6-2-2005

On-line Metasearch, Pooling, and System Evaluation

Robert A. Savell
Dartmouth College

Dartmouth College Computer Science Department Technical Report

TR2005-543

On-line Metasearch, Pooling, and System Evaluation

A Thesis

Submitted to the Faculty

in partial fulfillment of the requirements for the

degree of

Doctor of Philosophy

in

Computer Science

by

Robert A. Savell

DARTMOUTH COLLEGE

Hanover, New Hampshire

June 2th, 2005

Examining Committee:

(chair) Daniel Rockmore

Javed A. Aslam

Chris Bailey-Kellogg

John Lafferty

Charles K. Barlowe, Ph.D.
Dean of Graduate Studies

Abstract

This thesis presents a unified method for simultaneous solution of three problems in Information Retrieval— metasearch (the fusion of ranked lists returned by retrieval systems to elicit improved performance), efficient system evaluation (the accurate evaluation of retrieval systems with small numbers of relevance judgements), and pooling or "active sample selection" (the selection of documents for manual judgement in order to develop sample pools of high precision or pools suitable for assessing system quality). The thesis establishes a unified theoretical framework for addressing these three problems and naturally generalizes their solution to the on-line context by incorporating feedback in the form of relevance judgements.

The algorithm— *Rankhedge* for on-line retrieval, metasearch and system evaluation— is the first to address these three problems simultaneously and also to generalize their solution to the on-line context. Optimality of the Rankhedge algorithm is developed via Bayesian and maximum entropy interpretations. Results of the algorithm prove to be significantly superior to previous methods when tested over a range of TREC (Text REtrieval Conference) data. In the absence of feedback, the technique equals or exceeds the performance of benchmark metasearch algorithms such as CombMNZ and Condorcet. The technique then dramatically improves on this performance during the on-line metasearch process. In addition, the technique generates pools of documents which include more relevant documents and produce more accurate system evaluations than previous techniques. The thesis includes an information-theoretic examination of the original Hedge algorithm as well as its adaptation to the context of ranked lists. The work also addresses the concept of information-theoretic similarity within the Rankhedge context and presents a method for decorrelating the predictor set to improve worst case performance. Finally, an information-theoretically optimal method for probabilistic "active sampling" is presented with possible application to a broad range of practical and theoretical contexts.

Acknowledgments

This thesis is dedicated to Ruthie who made these years possible and to my daughter, Trillium FitzGerald Cserr, who made it all fun.

Many thanks to Jay Aslam, Dan Rockmore, and Chris Bailey-Kellog for helping me to thread the needle. Thanks also to my external committee member John Lafferty for his diligence and good will, and to Prasad Jayanti for his service on my proposal committee and for the valuable advice he has provided throughout my years at Dartmouth.

Contents

Abstract **ii**

Acknowledgments **iii**

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

1.1 Challenges in On-line Retrieval 2

1.1.1 Machine Learning and Information Theoretic Perspectives 3

1.2 Contributions 5

1.2.1 Practical Contributions 5

1.2.2 Basic Results 6

1.2.3 Modifications 6

1.2.4 Theoretical Contributions 8

1.2.5 Hedge in a Broader Context 9

1.3 Outline 11

2 IR Background **14**

2.1 Pooling for System Evaluation 15

2.1.1 Pooling without Relevance Judgements 17

2.1.2 On-line Pooling with Priority Queues 19
2.2 Metasearch 21
2.2.1 CombMNZ, CombSum, and CombANZ 22
2.2.2 Voting Methods: Borda Fuse and Condorcet 23
2.2.3 The Probabilistic Model 25
2.3 Cranking 27

3 A Unified Model 29

3.1 The Modified Hedge Algorithm 30
3.1.1 Intuition 31
3.1.2 Rankhedge for On-line Metasearch and System Evaluation 34
3.2 Results 40
3.3 Conclusions 45

4 Hedge for Ranked Lists 52

4.1 Background 53
4.1.1 Learning to Order Things 53
4.1.2 RankBoost 56
4.2 Prior Probability of Relevance and Inverse Rank 58
4.2.1 Empirical Evidence 60
4.2.2 Theoretical Justification for the Prior 64
4.3 Bounds on Accuracy 68
4.3.1 Output List Accuracy 69
4.3.2 Estimation of Predictor Accuracy 71
4.3.3 Restriction to Lossy Predictors 72
4.3.4 Expected Contribution to Sampling Pool 73
4.4 Optimization of Beta 78
4.4.1 Sensitivity to Selection of Beta 80

5 Hedge and Bayesian On-line Prediction 83

5.1 Bayesian On-line Prediction 84

5.1.1 Classification Rules 85

5.2 Hedge vs. Bayesian On-line Prediction 87

5.3 Information Theoretic Analysis 89

5.3.1 Proportional Resource Allocation 90

5.3.2 The Exponential Update Rule 92

5.4 Information Theoretic Hedge 95

5.5 Results 100

6 Similarity Measures 111

6.1 Background 112

6.1.1 Hedge and the Predictor Space 113

6.1.2 Markov Random Fields 114

6.1.3 Gibbs Distributions 115

6.2 Information Theoretic Similarity Measures 116

6.2.1 Background 117

6.2.2 Similarity Measures on the Space of Predictors 118

6.3 Decorrelation of Predictors 120

6.3.1 Results 121

7 Active Sampling 126

7.1 Background 127

7.1.1 The Query by Committee Algorithm 128

7.1.2 Semi-supervised Learning 129

7.2 Version Space and Risk 130

7.2.1 Risk 131

7.2.2 Sample Complexity 133

7.3 Error Bounds for On-line Algorithms 134

7.3.1 Expected Risk and Uncertainty 136
7.3.2 Cumulative Upper Bounds 138
7.3.3 Hedge and the Version Space 140
7.4 Active Sampling 142
7.4.1 A Probabilistic Method 143
7.4.2 Controlling Sampling 149
7.5 A Search Problem 150

8 Conclusion 154

8.1 Future Work 157
8.2 Beyond Traditional Metasearch 158

List of Tables

2.1 The size of the pool (averaged over 50 queries) for various pool depths if the pooling is performed TREC-style. Here n is the number of input systems contributing to the data set. 17

3.1 Rankhedge-0 Method (Rankhedge Metasearch List at depth zero) vs. Metasearch Techniques CombMNZ and Condorcet. 30

3.2 Constants used to establish β and val(r) for rankHedge trials (number of systems, minimum loss of best system, and expansion constant for harmonic series). 41

7.1 Joint distribution of hypotheses $\bar{h}_x = 1/4$ and field $\bar{f}_x = 5/8$ . . . 145

List of Figures

2.1 "TRECstyle 100 normal" method vs. actual rankings— TREC 8. 18

2.2 On-line Pooling Algorithm (Cormack et al. [23]). 20

2.3 Percent of total relevant documents discovered: On-line pooling with priority queue vs. Depth-n pools— TREC 8. 20

3.1 Hedge Algorithm. 33

3.2 Rankhedge Algorithm for On-line Metasearch. 35

3.3 RankHedge Subroutines: value() and loss(). 36

3.4 RankHedge Subroutine: poolDocuments(). 39

3.5 RankHedge Subroutine: generateMetasearchList(). 40

3.6 RankHedge Subroutine: evaluateSystems(). 41

3.7 Trecs 3 and 5: Results of Rankhedge— with log prior under three performance measures: (A) percent of total relevant documents discovered (B) system ordering via k-τ (C) metasearch performance via Mean Average Precision. 47

3.8 Trecs 6 and 7: Results of Rankhedge— with log prior under three performance measures: (A) percent of total relevant documents discovered (B) system ordering via k-τ (C) metasearch performance via Mean Average Precision. 48

3.9 Trecs 8 and 9: Results of Rankhedge— with log prior under three performance measures: (A) percent of total relevant documents discovered (B) system ordering via k-τ (C) metasearch performance via Mean Average Precision. 49

3.10 Trecs 3, 5, and 6: Depth-1 and equivalent Rankhedge-n rankings vs. actual ranks. 50

3.11 Trecs 7, 8, and 9: Depth-1 and equivalent Rankhedge-n rankings vs. actual ranks. 51

4.1 Algorithm for Ranking using Preference Functions. 54

4.2 The RankBoost Algorithm 56

4.3 $Pr(relevance|rank)$ vs. Actual: TRECs, 3,5,6,7,8 and 9. 60

4.4 $Pr(relevance|rank)$ (log-log scale): TRECs, 3,5,6,7,8 and 9. . . . 61

4.5 Trecs 3, 5 and 6: Precision at rank for Top 10 and Top 80% system bundles sorted by Mean Average Precision. Precision at rank is shown, rather than prior relevance at rank, for noise reduction. Maximum precision of all systems is scaled to one for clarity. . . 62

4.6 Trecs 7, 8, and 9 : Precision at rank for Top 10 and Top 80% system bundles sorted by Mean Average Precision. Precision at rank is shown, rather than prior relevance at rank, for noise reduction. Maximum precision of all systems is scaled to one for clarity. 63

5.1 General Bayesian On-line Prediction. 86

5.2 Bregman Divergences 93

5.3 Simple Properties of Bregman divergences. 93

5.4 Trec 3. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision. 103

5.5 Trec 5. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision. 104

5.6 Trec 6. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision. 105

5.7 Trec 7. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision. 106

5.8 Trec 8. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ.(C) metasearch performance via Mean Average Precision. 107

5.9 Trec 9. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision. 108

5.10 Trec 8: Evolution of predictor weights to depths 100 and 1000— averaged across 50 queries. Pseudo-logarithmic sample dimension. 109

5.11 Trec 8: Evolution of predictor weights to depth 1000— Queries 407, 436, and 450. Pseudo-logarithmic sample dimension. 110

6.1 Six Assumptions of IT-Similarity. 118

6.2 Trec 8: Rapid converge of Rankhedge weight history (uniform initial distribution) to corresponding history for initial debiased distribution. Figures a) and b) demonstrate qualitative similarity of evolving histories. Figure c) gives average ratio of system weights at sample depth. 123

6.3 TRECs 3, 5, and 6: Column a) System ranking based on net system similarity vs. actual TREC ranks. Column b) Rankhedge-m ranking (pool size m equivalent to Depth-1) vs. actual TREC ranks. 124

6.4 TRECs 7, 8, and 9: Column a) System ranking based on net system similarity vs. actual TREC ranks. Column b) Rankhedge-m ranking (pool size m equivalent to Depth-1) vs. actual TREC ranks. 125

7.1 Probabilistic vs. Greedy Sampling Strategy: a) precision of selected sample pool, b) weight history of best predictor, and c) w.l.c. of predictions on samples. 153

Chapter 1

Introduction

This thesis presents a simultaneous solution to several interrelated problems in Information Retrieval (IR) arising in the context of the effective use of ranked retrieval lists provided by a collection of retrieval systems. These problems—metasearch, system evaluation, and pooling— are usually treated as separate issues in the IR literature. But we demonstrate that each of the problems may be viewed as a particular aspect of the more general problem of combination of expert advice in an on-line setting.

The thesis proceeds along parallel practical and theoretical courses. We begin by introducing a practical solution to the three IR problems based on the Hedge on-line learning algorithm of Freund and Schapire [37]. We shall refer to our particular variant of the algorithm, adapted specifically to the context of ranked lists, as *Rankhedge*. As defined in this paper, Rankhedge is demonstrated to be an approximation of an optimal Bayesian on-line algorithm and thus approaches an optimal solution of the three IR problems.

As is often the case, the practical considerations associated with adapting an algorithm to a new domain provide an opportunity to engage in a thorough theoretical exploration of the underpinnings of the original algorithm, and we provide an exact information-theoretic interpretation of the Hedge algorithm and demonstrate optimality of our particular variation to the problem of on-line learning in the context of ranked lists. This information-theoretic analysis, in turn, suggests enhancements to the algorithm; and we present several theoretically grounded methods for improving performance, including a method for decorrelation of the initial distribution of predictors and an optimal method for actively sampling the instance space which employs a probabilistic technique to maximize expected learning or retrieval rates.

1.1 Challenges in On-line Retrieval

Our focus in information retrieval is on three interrelated problems associated with the use of ranked retrieval lists returned by a group of experts in response to a query or queries. The first problem, *metasearch*, may be described as the efficient fusion of ranked lists of documents in order to obtain a high-quality combined list. The on-line context for the metasearch problem, as developed in this thesis, generalizes the original definition of metasearch in which the algorithm is assumed to operate in the absence of feedback. Our on-line method is the first to efficiently incorporate user feedback into the metasearch process, and the Rankhedge learning algorithm provides a natural, theoretically consistent basis for incorporating iterative feedback.

Since the ability to effectively merge retrieval lists is dependent upon an algorithm's ability to estimate the quality of the underlying retrieval systems, our second IR problem, the accurate evaluation of retrieval systems, may be thought of as in some ways precedent to the on-line metasearch problem. We may define succinctly the problem of *system evaluation* as the accurate assessment of the quality of retrieval systems, given a limited number of relevance judgements.

Ability to incorporate feedback in the service of these two goals in an efficient manner demands a method for actively selecting samples for user feedback. In the IR literature, the collection of judged documents is often referred to as the *pool*, and *pooling* is most often described as the selection of document collections which provide a sufficient basis for the fair and accurate evaluation of retrieval systems. An alternate treatment is that pooling is the selection of document collections containing a large fraction of relevant documents— with the implicit assumption that these collections will provide a good basis for system evaluation. Thus, the third IR problem we consider, pooling, may be seen as a component of the algorithm employed to solve the metasearch and system evaluation problems.

1.1.1 Machine Learning and Information Theoretic Perspectives

In the field of Information Retrieval, these three problems—metasearch, system evaluation, and pooling— are usually treated as separate issues. Our work recasts the problems in a single unified framework and demonstrates that all three can be efficiently and effectively solved with a single technique based on the Hedge algorithm for on-line learning.

Hedge predictions of the labels associated with unlabelled instances are based on a weighted linear combination of predictions provided by a finite set of *experts*. As we shall demonstrate in the course of our analysis, if the update method of Hedge has access to a reasonable estimate of the posterior distribution over instance labels, the method may be defined in a manner which approximates an optimal Bayesian on-line algorithm. For metasearch purposes, this property translates to the fact that evidence from various experts is combined to generate predictions on the unlabelled instances in a manner which is consistent with a best estimate of the posterior probabilities of the labels as conditioned by the evidence from the labelled sample set. In our implementation of Rankhedge we define a method for estimation of the posterior probabilities of relevance of instances which is demonstrated to be both empirically and theoretically consistent with the expected behavior of the ranked predictions. Thus, Rankhedge may be viewed as a natural generalization of the metasearch process which incorporates on-line feedback into the retrieval process in a manner which approximates the optimal Bayesian solution.

To show Rankhedge's utility as a method for system evaluation, we shall present an information-theoretic analysis which demonstrates that the exponential update rule governing evolution of the weight distribution over predictors in Rankhedge adheres to desirable maximum entropy principles, resulting in a distribution which is defined solely by the performance of the predictors in response to the actual constraints associated with the pool of judged samples. The evolution of the distribution over predictors is consistent with a common notion of *sample complexity*, which may be defined in terms of the log *posterior volume* of possible configurations of labels on the unlabelled sample set. In the case of a weighted linear combination of finite length predictors, an analogous definition of the posterior function volume corresponds to the maximum Kullback Liebler distance of the distribution over predictors from the optimal distribution as defined by the labelling of the entire sample set.

Evolution of the complexity of a sample pool may be seen as the dual of the the evolution of the distribution over predictors, and the increase in complexity due to inclusion of a sample in the pool provides a measure of the expected instantaneous learning rate (or equivalently the constraint of the function volume) due to the sample. We define a method for active sample selection suggested by the optimal gambling paradigm used to prove information-theoretic optimality of Hedge, which may be used to maximize either the expected instantaneous learning rate or the expected instantaneous return (relevance) of the sample.

1.2 Contributions

The contributions of this thesis fall into three categories. In this section, we first discuss the practical results of the algorithm as applied to our basic problems of generalized metasearch, assessment of the quality of retrieval systems, and selection of efficient sample pools suitable for these tasks. Next, we delineate theoretical contributions which ground our Hedge algorithm for ranked lists and its modifications in an information-theoretic framework. And finally, we mention contributions which we shall refer to as contextual. The information-theoretic discussion of the Hedge on-line learning technique exposes the elemental connections between the Hedge technique and other methodologies from learning theory as well as from other disciplines such as statistical mechanics with the aim of broadening the context of applicability of Hedge related techniques.

1.2.1 Practical Contributions

The thesis presents a single theoretically grounded solution for several important problems in information retrieval which have, to date, been treated separately in the IR literature. In developing our algorithm, we establish a unified framework for addressing the three problems of generalized metasearch, pooling, and system evaluation. Our basic algorithm, Rankhedge, is a modified version of the Hedge algorithm which has been adapted to the context of the ranked lists returned by retrieval systems in the TREC contest. In making these and other modifications, we take a Bayesian viewpoint and base our Hedge loss function on an estimate of prior probability of relevance at rank. In conjunction with our theoretic explorations, we suggest several other useful algorithmic modifications to the basic algorithm.

