wang, Jimmy Lin, and Donald Metzler. Learning to efficiently rank. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 138–145. ACM, 2010.

[117] Man Wang, Yifan He, and Minghu Jiang. Text categorization of enron email corpus based on information bottleneck and maximal entropy. In *IEEE 10th International Conference on Signal Processing*, pages 2472–2475. IEEE, 2010.

[118] Min-Feng Wang, Sie-Long Jheng, Meng-Feng Tsai, and Cheng-Hsien Tang. Enterprise email classification based on social network features. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 532–536. IEEE, 2011.

[119] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1313–1322, 2018.

[120] Shawn R Wolfe and Yi Zhang. User-centric multi-criteria information retrieval. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 818–819. ACM, 2009.

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

[122] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th international conference on Machine learning*, pages 1192–1199. ACM, 2008.

[123] Jun Xu, Yunbo Cao, Hang Li, and Yalou Huang. Cost-sensitive learning of svm for ranking. In *European conference on machine learning*, pages 833–840. Springer, 2006.

[124] Jun Xu and Hang Li. Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 391–398. ACM, 2007.

[125] Emine Yilmaz and Javed A Aslam. Estimating average precision with incomplete and imperfect judgments. In *Proceedings of the 15th ACM international conference on Information and knowledge management*, pages 102–111. ACM, 2006.

[126] Seongwook Youn and Dennis McLeod. A comparative study for email classification. In *Advances and innovations in systems, computing sciences and software engineering*, pages 387–391. Springer, 2007.

[127] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. 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*, pages 271–278. ACM, 2007.

[128] Zhaohui Zheng, Hongyuan Zha, Tong Zhang, Olivier Chapelle, Keke Chen, and Gordon Sun. A general boosting method and its application to learning

ranking functions for web search. In *Advances in neural information processing systems*, pages 1697–1704, 2008.