1.2.2 Basic Results

The testbed for Rankhedge is the data from retrieval system submissions and hand-judged feedback of the annual TREC conference *retrieval track*. In experiments with the algorithm on multiple TREC data sets, the initial unmodified Rankhedge algorithm produced excellent results on a representative collection of TREC data (TREC's 3, 5, 6, 7, 8, and 9). As we shall demonstrate in Chapter Three, tests of the basic algorithm yielded the following results: (1) As an algorithm for metasearch, the technique combines ranked lists of documents in the absence of feedback in a manner whose performance equals or exceeds that of benchmark algorithms such as CombMNZ and Condorcet, and then dramatically improves on this performance through seamless integration of relevance judgements into the on-line metasearch process. (2) As an algorithm for pooling and system evaluation, our technique generates sets of documents which both include more relevant documents and result in more accurate system evaluations than standard techniques such as TREC-style depth pooling (i.e. the inclusion of all documents returned by any system to depth-n).

1.2.3 Modifications

Our most significant modifications of the Rankhedge algorithm consist of manipulations of the loss function. Two different models of expected relevance at rank are implemented via loss models defined in terms of functions on ranks. As shown in Chapter Five, the first method corresponds roughly to a logarithmic loss at rank and has the advantage in the IR context that the net loss is defined in terms of a familiar measure of the quality of retrieval systems (average precision). The second method employs a loss function defined directly in terms of the expected prior relevance at rank, resulting in a maximum entropy method which has a Bayesian interpretation, also discussed in Chapter Five. Both methods prove successful in practice, but results of the second method appear to approach optimality.

A main advantage of the Hedge algorithm over ad-hoc methods for on-line prediction is the fact that the exponential update method provides a theoretically grounded means for eliminating the bias-error associated with the overlap or lack of independence among the more generic retrieval systems. The second modification to our basic algorithm explores the effect of eliminating this bias error in the initial distribution of predictor weights. The method attempts to debias the initial predictor space using an information-theoretic similarity measure which is implicitly defined by the exponential update method. By eliminating correlations in the initial distribution over predictors, we reduce the volume of the initial predictor set, resulting in an improved worst case bound on performance of the algorithm. Results associated with this modified version of Rankhedge are presented in Chapter Six.

All Rankhedge results are dependent on our ability to efficiently select samples to add to the pool. A successful active learning method for selection of samples for pooling results in improved performance in metasearch and system evaluation. The basic Rankhedge algorithm employs a simple greedy selection method which attempts to maximize the instantaneous expected accuracy of samples selected for the pool. Chapter Seven introduces an alternative method for sample selection which probabilistically draws from a distribution over the unlabelled sample space with instances weighted in proportion to the expected relevance (to maximize instantaneous retrieval accuracy) or expected *risk* or error rate (to maximize the instantaneous learning rate). It is expected that, as a general technique, the sampling method will have applicability to a broad class of problems beyond the context of the Rankhedge algorithm.

1.2.4 Theoretical Contributions

To be able to address the various subtleties associated with the retrieval context, we embark on an information-theoretic examination of the original Hedge algorithm of Freund and Schapire. We establish an expression for the *exact* loss of the Hedge algorithm based on the Kullback Liebler Distance or relative entropy between successive instances of the distribution over predictors. A corollary to our proof demonstrates a derivation of a simple lower bound on Hedge loss as well as an upper bound which is independent of the evolution of the sample distribution and thus is consistent with the original Hedge analysis.

To adapt the Hedge algorithm to the retrieval context, we develop a rank-sensitive version of the algorithm, Rankhedge, in which rank effects are incorporated into the loss function via an estimate of prior probability of relevance at rank which is both empirically and theoretically justified. An examination of the aggregate behavior of the retrieval systems across queries demonstrates that the simple relevance at rank curve (based on the harmonic series) used by Rankhedge provides an accurate representation of observed behavior.

We examine bounds on algorithmic performance relative to several quantities of interest and demonstrate that the algorithm performs well, given reasonable definitions of loss and accuracy. We demonstrate via counterexample that a worst case bound on the loss of the algorithm measured in terms of the accuracy of the output list of selected samples is poor. However, the usual Hedge bound on the loss suffered by the algorithm (that is, a loss defined in terms of expected accuracy of the output list, rather than in absolute terms) is still available. Likewise, a worst case bound on the quality of the estimate of predictor accuracy is demonstrated, via counterexample, to be poor due to the effects of predictor bias; but an average case analysis provides an acceptable bound on estimated quality of the predictors.

To address the issue of lack of independence in the predictor set, the relationship of the ranked lists provided by predictors is defined via an information-theoretic similarity measure on the space of ranked lists. The log of the exponential loss associated with a list, given the expected relevance of the sample set as defined by a second list provides the basis for a similarity measure appropriate for debiasing the predictors. Both worst case and expected case measures are provided with the expected case measure corresponding to a familiar measure of similarity known as the dice coefficient [35].

Finally, a new method for active learning based on probabilistic sampling of the unlabelled sample set is presented. The method is based on a Bayesian or expected case notion of sample complexity as defined by the convergence of the posterior volume of functions to a target distribution due to the constraints inherent in a sample. As in our earlier analyses, an information-theoretic analysis of the evolution of the distribution over predictors given the probabilistic sampling scheme demonstrates optimal expected convergence toward the target distribution (the true probability of relevance of the predictors on the sample set).

1.2.5 Hedge in a Broader Context

In the course of our practical and theoretical explorations, we demonstrate several equivalences between the Hedge on-line algorithm and other methods for on-line learning. By viewing the exponential update rule as the limiting continuous approximation of a discrete proportional update process, we are able to place the Hedge algorithm in a Bayesian context; and we demonstrate that Hedge is, in fact, a generalization of the Bayesian on-line prediction technique of Cesa-Bianchi, et al. [19].

Our results in prediction and system evaluation derive from the duality of the learning process as viewed in terms of the evolution of the posterior distribution on the predictors, or alternatively, in terms of the sample complexity defined in terms of the constraint volume associated with an arbitrary sample set. The close relationship of an instantaneous snapshot of the Hedge algorithm to a familiar construction in statistical physics known as the Gibbs Distribution is noted, and this serves as an introduction to the notion of the inherent duality of the predictor and sample axes via an examination of Gibbs Distributions' companion entity, the Markov Random Field. Further examination of the learning algorithm via this dual representation yields the information-theoretic definition of a similarity measure on the space of predictors.

The duality of the evolving distribution over predictors and the informational complexity of the sample set is delineated in Bayesian terms by Haussler, et al. in [45] for the *lossless* context where the best predictor has an error rate $\epsilon = 0$. In the course of our development of active selection techniques, we demonstrate that the Bayesian view of Hedge implies that this Haussler learning method is actually an instance of the Hedge algorithm with β restricted to zero. This allows us to incorporate the sample complexity arguments of [45] into our discussion of active selection methods.

We also mention the connections of our algorithm to two other specific techniques from the literature of information retrieval. The application of Hedge to the context of ranked lists is demonstrated to be closely related to a technique for combining rankings due to Lebanon and Lafferty known as *Cranking* [55]. While Hedge produces a linear combination of predictor lists, the Cranking method uses gradient descent to determine the minimum energy configuration in the space of permutations. These two techniques may be seen as related methods for localizing the optimal configuration within the space of permutations, with both employing an exponential distance relationship, explicitly defined in Cranking and implicitly defined in Hedge.

1.3 Outline

In the chapters that follow, we first establish our three goals for the Rankhedge technique in the context of Information Retrieval. Chapter Two provides some background for the IR problems, examining some of the fundamental issues encountered by metasearch, system evaluation and pooling techniques as well as discussing several systems which have been developed in the three areas. In Chapter Three, we present a unified solution to the three problems based on the machine learning algorithm known as Hedge. Our modified version of the Hedge algorithm adapted to the context of ranked lists (Rankhedge) is presented, and results of the application of Rankhedge are demonstrated to be uniformly excellent across a representative collection of TREC conferences.

After presentation of our early results, we examine the theoretical basis for success of the algorithm. Chapters Four through Seven delve into the underlying theory of the Hedge algorithm and its application here to ranked lists. Taking an information-theoretic perspective, the connection between the Hedge algorithm and Bayesian methods for on-line prediction is established, allowing a rigorous examination of the on-line algorithm in terms of expected (as opposed to worst-case) behavior. The resulting analysis leads to several methods for enhancing behavior of the algorithm.

Chapter Four examines the specific modifications of the Hedge algorithm required to adapt it to the context of the combination of ranked lists. Examples of earlier machine learning algorithms directed toward combination of ranked-lists are provided, followed by a discussion of our Rankhedge scheme based on an explicit estimate of prior probability of relevance at rank. Both theoretical and empirical evidence for our definition of prior probability of relevance at rank is given, as well as an analysis of worst-case results for both quality of the output list and estimates of system accuracy. Finally, a discussion of the optimal selection of an exponential decay rate leads to a method for improving the basic algorithm by adapting the decay rate in accordance with the evolution of the probability of sample relevance with rank.

In Chapter Five, we establish the fundamental relationship of the Bayesian and Hedge methods for on-line learning by demonstrating that the Hedge algorithm reflects a continuous approximation of a discrete on-line algorithm corresponding to a proportional betting scheme. Continuing with the information-theoretic analysis indicated by this relationship to proportional gambling, we further establish exact bounds for the Hedge algorithm in terms of the relative entropy (Kullback Liebler distance) between successive distributions over predictors.

Chapters Six and Seven expand upon this information-theoretic framework and develop algorithmic enhancements to address the initial bias error of predictors as well as a method for active selection of samples. The method for debiasing predictors in Chapter Six proceeds directly from the development of an information-theoretic similarity measure on the space of ranked lists. The similarity measure has potential practical application beyond its direct uses in the context of the Hedge algorithm and we employ it to demonstrate a negative result in the area of pseudoevaluation of retrieval systems (the attempt to evaluate retrieval systems in the absence of user feedback).

Chapter Seven addresses the question of directed selection of samples or *active sampling* by switching the focus of interest in the learning process from the evolution of the distribution of predictors to that of the constraint volume associated with the sample set. Connections to sample complexity results of Haussler, et al. [45] are examined, and the Hedge algorithm is demonstrated to be a generalization of the Bayesian on-line learning algorithm of [45]. The algorithm of [45] is defined in the lossless context in which a predictor with zero error is guaranteed to exist. Hedge provides a practical and theoretically justifiable means for extending the results obtained in the lossless situation to the lossy context in which the best predictor has an error rate $\epsilon > 0$. A new probabilistic method for sample selection is presented, with unlabelled instances sampled in proportion to their expected risk (a property closely related to the sample complexity). Optimal convergence of the distribution over predictors given the probabilistic sampling method is demonstrated.

We conclude, in Chapter Eight, with an examination of possible areas for future research.

Chapter 2

IR Background

In this section, we provide the background for our discussion of the three basic Information Retrieval problems under consideration and examine some of the fundamental issues encountered by metasearch, system evaluation and pooling techniques. We describe earlier approaches to these problems and discuss systems which have been developed in each of the areas. The chapter begins with a discussion of pooling and system evaluation techniques, since the ability to accurately estimate the quality of a retrieval system is (at least implicitly) a prerequisite for generation of a quality fused list in the metasearch task. We proceed with an examination of metasearch, and the chapter moves from discussion of early ad-hoc techniques toward the Bayesian metasearch technique of Aslam and Montague which is an intellectual precursor to our current algorithm. We conclude with an examination of the *Cranking* method of Lebanon and Lafferty.

2.1 Pooling for System Evaluation

Collections of retrieval systems are traditionally evaluated by (1) constructing a test collection of documents (the "corpus"), (2) constructing a test collection of queries (the "topics"), (3) judging the relevance of the documents to each query (the "relevance judgments"), and (4) assessing the quality of the ranked lists of documents returned by each retrieval system for each topic using standard measures of performance such as Mean Average Precision (MAP). Much thought and research has been devoted to each of these steps in, for example, the annual TREC conference [43].

As it involves the manual intervention of a human agent, step (3) can quickly become excessively burdensome. For example, in TREC, participating systems return ranked lists of up to 1000 documents for 50 queries, based on estimated relevance of documents. Clearly, it would be impractical to assess the relevance of each document to each topic. With a document space on the order of millions, heuristics must be employed for limiting the size of pools of documents to be judged. In practice a relatively small subset of the documents is chosen, and the relevance of these documents to the topics is assessed. The method by which a subset of a document collection is chosen for purposes of system evaluation is referred to as the pooling technique.

Pools are used to evaluate retrieval systems in the following manner. The documents within a pool are manually judged to determine whether they are relevant to the given user query or topic. Documents not contained within the pool are *assumed* to be non-relevant. The ranked lists returned by the retrieval systems are then evaluated using standard measures of performance (such as Mean Average Precision) using this "complete" set of relevance judgments. Since documents not present in the pool are assumed non-relevant, the *quality* of the assessments produced by such a pool is often in direct proportion to the fraction of relevant documents found in the pool (its *recall*).

The current pooling technique employed by the annual TREC conference [43], is *Depth-n* pooling. In this method, the "pool" of documents to be judged is constructed by taking the union of the top n documents returned by each system in response to a given query. In TREC, $n = 100$ has been shown to be an effective cutoff in evaluating the relative performance of retrieval systems [43]. Both shallower and deeper pools have been studied [102, 43], both for TREC and within the greater context of the generation of large test collections [23].

Depth-n pooling has proven to be an effective technique since many of the documents relevant to a topic will appear near the top of the lists returned by (quality) retrieval systems. These relevant documents will be judged and used to effectively assess the performance of the collected systems. In addition, the Depth-n pooling method is a "fair" method for establishing system quality, given a restricted sample pool, since all systems' lists are guaranteed to be sampled to equivalent depths. The results of the method are also relatively "complete" in terms of the set of relevant documents retrieved, given a sufficient depth of sampling.

Judging all documents appearing in the depth-100 pool is still quite expensive, however. For example, in TREC 8 86,830 manual relevance judgments were used to assess the quality of the retrieved lists submitted by 129 systems in response to 50 topics [95] . An inventory of the number of documents at various pool depths for multiple TRECs is shown in Table 2.1.

Both "blind" pooling techniques (techniques using no relevance judgements) and on-line pooling techniques— where user feedback is solicited to direct the pooling process— have been proposed as alternative solutions to attempt to identify relevant documents as quickly as possible.

	TREC					
Pool	3	5	6	7	8	9
Depth	$n = 40$	$n = 82$	$n = 79$	$n = 103$	$n = 129$	$n = 105$
1	19	38	38	32	40	38
2	39	68	67	55	69	68
3	47	98	95	76	95	95
4	60	126	120	95	119	120
5	73	153	146	114	144	146
6	85	181	172	134	167	171
7	96	208	197	152	191	196
8	107	234	221	170	215	220
9	118	262	246	189	238	245
10	129	288	271	207	260	269
15	183	418	393	297	379	388
20	235	543	513	389	494	502
30	336	791	743	571	717	726
40	436	1034	969	754	939	942
50	531	1273	1191	936	1155	1178
60	626	1509	1410	1114	1366	1366
70	718	1745	1629	1299	1574	1575
80	811	1978	1845	1486	1777	1782
90	903	2206	2058	1675	1978	1983
100	995	2434	2271	1860	2176	2184

Table 2.1: The size of the pool (averaged over 50 queries) for various pool depths if the pooling is performed TREC-style. Here n is the number of input systems contributing to the data set.

2.1.1 Pooling without Relevance Judgements

In a recent work [84], Soboroff, et al. presented a method for *pseudoevaluation* of retrieval systems which probabilistically selects documents from the pool of documents returned by systems in a TREC retrieval track according to various models of how relevant documents should occur in the document pool. The selected documents were assumed to be relevant and formed the basis for a *pseudo-qrel* (a qrel being a vector of relevance judgements) against which the performance of the retrieval systems could be evaluated. Different sampling methods included viewing the pool of documents as a set versus as a multiset and sampling at various pool depths. The resultant rankings proved to be well

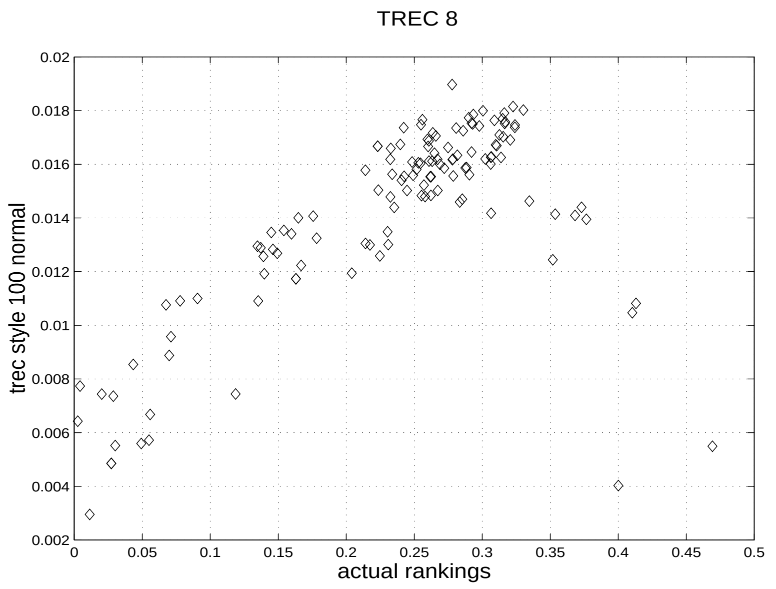


Figure 2.1: “TRECstyle 100 normal” method vs. actual rankings— TREC 8.

correlated with actual TREC rankings.

As noted in [84], the pseudoevaluations suffered from a common phenomenon, further explored by Aslam and Savell in [11]. While the bulk of the systems were classified correctly, the best systems in terms of precision (also most important for metasearch purposes) were consistently ranked with the poor performers (see Figure 2.1). The intuition behind this phenomenon is, of course, that the better systems are doing something significantly different from the more generic systems in the pack. An example of this effect may be seen in Figure 2.1 which compares the actual TREC-8 rankings to rankings produced by Soboroff’s “TRECstyle 100 normal” technique, which corresponds to a random sampling of the *set* (unique documents included once) of all documents returned to depth-100 by all systems. In this sampling technique, the ranking phenomenon is a result of correlations among non-relevant items in the pool, which tend to enhance retrieval rates of similar systems. The results are consistent with those of the other sampling methods and of pseudoevaluation techniques in general which tend to produce an enhanced ranking of the generic systems and to underestimate the quality of the best predictors.

The susceptibility of pseudoevaluation methods to a “tyranny of the masses” effect or “bias error” introduces a theme which is common to methods seeking to limit pool sizes for system evaluation as well as to metasearch methods operating in the absence of relevance judgements. That is, methods reliant upon document popularity as an indicator of relevance, while capable of weeding out the worst performers, also tend to weed out the best.

2.1.2 On-line Pooling with Priority Queues

An interesting and surprisingly effective ad-hoc technique for dealing with these sorts of errors by utilizing on-line relevance judgements is presented in Cormack, et al. [23]. The algorithm employs a priority queue to keep track of systems which have most recently yielded a relevant document.

In the algorithm presented in Figure 2.2, systems are initially placed in random order into a priority queue. At each iteration, the system with highest priority is withdrawn from the heap. Documents from this system’s list which have not previously been judged are examined in descending order. If a relevant document is found, the system’s priority is set to the maximum. Documents are drawn until a non-relevant document is found (whether or not it has been previously judged), and then the system is returned to the heap with its priority decremented by one. Figure 2.3 shows the relative effectiveness of the priority queue method in discovering relevant documents relative to the Depth-n pooling technique. This figure provides TREC 8 results, but the method proved similarity effective across all TREC’s tested.

A brief analysis demonstrates how this simple ad-hoc algorithm greatly increases the rate of relevant documents found over normal Depth-n pooling. With

```
ALGORITHM: On-line Pooling via Priority Queue():

1 Randomly place all systems on heap with max priority.
2 WHILE unjudged documents remain:
3       Remove system s from top of heap.
4       WHILE any document in list of s:
5           IF unjudged document:
6               Assign document relevance.
7           IF document relevant:
8               Set priority of s to maximum.
9           ELSE:
10              Decrement priority of s.
11              Return s to heap.
11              BREAK.
```

Figure 2.2: On-line Pooling Algorithm (Cormack et al. [23]).

.

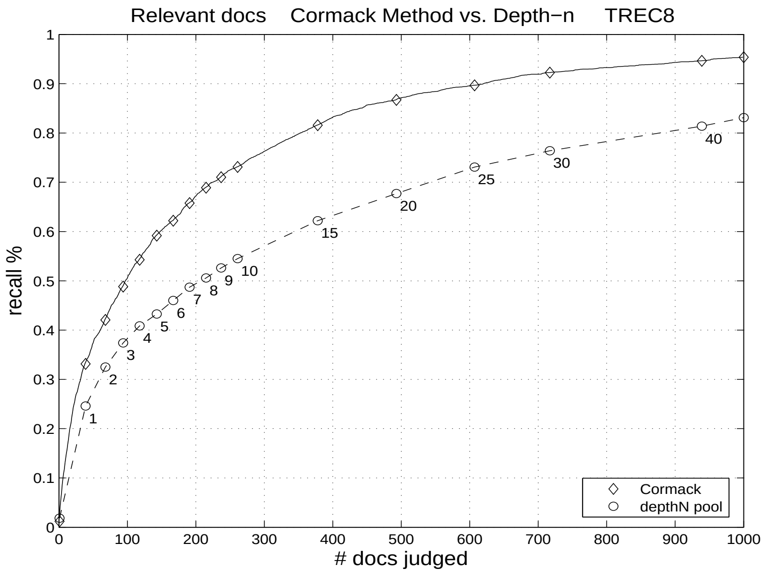


Figure 2.3: Percent of total relevant documents discovered: On-line pooling with priority queue vs. Depth-n pools— TREC 8.

N as the number of systems, d_{rel_i} as the number of relevant documents returned by an arbitrary system s at ranks 1 to i and d_{nrel_i} as the corresponding number of nonrelevant documents, one may estimate the expected rate of discovery of relevant documents returned by system s.

Given that: a) a system which has recently drawn a non-relevant document will have to wait on the order of N draws before returning to the top of the heap, and b) there is no waiting when a relevant document is drawn— the cost of seeing the top i documents of an arbitrary system s is approximately $N * d_{nrel_i}$ for N of significant size. Thus, the expected rate of discovery of the d_{rel_i} relevant documents occuring in the first i documents returned by s is approximately $d_{rel_i}/(N * d_{nrel_i})$. In other words, a document returned by system s at rank i is sampled at a rate proportional to the expected odds of relevance of documents returned by s at ranks 1 to $i-1$. As demonstrated in Figure 2.2, this technique of sampling a system list at a rate proportional to the odds of relevance proves to be quite effective, and we shall see in future chapters that the technique of proportional sampling at expected rates of accuracy serves also as the foundation of the Hedge based methods presented in this thesis.

2.2 Metasearch

Metasearch is the well-studied process of fusing the ranked lists of documents returned by a collection of systems in response to a given user query into a single ranked list. The use of data fusion to combine retrieval results has been an active area of study in IR since 1972, when Fisher and Elchesen [32] showed that document retrieval results could be improved by combining the results of two Boolean searches: one over the title words of documents, and one over manually-assigned terms. This early work was followed by many more extensive studies, [12, 13, 14, 25, 33, 48, 56, 57, 61, 69, 70, 79, 83, 86, 87, 89, 91, 92, 93, 94, 96].

For a survey of the earlier literature, see [26]. Direct ancestral influences of this thesis may be found in [5, 7, 63, 64, 65, 55].

In general, metasearch algorithms produce quality ranked lists of documents by fusing the ranked lists provided by a collection of underlying systems. Given ranked lists produced by good but sufficiently different underlying systems, these techniques can produce fused lists whose performance exceeds that of any of the underlying lists. Likewise, given ranked lists produced by possibly correlated systems of varying performance, these metasearch techniques will most often produce fused lists whose performance exceeds that of the "average" underlying list but which rarely exceeds that of the best underlying list.

The great majority of metasearch techniques involve formulating a weighted linear combination of the ordered lists of documents returned by the underlying retrieval systems using one of a number of diverse document scoring methods. In this context, the relationship to system evaluation methods is clear, since a reasonable estimate of predictor quality provides a valuable input to the document weighting function. In the following sections, we examine the details of several metasearch methods.

2.2.1 CombMNZ, CombSum, and CombANZ

The benchmark technique CombMNZ [34, 81, 57] is one of a group of adhoc methods— CombMNZ, CombSum, and CombANZ— defined in terms of weighted combinations of the normalized relevance scores given to each document by the underlying systems. Central to these techniques is the implicit assumption that a document's relevance score provided by a retrieval system corresponds to a system's best estimate of the document's relative probability of relevance.

Fox and Shaw [34] present three different methods which attempt to bias the simple weighted linear combination of relevance scores by weighting each document d more or less heavily according to the number of systems that returned the document. With n_d corresponding to the number of systems returning document d, the general formula for relevance is:

$$rel(d) = n_d^\gamma \sum_i rel_i(d)$$

with $\gamma \in \{-1, 0, 1\}$.

When $\gamma = -1$, the system is equivalent to the average relevance over systems that returned d, and is referred to as "CombANZ". When $\gamma = 0$, the result is the sum of the relevance scores over all systems or "CombSum". Assuming, as in the paper, a similarity of zero for documents not returned, "CombSum" is simply the average relevance over all systems. Finally, with $\gamma = 1$, the result is "CombMNZ" (Multiply-by-number-Non-Zero). In this case, multiplication by the number of systems returning document d tends to bias the results toward documents with broader support in the predictor set.

In the original work, Fox and Shaw find CombSum to be slightly more effective than CombMNZ. But later experiments such as those by Lee [57] have tended to favor CombMNZ. Due to the ease of implementation and relative effectiveness in the absence of relevance judgements, CombMNZ scores are often cited as a baseline for comparison by other metasearch techniques. A variation of CombMNZ (rCombMNZ) which uses ranks rather than relevance scores is also popular and produces comparable results.

2.2.2 Voting Methods: Borda Fuse and Condorcet

Several algorithms by Aslam and Montague [7, 65] approach the metasearch problem as a multi-candidate election where the documents are candidates and the systems are voters expressing preferential rankings among the candidates. These techniques are based on two *fair* methods for determining the winner of a multi-candidate election. These fair voting algorithms were originally defined by two French theorists Borda and Condorcet in the 1780's [18, 29]. The branch of decision theory which has since evolved to address questions of cooperative decision making is known as "Social Choice Theory" [17, 52, 73, 66].

The first technique, Borda-Fuse [7] is based on a *positional* method— a method which is defined in terms of a function on the ranks of the candidates— known as Borda-Count. The Borda Count works as follows. Each voter ranks a fixed set of c candidates in order of preference. For each voter, the top ranked candidate is given r points, the second ranked candidate is given $r-1$ points, and so on. If there are some candidates left unranked by the voter, the remaining points are divided evenly among the unranked candidates. The candidates are ranked in order of total points, and the candidate with the most points wins the election. As shown by [76, 77], Borda Count is optimal in the sense that it satisfies all of the symmetry properties that one would expect of a reasonable election strategy.

To adapt Borda Count to the metasearch problem, a simple weighting scheme multiplies the points assigned to a document by system s by the system weight w_s. Documents are then ranked by weighted linear combination $score(d) = \sum_s w_s * (N - r_{(s,d)})$, with N corresponding to the number of candidates ranked by system s.

A second voting algorithm, the Condorcet method [65], achieves slightly better performance. It is based on a *majoritarian* algorithm in which an ordering of candidates is achieved through a series of pairwise runoff elections. The Condorcet winner among a series of candidates is the candidate that wins (or ties) in every possible pairwise majority contest. An important result from Social Choice Theory known as May's theorem states that in a two-candidate election, "majority voting is the only method that is anonymous (equal treatment of voters), neutral (equal treatment of the candidates), and monotonic (more support for a candidate cannot jeopardize its election)". Since the Condorcet winner is the candidate that wins or ties in every possible pairwise majority contest, May's theorem lends strong support to the selection of Condorcet as a voting algorithm. Cycles in the Condorcet graph are possible and are often treated in the literature as paradoxes. However, Aslam and Montague point out that these cycles are a natural artifact of the complexity of the voting profile. In the context of metasearch, it is possible to treat these cycles as ties and this is the method implemented in [65].

Weighting schemes may be incorporated into the Condorcet paradigm by running the comparisons after system weights have been applied to the individual candidate scores. In this case, the value associated with system s's scoring of candidate d is defined in terms of the system weight w_s and the rank assigned to d by the system ($c_{(s,d)}$) so that: $score(s,d) = w_s * c_{(s,d)}$.

Though the Condorcet algorithm is combinatorial, the expected outcomes of the pairwise contests are similar to those of Borda Count; since, for any document d_x returned by a system s, the expectation that d_x will win a runoff election against another randomly chosen candidate d_y ranked by the same system is $Pr(d_x > d_y | c_{(s,d_x)}) = (N - c_{(s,d_x)})/N$, where N corresponds to the number of candidates ranked by the system. For ranking purposes, this is equivalent to a Borda Count weighting of $r, r-1, \cdots, r-r$. Thus, the majority algorithm may be seen as a more sophisticated manner of combining votes, but with the expected outcome still determined by $score(s,d) = \sum_s w_s * c_{(s,d)}$.

2.2.3 The Probabilistic Model

A Bayesian technique by Aslam and Montague [5, 7] attempts to calculate the metasearch ranking by using odds of relevance as determined directly from the estimated probability of relevance at rank under naive Bayes assumptions.

Given the ranked lists of documents returned by n retrieval systems, let $r_i(d)$ be the *rank* assigned to document d by retrieval system i (a rank of ∞ may be used if document d is not retrieved by system i). This constitutes the *evidence of relevance* provided to the metasearch strategy concerning document d. For a given document, let

$$\begin{aligned} P_{\text{rel}} &= \Pr[\text{rel}|r_1, r_2, \ldots, r_n] \text{ and} \\ P_{\text{irr}} &= \Pr[\text{irr}|r_1, r_2, \ldots, r_n] \end{aligned}$$

be the respective probabilities that the given document is *relevant* and *irrelevant* given the rank evidence $r_1, r_2, \ldots, r_n$. The Bayes optimal decision rule for determining the relevance of a document dictates that a document should be assumed relevant if and only if $P_{\text{rel}}/P_{\text{irr}} \geq \tau$ for some threshold τ chosen so as to optimize the expected loss suffered if incorrect. Since we are interested in *ranking* the documents, the algorithm need only compute the *odds* of relevance

$$O_{\text{rel}} = P_{\text{rel}}/P_{\text{irr}}$$

and rank documents according to this measure.

As shown in the derivation in [7], under naive Bayes assumptions a relevance measure sufficient for ranking of documents is given by:

$$rel(d) = \sum_i \log \frac{\Pr[r_i(d)|\text{rel}]}{\Pr[r_i(d)|\text{irr}]}, \tag{2.1}$$

with $\Pr[r_i|\text{rel}]$ the probability that a relevant document would be ranked at level r_i by system i and $\Pr[r_i|\text{irr}]$ the probability that an irrelevant document would be ranked at level r_i by system i. As implemented in [7], the posterior odds of rank given document relevance are determined by examination of the per-system statistics generated by the TREC eval program on a randomly chosen subset of queries.

In the following chapters, we shall eliminate the need to directly sample the probabilities of relevance at rank by demonstrating the existence of an empirically and theoretically grounded univariate family of rank relevance curves based on the harmonic series. While the Hedge algorithm is not usually considered to be a Bayesian method, we shall establish in the course of our information-theoretic analysis of the algorithm that Rankhedge (with an appropriate definition of loss function) does, in fact, constitute a continuous approximation of a discrete Bayesian on-line algorithm, and thus may be seen as a descendent of this earlier Bayesian technique.

2.3 Cranking

Finally, as an alternative to the various methods based on linear combinations of classifiers, the Cranking Method [55] provides an interesting technique for generating metasearch lists which is somewhat immune to the problems of bias error of the other techniques. The Cranking Method seeks to build probability distributions over rankings of labels— leading to conditional probability models on permutations of the labels. The method attempts to determine the *maximum a posteriori* (MAP) hypothesis in the space of permutations of document rankings. That is, given the evidence of a training set consisting of pairs $(\pi^{(i)}, \sigma^{(i)})$ where $\pi^{(i)}$ is a target ranking of instances i and $\sigma^{(i)}$ is the set of rankings returned by the underlying retrieval systems, the method tries to find the MAP hypothesis as fixed by this evidence.

The method utilizes the Mallows conditional ranking model $\mathcal{M}_d(\theta, \sigma)$, where $\theta \in \mathbb{R}$ is a dispersion parameter and $\sigma \in \mathcal{S}_n$ is a location parameter from the symmetric group of order n. The model has the exponential form

$$p(\pi|\theta, \sigma) = e^{\theta d(\pi,\sigma) - \psi(\theta,\sigma)}$$

and $d(\pi, \theta)$ may be any one of a number of distance metrics $d : \mathcal{S}_n \times \mathcal{S}_n$ such as Kendall's τ (the minimum number of adjacent transpositions needed to bring

π to σ), rank correlation ($R(\pi, \sigma) = \sum_{i=1}^{n} (\pi(i) - \sigma(i))^2$), or Spearman's foot rule ($F(\pi, \sigma) = \sum_{i=1}^{n} |\pi(i) - \sigma(i)|$) and ψ is the cumulant function $\psi(\theta, \sigma) = \log \sum_{\pi \in \mathcal{S}_n} \exp(\theta \cdot d(\pi, \sigma))$.

The Mallows model is generalized to allow multiple instances with each instance associated with a possibly different set of rankings. With $\sigma_j \in \mathcal{S}_n$ and $\theta_j \in \mathbb{R}$ for $j = 1, \cdots, k$, the conditional model is defined by:

$$p(\pi|\sigma, \theta) = \frac{1}{Z(\theta, \sigma)} e^{\sum_{j=1}^{k} \theta_j d(\pi, \sigma_j)}$$

A ranked list may be generated from the model by evaluating the expected rank of label $y^{(i)}$ associated with instance $x^{(i)}$ as follows:

$$\begin{aligned} E[\pi(y^{(i)})|\theta, \sigma^{(i)}] &= \sum_{k=1}^{n} kp(\pi(y) = k|\theta, \sigma^{(i)}) \\ &= \sum_{k=1}^{n} k \sum_{\pi \in \mathcal{S}_{\delta_k \pi_k^{(i)}}} p(\pi|\theta, \sigma^{(i)}) \end{aligned}$$

where $\mathcal{S}_{\delta_k \pi_k^{(i)}}$ is the coset of permutations which fix $y^{(i)}$ in position k.

In the Cranking method, the parameters to be set are the θ_j, which are found directly via maximum likelihood estimation. Since the method finds weights θ which best fit the input permutations, it is relatively immune to the bias problems encountered in the other metasearch methods. The fundamental insight which is the basis of the Cranking method, that a distance measure may be defined on the space of permutations enabling learning in this space will be further examined in Chapter Six in which a similarity measure consistent with the exponential update method of Rankhedge is defined and employed to decorrelate the predictor set.

Chapter 3

A Unified Model

In this chapter, we present a unified model which addresses the related problems of metasearch, pooling and system evaluation (see also [8, 9]). In the context of this unified framework, we present an algorithm, *Rankhedge*, which is a modified version of the familiar Hedge algorithm for on-line learning adapted to the setting of ranked lists. Rankhedge may be thought of as a principled generalization of the metasearch technique to the on-line setting, which with an appropriate definition of prior probability of relevance at rank elicits a Bayesian interpretation. In addition, the Rankhedge algorithm incorporates an active learning component for sample selection to enhance learning rates.

The problem of bias error which is endemic to most of the common metasearch algorithms and the "pseudoevaluation" methods examined in the previous chapter are naturally and systematically addressed by the Rankhedge algorithm's incorporation of on-line relevance judgements. As with the usual form of the Hedge algorithm, theoretical bounds for Rankhedge are available, guaranteeing rapid convergence of the algorithm's performance to that approaching the best weighted linear combination of the underlying systems. While not the first instance of an application of computational learning methods to the context of ordered lists, our algorithm differs from earlier adaptations of Hedge and Boosting algorithms [21, 36] by establishing a theoretically and empirically grounded loss function, resulting in an approximately Bayesian or *pseudo-Bayesian* algorithm.

In this Chapter, we introduce the framework for solution of our three stated problems in IR (metasearch, pooling, and system evaluation) via the Rankhedge on-line algorithm and follow with a discussion of an early implementation of Rankhedge which employs a loss function based on the familiar Average Precision measure of retrieval list accuracy. The relationship of our Rankhedge algorithm to earlier techniques will be examined in detail in Chapter Four and connections to Bayesian and Maximal Entropy methods for on-line learning and metasearch will be presented in Chapter Five.

3.1 The Modified Hedge Algorithm

The results that follow clearly demonstrate the algorithm's effectiveness as a metasearch engine. In the absence of feedback, the metasearch performance of our technique most often equals or exceeds that of benchmark techniques such as CombMNZ and Condorcet (see Table 3.1). In experiments using TREC data, Rankhedge effectively equalled the performance of CombMNZ on five out of six data sets tested (TRECs 3, 5, 6, 7, 8, and 9) and significantly outperformed

TREC	MNZ	COND	Rankhedge-0	$\%MNZ$	$\%COND$
3	0.423	0.403	0.418	-1.2	$+3.7$
5	0.294	0.307	0.309	$+5.1$	$+0.6$
6	0.341	0.315	0.345	$+1.2$	$+9.5$
7	0.320	0.308	0.323	$+0.9$	$+4.9$
8	0.350	0.343	0.352	$+1.4$	$+2.6$
9	0.351	0.348	0.358	$+1.9$	$+2.9$

Table 3.1: Rankhedge-0 Method (Rankhedge Metasearch List at depth zero) vs. Metasearch Techniques CombMNZ and Condorcet.

CombMNZ on TREC 5. The algorithm consistently outperformed Condorcet on each of the data sets tested, significantly so on TRECs 6 and 7. In the presence of relevance judgements, Rankhedge rapidly and effectively "learns" how to fuse the underlying ranked lists, often outperforming the best underlying system after only a handful of relevance judgments.

As a pooling technique, the method likewise succeeds in generating efficient pools for effective evaluation of retrieval systems. As the algorithm learns which documents are likely to be relevant, these documents can then be selected for judgement and added to the pool, and their relevance judgments can be used as feedback to improve the learning process— thus generating more relevant documents in subsequent rounds. The quality of the pools generated can be judged in two ways: (1) At what rate are relevant documents found (recall percentage as a function of total judgments)? (2) How well do these pools evaluate the retrieval systems (score or rank correlations vs. "ground truth")? In our experiments using TREC data, Rankhedge found relevant documents at rates *nearly double* that of benchmark techniques such as TREC-style depth pooling. When used to evaluate the underlying retrieval systems, these Rankhedge pools performed much better than TREC-style depth pools of an equivalent size (as measured by Kendall's τ rank correlation, for example). In addition, these Rankhedge pools seemed particular effective at properly evaluating the best underlying systems, a task which is difficult to achieve using small pools as was demonstrated in the discussion of "pseudoevaluation" techniques in Chapter Two.

3.1.1 Intuition

The intuition for our algorithm can be described as follows. Consider a user who submits a given query to multiple search engines and receives a collection of ranked lists in response. How would the user select documents to read in order to satisfy his or her information need? In the absence of any knowledge about the quality of the underlying systems, the user would probably begin by selecting some document which is "highly ranked" by "many" systems. Such a document has, in effect, the collective weight of the underlying systems behind it. If the selected document were relevant, the user would begin to "trust" systems which retrieved this document highly (i.e., they would be "rewarded"), while the user would begin to "lose faith" in systems which did not retrieve this document highly (i.e., they would be "punished"). Conversely, if the document were non-relevant, the user would punish systems which retrieved the document highly and reward systems which did not. In subsequent rounds, the user would likely select documents according to his or her faith in the various systems in conjunction with how these systems rank the various documents; in other words, the user would likely pick documents which are *ranked highly* by *trusted* systems.

How can the above intuition be quantified and encoded algorithmically? Such questions have been studied in the machine learning community for quite some time and are often referred to as "combination of expert advice" problems. One of the seminal results in this field is the Weighted Majority Algorithm due to Littlestone and Warmuth [60]. In this work, we use a generalization of the Weighted Majority Algorithm called Hedge due to Freund and Schapire [37].

Hedge is an on-line allocation strategy which solves the problem of systematic combination of expert advice. The original algorithm of Freund and Schapire is described in Figure 3.1. Hedge is parameterized by a tunable learning rate $\beta \in [0, 1]$, and in the absence of any *a priori* knowledge, begins with an initially uniform "weight" w_i^1 for each expert i (in our case, $w_i^1 = 1 \ \forall \ i$). The relative weight associated with an expert corresponds to one's "faith" in its performance.

For each round $t \in \{1, \ldots, T\}$, these weights are normalized to form a probability distribution $\mathbf{p}^t$ where

$$\mathbf{p}_i^t = \frac{w_i^t}{\sum_j w_j^t},$$

```
Algorithm Hedge($\beta$)

Parameters:

        number of systems $N$.
        initial weight vector $\mathbf{w}^1 \in [0,1]^N$
        number of trials $T$.
        $\beta \in [0,1]$.

Do for $t = 1, 2, \ldots, T$.

    1. Choose allocation $\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$.

    2. Receive loss $\ell^t \in [0,1]^N$ from environment.

    3. Suffer loss $\mathbf{p}^t \cdot \ell^t$.

    4. Set the new weight vector to be $w_i^{t+1} = w_i^t \beta^{\ell_i^t}$.
```

Figure 3.1: Hedge Algorithm.

and one places $\mathbf{p}_i^t$ "faith" in system i during round t.

This "faith" can be manifested in any number of ways, depending on the problem being solved. If the underlying experts are making predictions about which stocks will rise in the next trading day, one might invest one's money in stocks according to the weighted predictions of the underlying experts. If a stock goes up, then each underlying expert i which predicted this rise would receive a "gain," and the investor would also receive a gain in proportion to the money invested, p_i^t. If the stock goes down, then each underlying expert i which predicted a rise would suffer a "loss," and the investor would also suffer a loss in proportion to the money invested. This is encoded in Hedge via the *mixture loss*. In each round t, expert i suffers a *loss* ℓ_i^t, and the algorithm suffers a weighted average (*mixture*) loss of $\sum_i p_i^t \ell_i^t$. For the purposes of the Hedge algorithm and its analysis, it is assumed that the losses and/or gains are bounded so that they can be mapped to the range $[0,1]$.

The Hedge algorithm updates its "faith" in each expert according to the losses suffered in the current round, $w_i^{t+1} = w_i^t \beta^{\ell_i^t}$. Thus, the greater the loss an expert suffers in round t, the lower its weight in round $t+1$. The "rate" at which this change occurs is dictated by the tunable parameter β.

Over time, the "best" underlying experts will develop the "highest" weights, and the cumulative (mixture) loss suffered by Hedge will be not much higher than that of the best underlying expert. Specifically, Freund and Schapire show that if $L_i = \sum_t \ell_i^t$ is the cumulative loss suffered by expert i, then the cumulative (mixture) loss suffered by Hedge is bounded by

$$L_{\text{Hedge}(\beta)} \leq \frac{\min_i\{L_i\} \cdot \ln(1/\beta) + \ln N}{1-\beta} \tag{3.1}$$

where N is the number of underlying experts.

3.1.2 Rankhedge for On-line Metasearch and System Evaluation

To adapt the Hedge algorithm to on-line metasearch, it is sufficient to define a measure of document value at rank, a loss function and a pooling method. Pseudocode for a generic Rankhedge algorithm may be seen in Figure 3.2. The first three overloaded methods in the figure provide a means for determining the Hedge loss associated with a sample. The *rel*() method corresponds to a user supplied judgement for document d. The *value*() method shown in Figure 3.3 provides a measure of loss amplitude at rank and is used in conjunction with *rel*() by *loss*() (see also Fig. 3.3) to determine the Hedge loss associated with a particular document d. The *poolDocuments*(), *generateMetasearchList*(), and *evaluateSystems*() methods (Figures 3.4, 3.5, and 3.6) correspond, respectively, to our pooling strategy for selecting a document for judgement, our strategy for generating a metasearch list given the vectors of system weights, ranked lists, and previously pooled documents, and our chosen method for evaluating system accuracy given the document pool.

```
Algorithm: On-line Metasearch()

Parameters:
    Inputs:
        document space D.
        retrieval systems S.
        number of systems $N$.
        number of trials $T$.
        initial weight vector $\mathbf{w}^1 \in [0,1]^N$.
        $\beta \in [0,1]$.
        span of relevance at rank calculations $T'$.
        decay constant of prior rel at rank $c_H$.
    Outputs:
        pool list for output P.
        metasearch list for output M.
        system ordering for output $\mathbf{S}'$.

Methods:
        $b = \mathbf{rel}(d)$.
        $v = \mathbf{value}(r, c_h, T')$.
        $\mathbf{L} = \mathbf{loss}(d, \mathbf{S}, value(), rel())$.
        $d = \mathbf{poolDocuments}(\mathbf{D}, \mathbf{S}, \mathbf{w}, value())$.
        $\mathbf{M} = \mathbf{generateMetasearchList}(\mathbf{D}, \mathbf{S}, \mathbf{w}, \mathbf{P}, T, value())$.
        $\mathbf{S}' = \mathbf{evaluateSystems}(\mathbf{S}, \mathbf{P}, rel())$.

Initialize:
    For each query:
        1. Choose allocation $\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$.

        2. $\mathbf{M} = \mathbf{generateMetasearchList}(\mathbf{D}, \mathbf{S}, w, \mathbf{P}, T, value())$.

Do for $t = 1, 2, \ldots, T$.
    For each query:
        1. $\mathbf{P}[t] = d = \mathbf{poolDocuments}(\mathbf{D}, \mathbf{S}, \mathbf{w}, value())$.

        2. $\mathbf{L}^t = \mathbf{loss}(d, \mathbf{S}, value(), rel())$.

        3. Suffer loss: $\mathbf{p}^t \cdot \mathbf{L}^t$.

        4. Set the new weight vector: $w_i^{t+1} = w_i^t \beta^{\mathbf{L}_i^t}$.

        5. Choose allocation $\mathbf{p}^t = \frac{\mathbf{w}^t}{\sum_{i=1}^N w_i^t}$.

        6. $\mathbf{M}^t = \mathbf{generateMetasearchList}(\mathbf{D}, \mathbf{S}, \mathbf{w}, \mathbf{P}, T, value())$.

    Merge results and:

        $\mathbf{S}' = \mathbf{evaluateSystems}(\mathbf{S}, \mathbf{P}, rel())$.
```

Figure 3.2: Rankhedge Algorithm for On-line Metasearch.

```
Method: value $(k, c_H, T')$

Parameters:
        rank $k$.
        decay constant $c_H$.
        span of relevance at rank calculations $T'$.

    For Rankhedge algorithm 1 (log prior):

        1. $\mathrm{val}'(k) = \sum_{r=k..T'} \frac{1}{(1+(c_H(r-1))}$.

        2. return $\mathrm{val}(k) = \frac{\mathrm{val}'(k)}{\sum_{r=1..T'} \mathrm{val}'(r)}$,

    For Rankhedge algorithm 2:

        1. $return\,\mathrm{val}'(k) = \frac{1}{(1+(c_H(r-1))}$.
```

```
Method: loss $(d, \mathbf{S}, value(), rel())$

Parameters:
        document $d$.
        abs value of loss at $r = value(r)$.
        relevance judgement at $d = \mathrm{rel}(d)$.

    1. $\forall s \in \mathbf{S} : L_s = \frac{1}{2} - \frac{1}{2}^{\mathrm{rel}(d)} \cdot value(rank_{(d,s)})$,

        with $\mathrm{rel}()$ being a binary indicator
        function defining relevance of $d$.

    2. return L.
```

Figure 3.3: RankHedge Subroutines: value() and loss().

Value and Loss Functions

On a per query basis, each underlying retrieval system is an "expert" providing "advice" about the relevance of various documents to a given query. The estimate of relevance that a system s attributes to a document d at rank r is determined by the function $value(r)$. For the initial experiments presented in this chapter, the value of a document at rank was designed to reflect the document's incremental contribution to a measure of system performance which is similar to the Average Precision measure (AP) commonly used in IR applications.

Let $\text{Precision}(r) = \sum_{s=1}^{r} \text{rel}(d_s)/r$, with $\text{rel}(d_s)$ an indicator function for the relevance of d_s (i.e., 1 if d_s is relevant and 0 if it is not). Then the Average Precision is defined as the average of $\text{Precision}(r)$ taken only at ranks corresponding to relevant documents. In other words, making the usual assumption that the precision of all unretrieved relevant documents is zero, the expression for the Average Precision for a list of length N with R corresponding to the total number of relevant documents returned for the query, is given by:

$$\begin{aligned} AP &= \frac{1}{R} \cdot \sum_{i:\text{rel}(i)} \text{Precision}(i) \\ &= \frac{1}{R} \sum_{r=1}^{N} \text{rel}(i) \cdot \text{Precision}(i) \\ &= \frac{1}{R} \cdot \sum_{i=1}^{N} \text{rel}(i) \sum_{j=1} \text{rel}(j)/i \\ &= \frac{1}{R} \cdot \sum_{1 \le j \le i \le N} \frac{1}{i} \cdot \text{rel}(i) \cdot \text{rel}(j) \end{aligned}$$

As demonstrated in [10], the average precision effectively assigns an implicit weight $\frac{1}{R \cdot i}$ to each *pair* of ranks (i, j), for all $1 \le j \le i \le N$. To compute the implicit weight associated with a particular rank r, we sum the weights associated with all pairs involving r, yielding:

$$\begin{aligned} \sum_{j=1}^{r} \frac{1}{R \cdot r} + \sum_{i=r+1}^{N} \frac{1}{R \cdot i} &= \frac{1}{R} \cdot (1 + \frac{1}{r+1} + \frac{1}{r+2} + \cdots + \frac{1}{N}) \\ &= \frac{1}{R} \cdot (1 + H_N - H_r) \end{aligned}$$

where H_k is the $k - th$ harmonic number.

The Rankhedge loss function in this chapter employs a modified version of this implicit weight function which takes into account the actual rate of decay of expected relevance at rank observed in TREC conference data. Since the constant is lost in the normalization step of the Hedge update process, a value function reflecting a document's contribution to the Average Precision measure should correspond to the tails of the harmonic series: $\text{val}(r) = \frac{1}{R} \cdot (1 + H_N - H_r)$. Integrating the scaling factor into the decay rate of the series via constant c_H, the value function in our initial implementation of Rankhedge is given by:

$$\text{val}(k) = \frac{1}{Z} * \sum_{r=k..N} \frac{1}{(1 + (c_H(r-1))} \tag{3.2}$$

with $Z = \sum_{r=1..N} \frac{1}{(1+(c_H(r-1))}$.

Given this value function, the loss is simply:

$$\ell = \frac{1}{2}\big(1 - 1^{\text{rel}(d)} \cdot value(r)\big) \tag{3.3}$$

Due to the close relationship of the value function with the tails of the harmonic series, we shall at times refer to the Rankhedge implementation based on this loss function as *Rankhedge(Log)*. The loss function has the advantage of being simple and "symmetric" (the magnitude of the loss or gain is independent of relevance). For the purposes of the Rankhedge algorithm, these losses or gains are mapped to the range $[0, 1]$ by an appropriate shift and scale.

We shall discuss in detail in the next chapter the empirical and theoretical evidence for a value function based on the inverse rank, but for now we shall continue with details of the Rankhedge(Log) implementation, followed by results of the algorithm.

Pooling

Given this loss function, we implement a simple pooling strategy designed to improve the learning rate of the Rankhedge algorithm. With s referencing the

```
Method: poolDocuments($\mathbf{D}, \mathbf{S}, \mathbf{w}, value()$)

Parameters:
        document space $\mathbf{D}$.
        predictors $\mathbf{S}$.
        weight vector $\mathbf{w} \in [0,1]^{|\mathbf{S}|}$.
        abs value of loss at rank $r$: $v = \mathbf{value}(r)$.

    1. return $\underset{d \in \mathbf{D}}{\mathbf{argmax}} \left( \sum_{s \in \mathbf{S}} w_s \cdot \mathrm{val}(rank_{(d,s)}) \right)$
```

Figure 3.4: RankHedge Subroutine: poolDocuments().

systems and d the document space, the next document to be pooled is given by:

$$d^t_{pool} = \underset{d}{\mathbf{argmax}} \Big(\sum_{s=1}^{N} w_s^{t-1} \cdot \mathrm{val}(r_{(d,s)}) \Big).$$

This corresponds to the unlabelled document with the maximum expectation of relevance as voted by a weighted linear combination of the systems. Thus, the strategy is appropriate for selecting documents to be output in a metasearch list.

We may also view the greedily selected sample as the choice which, if found to be non-relevant, will maximize the weighted average (mixture) loss. In this case, the sample yields the maximum expected change in system weights. In the TREC context, where relevant documents are relatively sparse— that is, where the likelihood of relevance of a particular sample selected from a ranked list is on average substantially less than 0.5— this also results in rapid diminution of the relative weights of unsuccessful predictors. Thus, in the sparse context, the generally orthogonal goals of maximizing immediate precision of the selected sample and of maximizing the convergence rate of the distribution over predictors are satisfactorily accommodated by the pooling technique.

```
Method: generateMetasearchList ($\mathbf{D}, \mathbf{S}, \mathbf{w}, \mathbf{P}, T, value()$)

Parameters:
        document space D.
        retrieval systems S.
        weight vector $\mathbf{w} \in [0,1]^{|\mathbf{S}|}$.
        pool list for output P.
        size of output list $T$.
        absolute value of loss at rank $r$: $v$ = value($r$).

  1. $\forall\ d \in \mathbf{D} : v_d = \left(\sum_{s \in \mathbf{S}} w_s \cdot \mathrm{val}(rank_{(d,s)})\right)$.

  3. $\mathbf{D}' = \mathbf{D} - \mathbf{P}$ (the set D less pooled documents P).

  2. $\mathbf{M} = \mathbf{D}'$ sorted by $v_d$.

  4. return $\mathbf{M}' = \mathbf{P}$ append M.
```

Figure 3.5: RankHedge Subroutine: generateMetasearchList().

Metasearch Lists and System Evaluation

In these experiments, metasearch lists are assembled by placing all pooled documents— in the order they were selected— at the top of the list. The function *getMetasearchList()* of Figure 3.2 then completes the list by ranking all remaining judged documents according to their current mixture loss and appending this list to the ordered list of pooled documents. Finally, the pool of judged documents is used as input to the standard TREC evaluation program to determine how well the pool performs as a discriminator of the quality of the underlying systems.

3.2 Results

In the TREC experiments, the Rankhedge algorithm is applied on a per query basis for an entire course of 1000 iterations. Thus, the trials do not take advantage of possible mechanisms of feedback between queries. At each iteration, precision results are averaged across systems. At each pooling level, the metasearch results are averaged across systems and the sample pools are used to define a

```
Method evaluateSystems ($\mathbf{S}, w, \mathbf{P}, rel()$)

Parameters:
        weight vector $\mathbf{w} \in [0,1]^{|\mathbf{S}|}$.
        retrieval systems S.
        pool list for output P.
        relevance judgement for $d$ :  $b = \mathrm{rel}(d)$.

    1. $\forall\ s \in \mathbf{S}$ calculate Mean Average Precision given pool P.

        Note: relevance(d) $=$ $\mathrm{rel}(d)\ \forall\ d \in \mathbf{P}$ : otherwise 0.

    2. return $\mathbf{S}' = \mathbf{S}$ sorted by MAP.
```

Figure 3.6: RankHedge Subroutine: evaluateSystems().

set of relevance judgements for the system evaluation task, with all samples not included in the sample pool considered to be non-relevant.

The Rankhedge algorithm demonstrated excellent performance across all TRECs tested (TRECs 3, 5, 6, 7, 8 and 9) in all three measures of performance—as an on-line metasearch engine, as a pooling strategy for finding large fractions of relevant documents, and as a mechanism for rapidly evaluating the relative performance of retrieval systems. Figures 3.7, 3.8, and 3.9 included at the conclusion of this Chapter compare results of Rankhedge, to those of the Cormack priority queue method and the Depth-n pooling method.

The constants necessary to compute the loss function in Equation 3.2 are

TREC	NSystems	MinLoss	c_H
3	40	0.55	0.0124
5	82	0.5	0.0351
6	79	0.5	0.0467
7	103	0.5	0.0438
8	129	0.5	0.0366
9	105	0.4	0.0444

Table 3.2: Constants used to establish β and val(r) for rankHedge trials (number of systems, minimum loss of best system, and expansion constant for harmonic series).

fairly similar across TREC conferences, and a coarse selection of c_H is sufficient to achieve good results. The actual constants employed throughout our experiments are given in Table 3.2.

In the following analysis, we compare the performance of standard TREC-style pools to Rankhedge pools of an equivalent total size. That is, if a TREC-style depth k pool contains m total judgments, it is compared to a Rankhedge pool with m total judgments. These pools are denoted Depth-k and Rankhedge-m, respectively.[1]

The topmost plost of Figures 3.7, 3.8, and 3.9 demonstrate the algorithm's success in finding relevant documents. The vertical axis corresponds to recall percentage and the lower dashed line indicates the performance of Depth-k pools for depths 1–10, 15, 20 and above as a function of the total number of documents judged. Rankhedge performance far surpasses the recall rates of the Depth-n pooling method when compared at equivalent numbers of total judged documents. The most enlightening measure of the success of the Rankhedge (and Cormack) methods over Depth-n pooling is to examine the number of judgments required to achieve equivalent recall percentages. For example, examining the TREC 8 curves along the horizontal axis, we see that the Depth-k method requires approximately 104 judgments to match the Rankhedge-40 return rate, and the Rankhedge-68 rate (36 percent) is unmatched until Depth-8 (199 judgments). After almost 500 judgments, Depth-20 has found only approximately 55% of relevant documents— a rate achieved by Rankhedge in less than 150 judgments.

Our initial algorithm also proved superior to the method of Cormack, et al. in the critical early stages of the retrieval process, though the priority queue method succeeds in finding relevant documents at a greater rate in the later,

[1]Note that the size of a depth k pool may vary on a query-by-query (and TREC-by-TREC) basis. In any given TREC, the total size of a depth k pool over all 50 queries is calculated, and for simplicity this pool is compared to a Rankhedge pool containing an equal number of total judgments, spread uniformly over all 50 queries.

sparser stages of the search. This indicates that our initial algorithm tends to overfocus on the lists of the more successful predictors— a problem which we correct in a subsequent algorithm. This tendency to overfocus is especially problematic in TREC 9, where conference results exhibited exceptionally low retrieval rates. In this instance the Rankhedge(Log) algorithm focused on the better predictors, but failed to adapt as their shallow pools of relevant documents were depleted.

The middle plots in our figures compare the quality of the system rankings produced by Rankhedge and Cormack pools against those of Depth-k pools at equivalent numbers of total judged documents using the Kendall's τ measure. Here, ground truth is the system ordering established by TREC. Again, the dashed line indicates the results of system evaluations performed using standard TREC routines, given Depth-k pools of size 1–10, 15, 20 and above. Examination of TREC 8 demonstrates typical performance. At 40 documents, the τ for Hedge is 0.87. This compares with 0.73 for the Depth-1 equivalent— a substantial improvement. Likewise, Rankhedge-68 achieves an accuracy of 0.91 vs. a Depth-2 equivalent accuracy of 0.73.

Next, comparing the pool depths required to achieve equivalent rates of ordering accuracy, we scan along the horizontal axis and see that to achieve an accuracy of 0.87 (Hedge-40), the equivalent Depth-3 pool requires 95 judgments. An accuracy of 0.91 (Hedge-69) is not achieved in the depth-pooling method until approximately 198 judgments (Depth-8). Performance of Rankhedge and the priority queue method in the system ranking task was roughly equivalent.

It is a significant advantage of the Rankhedge over the priority queue and the Depth-n pooling methods that the latter do not yield a mechanism for generation of metasearch lists, whereas the metasearch function is a natural product of the Rankhedge method for combination of evidence.

In the the lower plots, we compare the performance of the evolving metasearch list to the benchmark techniques CombMNZ and Condorcet as well as to the performance of the best underlying system in any given TREC. The Mean Average Precision (MAP) taken by selecting the best system for each query is also shown, providing a maximum bound on the possible MAP of a meta-classifier defined in terms of weighted linear combination of the underlying classifiers.

As shown in Table 3.1, the Rankhedge algorithm begins (in the absence of feedback) with a baseline MAP score— the mean of Average Precision scores taken across all queries— which is equivalent or slightly better in almost all instances to the performance of the CombMNZ and Condorcet metasearch methods. Condorcet and CombMNZ scores are included as dashed lines in the Figures. As relevance judgements are provided to the algorithm, the Rankhedge metasearch results quickly surpass those of the best underlying retrieval system (the upper dashed lines). In TRECs 3, 5, and 7, the performance of the best system is equalled in 10 or fewer judgments. TRECs 6 and 8 require slightly more judgments to achieve the performance of the best underlying system. This reflects the fact that in both cases the best systems are outliers, both in their total performance and in the documents they retrieve. Hence, Rankhedge must evaluate more documents to "discover" them.

Finally, behavior of the algorithm is substantially different in TREC 9 than in the other TRECs. This is due to the relative difficulty of the TREC 9 data set. A slight tendency of the original algorithm to overfocus on the best predictors leads it to concentrate on the few good predictors in the space only to find that their lists yield few relevant documents in the medium to lower depths. The original algorithm's tendency to overfocus is discussed at greater length, when we introduce our modified algorithm in Chapter Five.

A look at the scatter plots in Figures 3.10 and 3.11 demonstrates another aspect of the algorithm's performance in ranking systems— one which is somewhat obscured by the traditional Kendall's τ measure. Each pair of plots shows

Depth-1 and equivalent Rankhedge-m predicted ranks vs. actual TREC rankings. Note in these plots that the rankings proceed from best systems in the lower left corner to worst in the upper right. TREC 3 plots are somewhat anomalous due to the relatively low number of systems in the conference, but later TRECs demonstrate the difficulty of establishing proper rankings for the best systems given a Depth-n method restricted to smaller pool sizes.

While poor systems tend to be easily identified due to their lack of commonality with any other systems, the better systems tend to exhibit a similar divergence from the fold. Thus, while the rankings of poorer systems may be established using standard techniques with depth pools as small as Depth-1, the better systems (for many purposes, the systems of most interest) tend to be the more difficult to rank correctly. As the Kendall's τ measure of accuracy in object ordering treats objects at all rank levels equally, much of the qualitative superiority of algorithms which perform well in classifying the best systems is obscured by a common tendency of most techniques to perform well on the poorer systems. Examination of tightened patterns of the Rankhedge plots in the region of the best systems suggests that performance of the algorithm in evaluating system orderings is somewhat better than the performance demonstrated in Figures 3.7 to3.9 (c).

3.3 Conclusions

In conclusion, our inital attempt at a unified method for on-line metasearch, pooling, and system evaluation exhibits very good performance in all three areas for which it was designed. However, results tend to suffer in later stages of the retrieval process due to a tendency to overfocus on better predictors in the early stages. In the next two chapters, we shall examine the Rankhedge algorithm in more detail and demonstrate several interesting theoretical results concerning the algorithm's behavior on ranked lists— including an in-depth look at the loss bounds on the quality of the output list, a definition of the expected accuracy of the system evaluations and both empirical and theoretical examinations of the optimality of the loss function. This examination will lead, in Chapter Five, to the development of a new loss function for Rankhedge, resulting in an algorithm with results superior to all methods considered in this chapter.

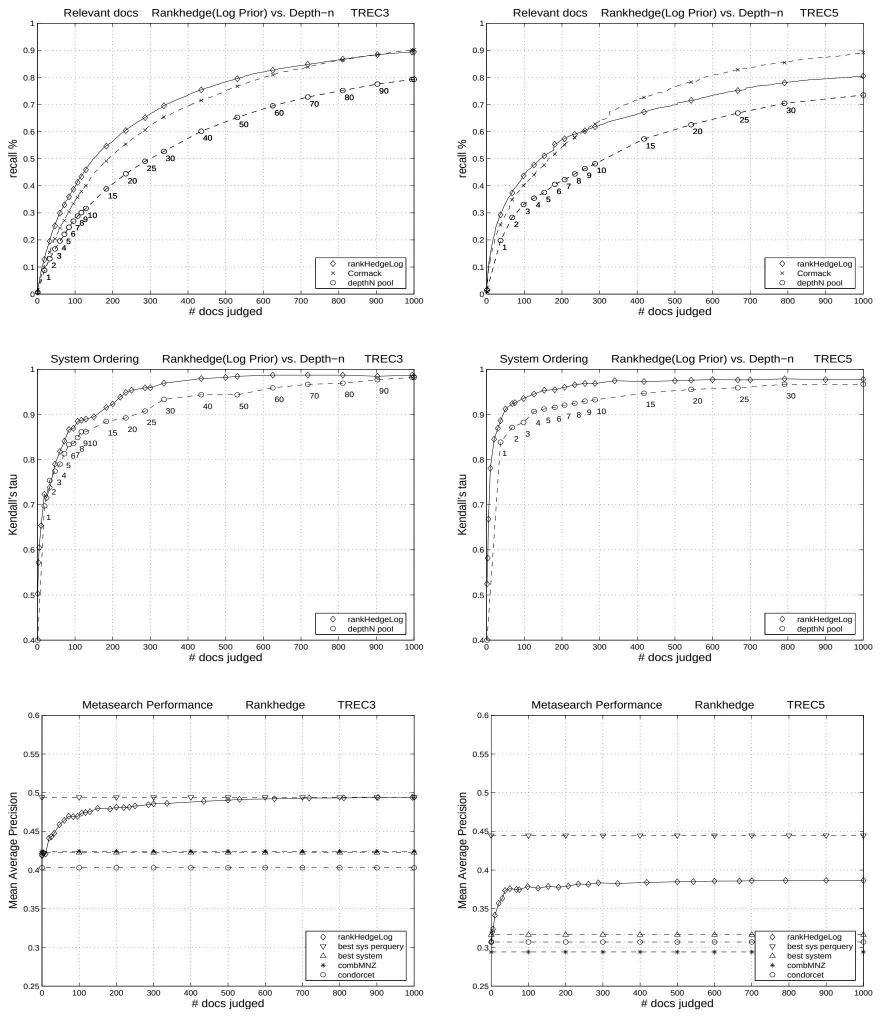


Figure 3.7: Trecs 3 and 5: Results of Rankhedge— with log prior under three performance measures: (A) percent of total relevant documents discovered (B) system ordering via k-τ (C) metasearch performance via Mean Average Precision.

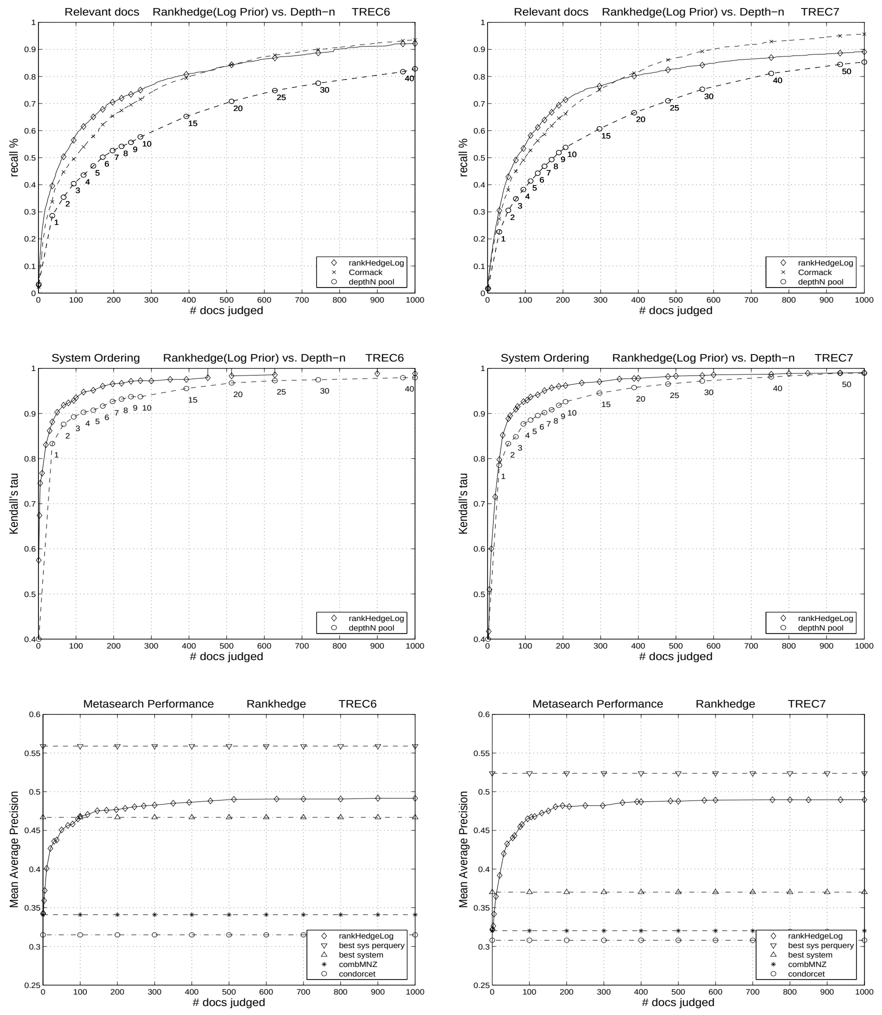


Figure 3.8: Trecs 6 and 7: Results of Rankhedge— with log prior under three performance measures: (A) percent of total relevant documents discovered (B) system ordering via k-τ (C) metasearch performance via Mean Average Precision.

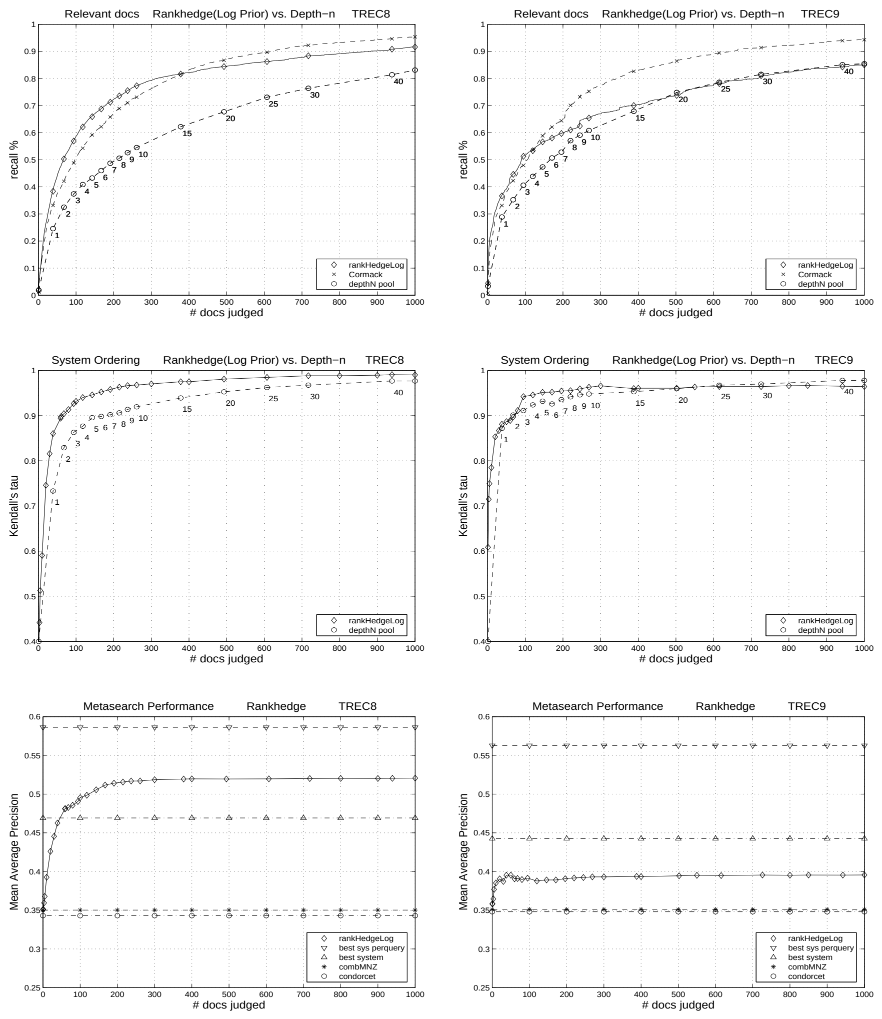


Figure 3.9: Trecs 8 and 9: Results of Rankhedge— with log prior under three performance measures: (A) percent of total relevant documents discovered (B) system ordering via k-τ (C) metasearch performance via Mean Average Precision.

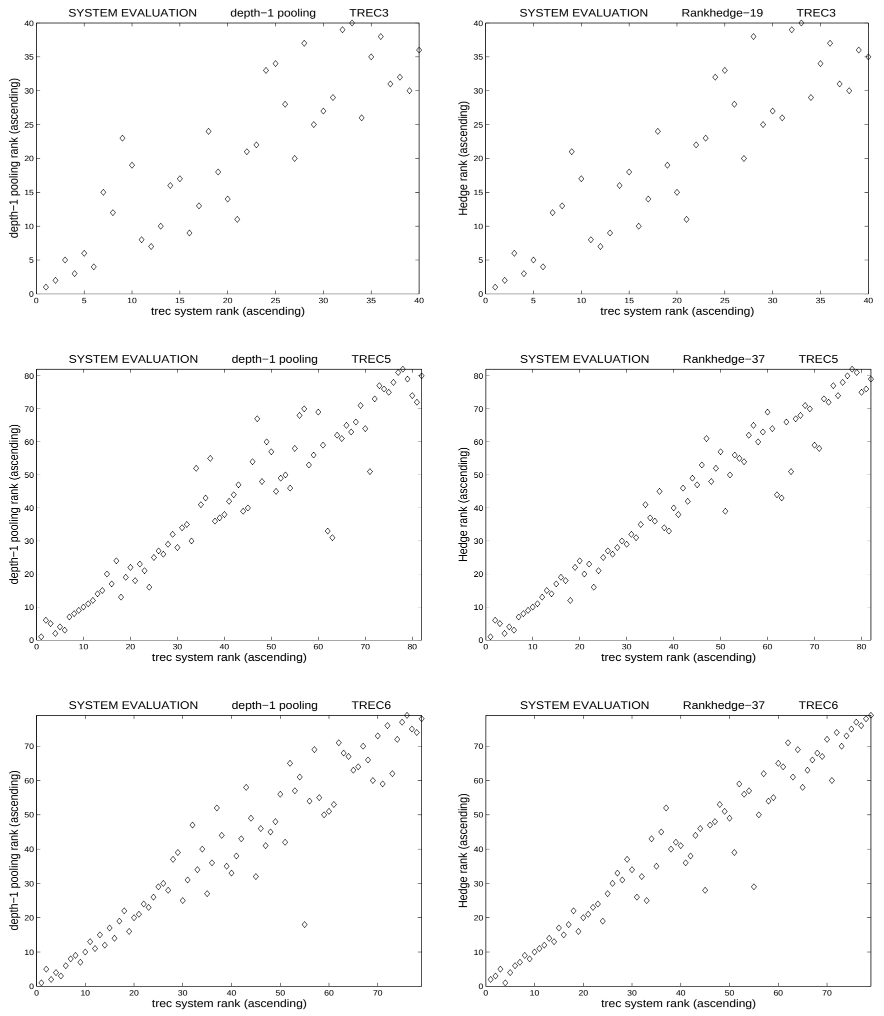
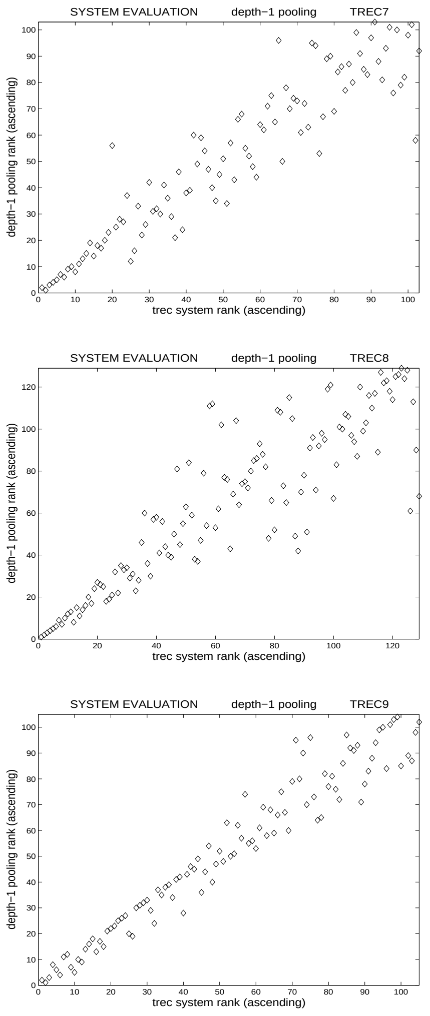


Figure 3.10: Trecs 3, 5, and 6: Depth-1 and equivalent Rankhedge-n rankings vs. actual ranks.



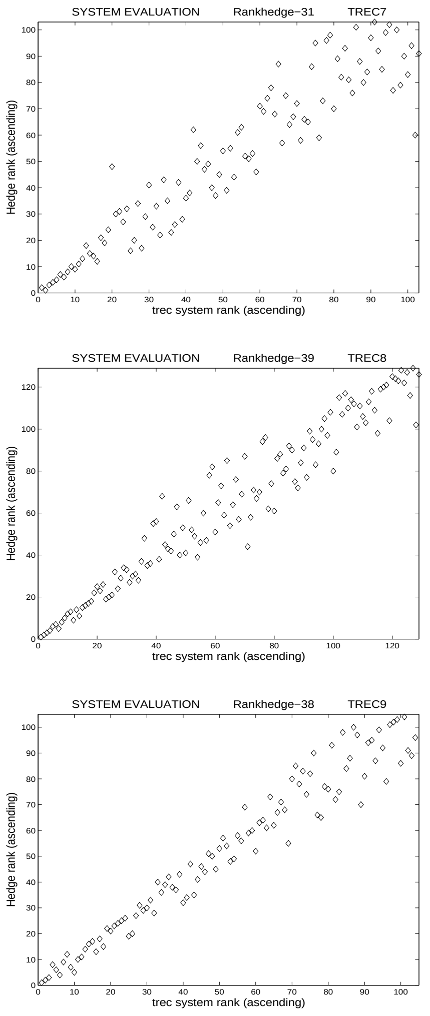


Figure 3.11: Trecs 7, 8, and 9: Depth-1 and equivalent Rankhedge-n rankings vs. actual ranks.

Chapter 4

Hedge for Ranked Lists

In this chapter, we consider in detail the extension of the Hedge algorithm to the problem of merging ranked lists. We begin by introducing two close relatives of our Rankhedge implementation. The first by Cohen, et al. [21] is a method for applying the Hedge algorithm in the context of pairwise preference functions to develop a rank ordering. A second method called *Rankboost* [36] based on the *Adaboost* algorithm [37] is a batch method for learning a combination of preference functions. Both algorithms share certain aspects with our own version of Rankhedge— the first, obviously, employing the Hedge method for on-line learning but with a substantially different formulation of loss function. The second method, Rankboost, generates linear combinations of ranking functions which are defined similarly to our own; however, the algorithm's focus is on batch learning via Adaboost. Our algorithm differs also in that it incorporates active selection of samples for labelling, while the earlier algorithms assume that samples are drawn uniformly from the unlabelled sample space.

In this and successive chapters, we introduce a Bayesian interpretation of the Hedge algorithm and exploit a prior knowledge of probability of relevance at rank to produce a "pseudo-Bayesian" metasearch algorithm generalized to the on-line setting. Our prior assumption of relevance at rank, based on the harmonic series, is shown to be supported by both empirical and theoretical evidence.

We examine the accuracy bounds for Hedge and demonstrate via counterexample that worst case bounds on accuracy of the output list and estimates of predictor accuracy are too large to be useful. However, bounds on the cumulative loss experienced by the user as well as average case estimates of predictor accuracy given a restricted class of predictors are available. Sensitivity to choice of β as well as to prior assumptions of relevance at rank are addressed in the final section.

4.1 Background

In establishing an historical context for our algorithm, we are primarily interested in two earlier methods for learning ranked orderings of lists. Both the Hedge based algorithm of [21] and *Rankboost* [36] possess certain aspects of our algorithm and in the following discussion we examine the properties of these methods as they relate to our Rankhedge implementation.

4.1.1 Learning to Order Things

Cohen, et al. [21] present a method for developing a rank ordering of instances given feedback in the form of pairwise *preference* judgements. The algorithm employs Hedge in the usual manner to learn a combined binary preference function providing a relative ranking between instances. Given the preference function, a strategy is developed to globally order all instances in a manner which maximizes the agreement with the learned preference function.

The Cohen method adopts a two stage approach. In the first stage, it learns a *preference function*, a two-argument function $\text{PREF}(u, v)$ returning a measure

```
Algorithm: HedgeOrder($\beta$)

Parameters:

        number of ranking experts $N$.
        initial weight vector $\mathbf{w}^1 \in [0,1]^N$ with $\sum_{i=1}^N w_i^1 = 1$
        number of rounds $T$.
        $\beta \in [0,1]$.

Do for $t = 1, 2, \ldots, T$.

    1. Receive a set of elements $X^t$ and ordering functions $f_1^t, \ldots, f_N^t$. Let $R_i^t$
       be the preference function induced by $f_i^t$.
    2. Compute a total order $\hat{p}^t$ approximating

            $PREF^t(u,v) = \sum_{i=1}^N w_i^t R_i^t(u,v)$

    3. Order $X^t$ using $\hat{p}^t$.
    4. Receive feedback $F^t$ from the user.
    5. Evaluate losses $Loss(R_i^t, F^t)$.
    6. Set the new weight vector:

            $\mathbf{w}_i^{t+1} = \frac{\mathbf{w}_i^t \cdot \beta^{Loss(R_i^t, F^t)}}{Z^t = \sum_{i=1}^N w_i^{t+1}}$
```

Figure 4.1: Algorithm for Ranking using Preference Functions.

of the probability that u is ranked before v. In stage two, the algorithm uses the learned preference function to order a set of new instances X by evaluating the function PREF(u, v) on all pairs of instances $u, v \in X$ and then choosing an ordering of X that agrees as much as possible with the pairwise preference judgements.

Formally, let X be a set of instances. Assume X is finite. A *preference function* PREF is a binary function PREF : $X \times X \rightarrow [0,1]$, with 1 indicating a strong recommendation that $u > v$. Value 0.5 constitutes an abstention. Ranking information from a set of N experts is provided by preference functions $R_1, \ldots, R_N$. The algorithm assumes that the R_i's are well formed— i.e. they reflect an underlying ordered set S. An ordering function $f : X \rightarrow S$ induces the preference function R_f defined as:

$$R_f(u,v) = \begin{cases} 1 & if\ f(u) > f(v) \\ 0 & if\ f(u) < f(v) \\ \frac{1}{2} & otherwise. \end{cases} \tag{4.1}$$

At each round t, the user is queried for feedback on preference of instance u_t or v_t, so that at the t-th round, we have the set of ordered pairs $F^t = \{(u_1, v_1)^{\pm}, \ldots, (u_t, v_t)^{\pm}\}$. To implement the Hedge algorithm, we need to define a loss function in terms of the preference function R and feedback F, so that

$$Loss(R, F) = \frac{\sum_{(u,v)\in F}(1 - R(u,v))}{|F|} = 1 - \frac{1}{|F|}\sum_{(u,v)\in F} R(u,v) \tag{4.2}$$

The loss has a natural probabilistic interpretation. With R viewed as a randomized prediction algorithm which predicts that u is ranked higher (in ascending order) than v with probability $R(u,v)$, the $Loss(R,F)$ reflects the probability that R disagrees with the feedback on pair (u,v) chosen uniformly at random from F.

The preference function at round t may be derived from a linear combination of preference functions $\mathrm{PREF}^t(u,v) = \sum_{i=1}^{N} w_i^t R_i^t(u,v)$. Losses at round t are defined relative to this preference function, and we may establish Hedge bounds of the usual form:

$$\sum_{t=1}^{T} Loss(\mathrm{PREF}^t, F^t) \leq a_\beta \min_i \sum_{t=1}^{T} Loss(R_i^t, F^t) + c_\beta \ln N \tag{4.3}$$

with $a_\beta = \ln(1/\beta)/(1-\beta)$ and $c_\beta = 1/(1-\beta)$. The cumulative loss of the combined preference function $\sum_t Loss(\mathrm{PREF}^t, F^t)$ will not be much worse than the cumulative loss of the best ranking expert ($\min_i \sum_{t=1}^{T} Loss(R_i^t, F^t)$).

A total ordering must be developed from the weighted preference functions.

```
Algorithm: RankBoost($\beta$)

Given: initial distribution $D$ over $X \times X$.
Initialize: $D_1 = D$.
For $t = 1, 2, \ldots, T$:

    1. Train weak learner using distribution $D_t$.
    2. Get weak hypothesis $h_t : X \to \mathbf{R}$.
    3. Choose $\alpha_t \in \mathbf{R}$.
    4. Update: $D_{t+1}(x_0, x_1) = \frac{D_t(x_0,x_1)\exp(\alpha_t(h_t(x_0)-h_t(x_1)))}{Z_t}$
       where $Z_t$ is a normalization factor chosen so that $D_{t+1}$ is a distribution.

Output the final hypothesis: $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.
```

Figure 4.2: The RankBoost Algorithm

A simple measure of agreement between a total ordering p (with $p(u) > p(v)$ if and only if u is above v in the ordering) and a preference function PREF(u, v) may be defined by summing PREF(u, v) over all pairs u, v such that $p(u) > p(v)$:

$$\text{AGREE}(p, \text{PREF}) = \sum_{u,v:p(u)>p(v)} \text{PREF}(u, v)$$

As Cohen, et al. [21] demonstrate, the task of finding an ordering which is optimal in terms of the weighted agreement measure is NP-complete; however, a simple greedy algorithm described in the paper produces an ordering which is within a factor of two of the optimal.

4.1.2 RankBoost

Freund, et al. [36] present a method for combining preferences based on the well-known Adaboost algorithm. The definition of the ranking problem presented in the paper is similar to that of our Hedge algorithm with ranking features assigning an absolute value to the instances. However, to avoid possible inconsistencies in scaling of the features, the algorithm converts the features to relative ordering relationships prior to boosting.

One might view the Rankboost algorithm as a batch method for learning relationships, as opposed to the on-line methods based on Hedge. In this discussion, we will focus on the formulation of the preference and feedback functions employed by the algorithm, since the mechanics of the boosting algorithm itself (delineated in Figure 4.2) are little changed by their adaptation to the specific case of ranked lists.

As formulated in [36], the Rankboost algorithm produces a function $H : X \to \mathbb{R}$ whose induced ordering of X will approximate the relative orderings encoded by the feedback function Φ. We may view this feedback function as equivalent to the ordering relationship function which is implicit in the relevance feedback supplied to our Hedge algorithm. That is:

$$\Phi(x_0, x_1) = \begin{cases} 1 & if\ rel(x_1) = 1\ \wedge\ rel(x_0) = 0 \\ -1 & if\ rel(x_1) = 0\ \wedge\ rel(x_0) = 1 \\ 0 & otherwise \end{cases} \tag{4.4}$$

The formulation of Rankboost presented in the paper operates on a distribution restricted to the positive relations, setting all negative entries of the feedback function to zero, so that $D(x_0, x_1) = c \cdot \max(0, \Phi(x_0, x_1))$. By instituting this convention, the Rankboost implementation restricts the loss function to the range $[0, 1]$, and thus guarantees monotonicity in the evolution of the individual feature weights.

The algorithm accepts as inputs a set of ranking features $f_1, \cdots, f_n$, which are functions of the form $f_i : X \to \widehat{\mathbb{R}}$. The set $\widehat{\mathbb{R}}$ consists of the real numbers plus an additional element $\perp$ that indicates no ranking has been given. From these features, the weak hypotheses may be derived. Each weak hypothesis assigns an estimate of relevance of item $x \in X$, given the ranking function f_i. Undefined instances ($\perp$) may be assigned an arbitrary constant value q_{def} in the range $[0, 1]$ as determined on a per-feature basis by the algorithm.

Many definitions for weak hypotheses are possible. The formulation based directly on the estimated values of $f_i(x)$ corresponds most closely to our own Bayesian formulation:

$$h_i(x) = \begin{cases} f_i(x) & if\ f_i(x) \in \mathbb{R} \\ q_{\text{def}} & if\ f_i(x) = \perp \end{cases} \tag{4.5}$$

The paper, however, focuses on an alternate formulation which is independent of the actual value assignments of the functions. A weak hypothesis is generated by selecting the feature f_i, partition point θ and $q_{def} \in \{0, 1\}$ which results in a hypothesis of maximum accuracy given the current distribution over the instances. In terms of the triple $[f_i, \theta, q_{\text{def}}]$ produced by the search process, the weak hypothesis is defined to be:

$$h_i(x) = \begin{cases} 1 & if\ f_i(x) > \theta \\ 0 & if\ f_i(x) \leq \theta \\ q_{\text{def}} & if\ f_i(x) = \perp \end{cases} \tag{4.6}$$

As noted earlier, learning proceeds in the relationship space, with the distribution over instances $D(x_0, x_1)$ defined on the space of relationships s.t. $x_0 > x_1$. The final hypothesis has the form $H(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$.

4.2 Prior Probability of Relevance and Inverse Rank

Our implementation of the Rankhedge algorithm combines some aspects of both Rankboost and the Hedge method of Cohen, et al. The earlier Hedge method bases its loss function on a binary preference relationship between documents and accepts feedback in the form of assertions of pairs (u, v) ordered according to preference. Aggregate results of a preference function defined in this manner are closely related to the familiar Kendall's τ measure of distance between ranked lists defined as the minimum number of transpositions needed to bring the two lists to agreement. The problem with the accuracy measure of [21], as with Kendall's τ, is that the use of a binary preference function in the context of ranked lists obscures the natural expectation of monotonically decreasing relevance at rank and thus has a tendency to attribute too much significance to the relationship of instances with lower rank (order ascending). Generalization of the algorithm of [21] to allow for a continuum of preference scores based on expected relevance at rank would lead to results similar to that of our Rankhedge algorithm. A major contribution of our algorithm is the definition of an empirically and theoretically grounded prior for relevance at rank.

Since Rankboost defines expected losses directly on the individual instances rather than indirectly via a preference function, the structure of Rankboost's predictions is closest to that of our algorithm. However, just as in [21], the Rankboost algorithm employs a thresholded relevance assessment, or, at best, an integer weighting based on the rank indices, both of which tend to attribute excessive significance to instances of lesser rank (order ascending).

In this section we shall examine our analytic prior for relevance at rank and present evidence for the choice of an inverse rank function. We establish not only an empirical basis for our prior but also a theoretical foundation for the phenomenological law governing our prior relevance at rank— demonstrating that the function is closely related to a *universal* or scale invariant function which accurately captures the expansion with rank of the underlying support of the space of samples.

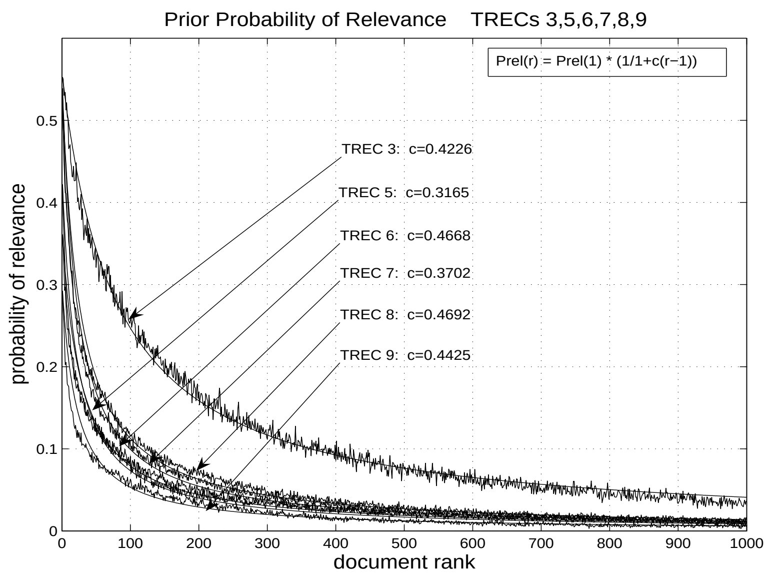


Figure 4.3: $Pr(relevance|rank)$ vs. Actual: TRECs, 3,5,6,7,8 and 9.

4.2.1 Empirical Evidence

Figure 4.3 plots probability of relevance at document rank for each of the TRECs 3,5,6,7,8 and 9. $Pr(d \in rel|rank(d))$ for each TREC is obtained by averaging the probability of document relevance at rank across systems in all fifty queries of a retrieval track. From these curves, we determine a characteristic function defining prior probability of relevance at rank to be:

$$Pr(d \in rel|trec\ t, rank\ r, system\ s) = p_s * \frac{1}{1 + c_t(r-1)}, r = 1, \ldots, n. \quad (4.7)$$

As demonstrated in Figure 4.3, curves corresponding to this prior probability of relevance show a strong qualitative resemblance to the actual probabilities. Figure 4.4 plots these same relevance at rank on a log-log scale. The linearity displayed by these plots throughout a substantial portion of their development indicates that the rule governing the evolution of relevance at rank is a power

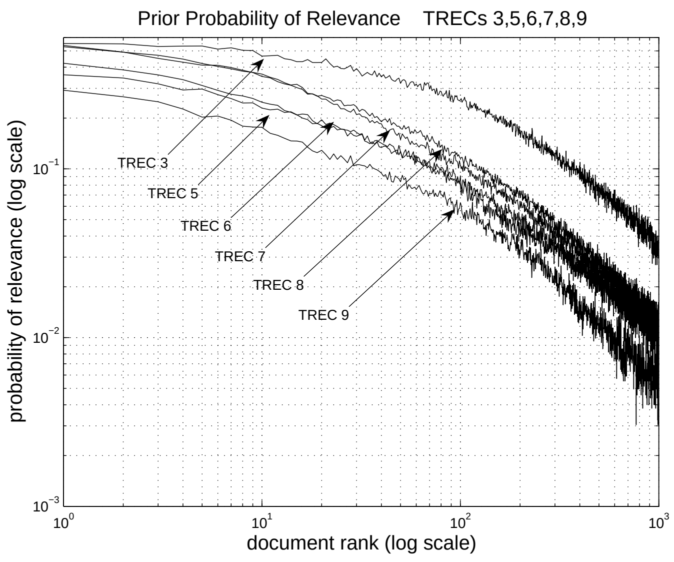


Figure 4.4: $Pr(relevance|rank)$ (log-log scale): TRECs, 3,5,6,7,8 and 9.

law. Constants c_t used in our experiments are given by:

$$\begin{aligned} TREC3 &= 0.0124 \\ TREC5 &= 0.0351 \\ TREC6 &= 0.0467 \\ TREC7 &= 0.0438 \\ TREC8 &= 0.0366 \\ TREC9 &= 0.0444 \end{aligned}$$

Figures 4.5 and 4.6 demonstrate the extent to which the bundle of systems follows this characteristic curve. Rather than plotting the relevance at rank directly, the figures plot the precision of the system lists at each rank to min-

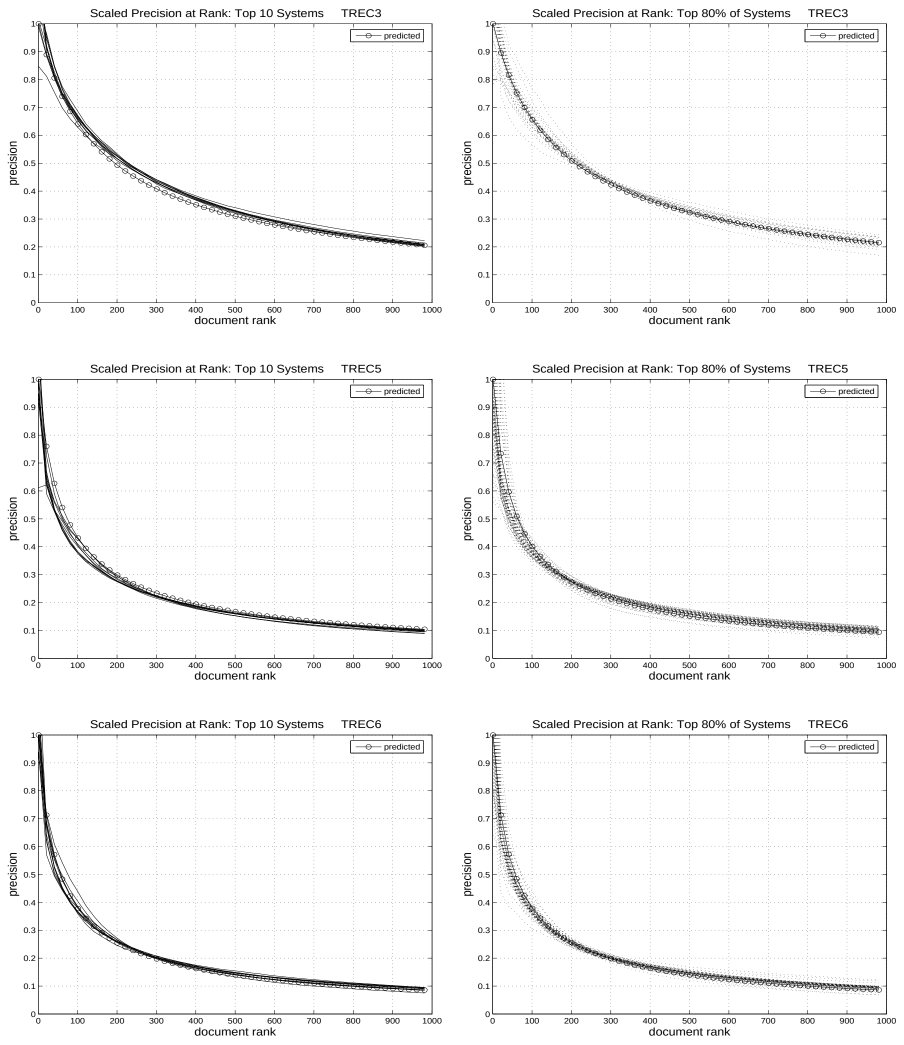


Figure 4.5: Trecs 3, 5 and 6: Precision at rank for Top 10 and Top 80% system bundles sorted by Mean Average Precision. Precision at rank is shown, rather than prior relevance at rank, for noise reduction. Maximum precision of all systems is scaled to one for clarity.

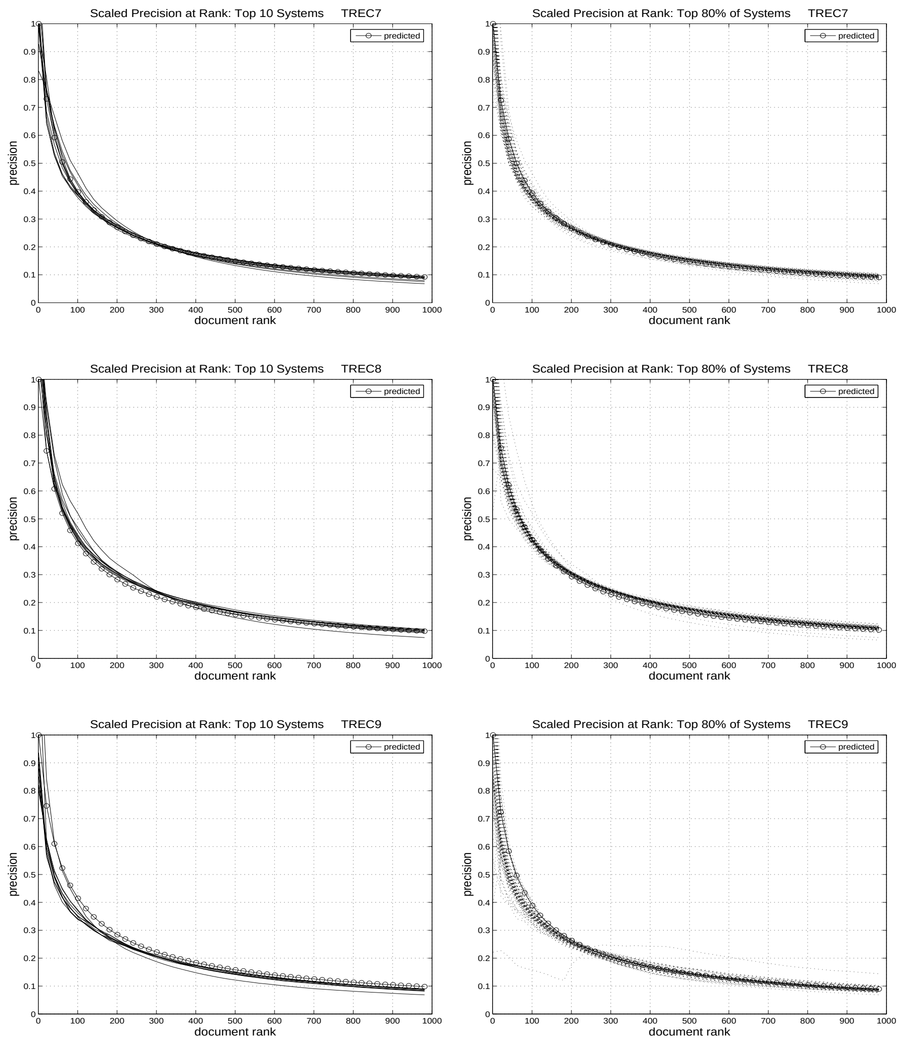


Figure 4.6: Trecs 7, 8, and 9 : Precision at rank for Top 10 and Top 80% system bundles sorted by Mean Average Precision. Precision at rank is shown, rather than prior relevance at rank, for noise reduction. Maximum precision of all systems is scaled to one for clarity.

imize the noise in the individual system statistics. Still, the degree to which the smoothed behavior of the systems adheres to the prior is remarkable. Figures 4.5a and 4.6a show a bundle consisting of the top ten systems versus the prediction, and Figures 4.5b and 4.6b plot the top eighty percent of systems.

As we shall demonstrate in a later section, the relevance at rank curves derive from a generalization of a scale invariant distribution over the finite sample set. Equation 4.7 defines expected relevance at rank for a predictor and requires two parameters to define relevance at rank for a particular system. The expected accuracy of a system p_s is defined by measuring the area under the averaged (across queries) relevance curves for the system. All systems in a bundle are scaled by $1/p_s$ to allow qualitative comparison of the curves. The decay rate c_t is the same for all systems and set on a TREC-wide basis by averaging the solving for the decay rate at each rank given the measured average relevance of the top 80% bundle.

An intuitive explanation for the prior is that the value of each incremental judgement in the ranked list is inversely proportional to the size of the pool already judged, thus reflecting the relative "uniqueness" of each rank level. That is, with S_r corresponding to the pool of documents returned by a system to rank r, $Pr(d \in S_r) \propto \frac{1}{c|S_r|}$. We shall examine this assertion in detail in the next section.

4.2.2 Theoretical Justification for the Prior

One expects that, given the strength of the empirical evidence for a scalable prior which is proportional to inverse rank, the observed probability of relevance at rank curves reflects an essential attribute of the supporting set of instances. We shall demonstrate that the *inverse rank property* does, in fact, reflect the relationship of each instance to an expanding instance space implicit in the rank ordering. We further demonstrate that the distribution based on the inverse rank function is a generalization of a scale invariant or *universal* distribution.

Scale invariance refers to the fact that all curves in the class devolve via inverse scalar transformation to a single curve $f(r)$, so that $f(c,r) = f(cr) = cf(r)$. In our context, c would be proportional to the inverse of the normalization factor associated with the expected total relevant documents. Universality corresponds to the fact that the curves defined by $f(cr) = 1/cr$ may be demonstrated to be the *only* curves of univariate paramaterization possessing this property of scale-invariance.

We shall first justify our statement that the inverse rank property results in a scale-invariant distribution by considering the limiting case in which the discrete elements of the support— the ranks— are assumed to be independent. The relevance at rank curves employed in Rankhedge correspond to a straightforward generalization of the scale-invariant inverse rank function, defined as before ($f(r) = 1/(1 + c(r-1))$, with the loosening of the independence restriction encoded in the constant of expansion c.

The Inverse Rank Property

The essence of the inverse rank property may be expressed in minimal terms as follows. Let S_r be the set of items in the range $1...R$ and $s_r \in S_R$ the rth sample in the range. When drawing r items at random from a bin of size $|S_r|$, the probability of drawing an arbitrary item s_r in the rth trial is equivalent to one minus the probability that the item was drawn in any of the previous rounds, and we may write (with $S_r = \{s_1...s_r\}$):

$$Pr(s = s_r | s \in S_r) = Pr(s \in S_r \cap s \notin S_{r-1}) = 1 - \frac{|S_{r-1}|}{|S_r|} = \frac{1}{|S_r|} \qquad (4.8)$$

Since the inverse rank property encodes the inverse of the dimensionality of the sample set, the relative weight of an element in the support of the distribution should reflect the incremental change in dimensionality of the sample set due to addition of the element.

Relevance at Rank Curves

For the inverse rank weighting property to be considered an *a* priori property of the rank ordered lists, it must submit to application to ranked lists with varying accuracies. Thus, we would like the rank weighting curves to be (approximately) scale-invariant. For the case in which discrete support elements are assumed to be independent, we demonstrate that the function of interest is strictly scale invariant.

For a distribution $P(r)$ defined on $r = 1, \ldots, R$, we may approximate an arbitrary generalization of the distribution in terms of a continuous transformation $r \mapsto f(r)$. That is, we may define $p(r) = \int_{r_i}^{r_i+\Delta} P(r)dr$ on $r \in [1, R]$ with $\Delta = 1$ corresponding to the usual discrete interval.

Now, we wish to define a constant scaling of the aggregate weight of the sample space in terms of a functional manipulation of the r axis. That is, we wish to define a transformation $f(r)$ such that:

$$\int_{r}^{r+\Delta} P(f(r))dr = k \int_{f(r)}^{f(r+\Delta)} P(r)dr$$

This transformation corresponds to a scalar transformation of the domain and has the familiar form: $f(r) = cr$ with $k = 1/c$. The resultant relevance at rank curve has a description prior to normalization of $P(cr) = c^{-1}P(r)$. Normalization results in the original p.d.f. $P(cr)/\sum_{r=1}^{R} P(cr) = P(r)$

Further, by taking the derivative of this scaling property in terms of c and solving for $c = 1$:

$$\begin{aligned}
\frac{\partial P(cr)}{\partial c} = \frac{\partial c^{-1} P(r)}{\partial c} \qquad & \big|\ c = 1 \quad \Rightarrow \\
rP'(cr) = -c^{-2} P(r) \qquad & \big|\ c = 1 \quad \Rightarrow \\
rP'(r) = -P(r) \qquad & \qquad\qquad \Rightarrow \\
P(r) = 1/r, \qquad &
\end{aligned}$$

we establish that the inverse rank weighting is, in fact, unique in possessing the scale invariance property.

Scale invariance produces a set of curves which have minimal log separation for arbitrary variations in the scaling parameter. In terms of the Kullback Liebler distance between unnormalized distributions, we may write:

$$\sum_{r=1}^{R} P(r) \ln \frac{P(r)}{P(cr)} = \sum_{r=1}^{R} P(r) \ln \frac{cP(r)}{P(r)} = \ln c$$

Unfortunately, by definition, normalization of the scale-invariant inverse rank function always results in the original function $f(r) = 1/r$. Therefore, to allow for variation in the the decay rate of elements of the p.d.f. describing relevance at rank, we relax the independence requirement and define a generalized prior relevance at rank curve which is equivalent to the scale-invariant function for value of $c = 1$. That is:

$$Pr(rel|r) = 1/(1 + c(r - 1))$$

Note that curves of this form still possess the desired property that for span R of sufficient size, the function is approximately scale-invariant with $\sum_{r=1}^{R} 1/(1+c(r-1)) \approx \sum_{r=1}^{R} 1/(cr)$.

Scale invariance has several nice properties, not the least of which is the preservation of a consistent Bayesian interpretation of the rank weightings on individual instances under scalar modification of the amplitude of the relevance at rank curve. In Chapter Five, we shall demonstrate that an accurate estimate of prior probability of relevance may lead to an optimal loss function, resulting in a Rankhedge algorithm which approximates an optimal Bayesian on-line algorithm. For the moment, however, we shall address the question of bounds on the Rankhedge algorithm in a context independent of the notion of prior relevance at rank.

4.3 Bounds on Accuracy

In this section, we examine the possibility of establishing useful bounds on the accuracy of the algorithm— both for the quality of the output list and for the estimation of accuracy of the individual predictors. We shall present counterexamples demonstrating that worst case bounds for the algorithm are poor for either of these measures. However, in both cases, we are able to demonstrate useful bounds via a reasonable modification of the definition of the quantity to be bounded.

In the first portion of our discussion, we examine output list accuracy and demonstrate that the monotonically decreasing nature of relevance at rank limits the ability to arrive at a useful bound for list accuracy in the on-line setting. We also show, however, that the accuracy of the sample stream may be defined practically in the usual Hedge manner, yielding useful bounds which are consistent with the actual experience of the user.

In the second section, we examine the possibility of bounding the accuracy of estimates of list quality. Again, we demonstrate via counterexample that no useful bounds exist in the general case in which ranked lists may have arbitrary accuracy. We circumvent this failure by placing a restriction on the maximum quality of the predictors and demonstrate that, given a restriction on the expected accuracy of the best predictor ($L_{\text{best}} = \epsilon > 1/2$), the algorithm does, in fact, sample the full set of predictors in a manner which allows useful bounds on quality estimates.

4.3.1 Output List Accuracy

In the traditional Hedge setting, the loss associated with the algorithm's prediction on a sample is independent of the iteration at which the sample is drawn. However, in the rank-sensitive application, the magnitude of a sample's loss (or gain) as defined by its depth in the output list decreases monotonically with rank. This makes it difficult to establish a worthwhile bound on the loss of the output sequence, as revealed by the following counterexample.

Counterexample 1 (Worst Case Output List Accuracy) *The worst case loss of the ranked list generated by the rank-Hedge algorithm is $O(N)$, where N is the number of underlying systems.*

Proof: *Define an initial set of predictors $\mathbf{p} = \{\mathbf{p}_0, \ldots, \mathbf{p}_N\}$ such that the first item in each list p_i^1 is returned by only one list. That is: $\forall i \neq j : \mathbf{p}_i^1 \notin \{\mathbf{p}_j^1, \cdots, \mathbf{p}_j^{\max}\}$. Assume $\mathbf{p}_0$ is the best predictor with loss L_{best} while the first items of lists $\mathbf{p}_{1..N}$ are non-relevant, i.e.* $\text{rel}(\mathbf{p}_{1..N}^1) = 0$.

Since all lists are orthogonal (no document returned by a system is returned by any other), labeling of sample p_j^1 results in a relative loss to list j only, and the expected round at which the first sample is drawn from list zero is $t = N/2$. Thus, we may trivially establish a bound on the loss of the output list L_{out} to be (with ℓ^t corresponding to loss at rank t): $L_{\text{out}} \geq L_{\text{best}} + \sum_{t=1}^{N/2} \ell_t = L_{\text{best}} + O(N)$.
$\square$

The worst case bounds on the loss of the output list are exponentially worse than the $O(\log(n))$ bounds we commonly expect from Hedge. This is due to the fact that this assessment of loss is based on an arbitrary measure imposed by output list rank, rather than on the expected losses associated with the actual weighted average predictions (i.e. the usual Hedge loss measure).

As an alternative to defining loss strictly in terms of the output list accuracy, we would like to assess loss to the algorithm in a manner which reflects the experience of the user. Intuitively, a sample drawn early in the learning process will likely have less certainty of relevance than a later sample. A definition of loss in terms of the expected loss of the weighted basis of predictors on the samples captures this quality of evolving certainty in the predictions. At early stages, the uncertainty of the prediction is manifested in the broader unfocused nature of the distribution over predictors and the resultant diminution of expected relevance of samples. At later stages in the learning process, the weights of the algorithm reflect the performance of the predictors on a larger sample as the expected loss of the algorithm converges to near that of the best predictor.

Since the expected loss measure is actually the basis for the original Hedge bounds, these bounds still apply. And, since they reflect the actual weighted average assessment of the sample quality at each iteration, they correspond more closely to the losses experienced by the user. In the case of our counterexample, we see that in the usual Hedge loss measure the loss associated with the first sample will be $O(1/N)$ rather than $O(1)$, reflecting the lack of agreement among predictors as well as the lack of certainty of the user in selecting this sample. Thus, though the worst case bound on the quality of the output list is poor, we may still bound the perceived loss in terms of the user's expectations of sample relevance via a Hedge bound of the usual form:

$$L_{Hedge(\beta)} \leq \frac{\min_i\{L_i\} \cdot \ln(1/\beta) + \ln N}{1-\beta}$$

4.3.2 Estimation of Predictor Accuracy

We would also like to be able to make a statement concerning the Rankhedge algorithm's ability to discern the quality of the underlying predictors. Experimental evidence demonstrates that accurate system evaluation is a useful side-effect of the Hedge process. This section begins with a counterexample demonstrating that the algorithm fails in the system evaluation task in the most general case of unrestricted accuracy of the predictor lists. However, for restricted cases consistent with our current application (i.e. lossy predictors), we demonstrate that the algorithm samples the lists associated with the individual predictors at a rate proportional to their expected accuracy.

In this restricted context, we may define a bound on the expected oversampling of a portion of the predictor space due to non-uniformity of the initial weight distribution resulting from correlations among predictors. This oversampling of correlated predictors serves to exaggerate the predictors' contribution to the sample pool and thus to enhance the quality estimates of the correlated systems. However, the effect diminishes progressively with growth of the sample pool.

To further alleviate these bias effects, an alternate technique for estimating system quality is presented, which takes advantage of Chernoff/Hoeffding bounds to define expected convergence of the observed system quality to the actual value. This technique provides a more rigorous estimate of the quality of system predictions as well as an explicit method for determining sampling quantities required for accurate estimation.

Counterexample 2 (Worst Case System Accuracy) *Given two sets of finite retrieval lists of equal length T, $\mathbf{p}_u = \{\mathbf{p}_0\}$ and $\mathbf{p}_v = \{\mathbf{p}_1, \ldots, \mathbf{p}_N\}$, let the accuracy of all lists be $\geq 1/2$. All items in set $\mathbf{p}_v$ are equivalent, so that $\forall r \in \{1 \cdots T\}$ and $i, j \in \{1 \cdots N\} : p_i(r) = p_j(r)$. Define $\mathbf{p}_u$ to be orthogonal to*

$\mathbf{p}_v$*, so that no items returned by* $\mathbf{p}_u$ *are returned by* $\mathbf{p}_v$ *and vice-versa. Using the greedy active sampling strategy maximizing instantaneous relevance, an arbitrary number of items will be sampled from lists* $\mathbf{p}_v$ *before any item returned by* $\mathbf{p}_u$ *appears in the pool.*

Proof: *The loss associated with a document returned by a list* i *at rank* r *is given by* $\pm\ell_i^r$*, with the loss associated with non-returned documents* $\ell_{r\mapsto\infty}$ *corresponding to a relative zero. Sampling from the list of a predictor from set* v *with an expected net gain* $\sum_r \mathrm{sgn}(rel(r))\ell_u^r \geq 0$*, results in an expected weight for the predictor of* $w_v^r \geq w_0 \cdot \beta^0 = w_0$*. Likewise, for predictor* u*, we expect the weight to be unchanged (i.e.* $w_u = w_0\beta^0 = w_0$*), due to the orthogonality of the lists. For an arbitrary* N*, the lists of the correlated predictors will be sampled until the aggregate weight of* $\mathbf{p}_v$ *becomes less than that of* $\mathbf{p}_u$*, that is, until* $\sum_{t=1}^{r} \mathrm{sgn}(\mathrm{rel}(r))\ell_u^t \leq \frac{\ln N}{\ln \beta}$*. Thus, lists* $\mathbf{p}_v$ *may be sampled to an arbitrary depth, determined by* N*, prior to observing a sample from* $\mathbf{p}_u$*.* □

4.3.3 Restriction to Lossy Predictors

The counterexample demonstrates that the natural focus of the Hedge algorithm is the accurate labeling of samples rather than determination of predictor accuracy. When coupled with a greedy pooling method which seeks to maximize the instantaneous accuracy of the selected sample, the effect is to focus on regions of the sample space which reinforce the current predictor distribution. When the goal is minimization of prediction error, this is not a problem, since intuitively we may say that as long as our current distribution is successful on the sample set, there is no incentive to explore other regions of the predictor space. However, if the goal is to establish the quality of predictors across the entire predictor set, this tendency to focus on successful predictors guarantees that other regions of the predictor space remain unexplored.

To avoid the difficulties described in Counterexample 2, we shall from this point restrict our discussion to the case in which the best predictor expects a net loss on the sample set. This guarantees that all predictors will be sampled by the greedy algorithm and allows us to develop a simple description for the expected sampling rates of predictors. Fortunately (in this context), the restriction to lossy predictors is consistent with our assumptions regarding the prior probability of relevance at rank encountered in practice.

4.3.4 Expected Contribution to Sampling Pool

We shall demonstrate that, in the case in which all predictors are expected to suffer a net loss on an arbitrary sample set drawn from their ranked lists, the contribution of a predictor's selections to the sample pool is inversely proportional to the loss rate of the predictor.

Neglecting, for the moment, oversampling due to correlations among predictors, we consider a set of independent predictors with initially uniform weights. In the restricted case in which the best predictor suffers a net loss on an arbitrary sample set, the active sampling algorithm will tend to focus on a particular predictor until the weighted value of the highest ranked sample has dropped below that of a sample offered by another predictor— at which point the sample selection process shifts its focus to a new predictor's list. With ℓ_r corresponding to the weight associated with a sample at rank r and L_i^r signifying the net loss of predictor i on samples $s_i^t : \{t = 1, \ldots, r\}$; that is, $L_i^r = \sum_{t=1}^{r} \text{sgn}(\text{rel}(s_i^t)) \cdot \ell_t$, we may define the point at which a sample is equally likely to be drawn from list i and j as:

$$\beta^{-L_i^{r_i - 1}} \ell_{r_i} = \beta^{-L_j^{r_j - 1}} \ell_{r_j}$$

with r_x corresponding to the rank of the highest scoring unlabeled sample from the respective lists. Thus, the Hedge algorithm enforces sampling rates inversely proportional to the losses of the predictors via an exponential restoring force.

That is, at a particular instant in the sampling process, we expect the greedy sampling strategy to attempt to force equality of the sample weights of predictors i and j, with the ratio of these quantities defined by:

$$\frac{\ell_{r_i}}{\ell_{r_j}} = \frac{\beta^{L_i^{r_i - 1}}}{\beta^{L_j^{r_j - 1}}}$$

Relevance at rank curves for our application are assumed to be monotonically decreasing, resulting in a weight disadvantage to items drawn later in the list. Therefore a uniform relevance at rank s.t. $r_s = r_t$ for all values of s and t provides a useful bounding case for defining the maximal difference in sampling depths of predictors.

In this case, in the absence of rank effects, the algorithm simply attempts to maintain uniformity of predictor weights:

$$w_0 \cdot \beta^{-L_i} \approx w_0 \cdot \beta^{-L_j}.$$

If we define the expected accuracy of the predictor in terms of the average relevance over an entire list (of arbitrary length R):

$$\hat{p}_i = \frac{\sum_{t=1}^{R} \mathrm{sgn}(\mathrm{rel}(s_i^t))\ell_t}{\sum_{t=1}^{R} \ell_t},$$

the expected loss on samples $S_i^r = \{s_i^1, \ldots, s_i^r\}$ is given by:

$$\hat{L}_i^r = \sum_{t=1}^{r} (1 - \hat{p}_i)\ell_t - \hat{p}_i \ell_t = \sum_{t=1}^{r} (1 - 2 \cdot \hat{p}_i)\ell_t,$$

For uniform relevance at rank, $\ell_t = 1$, this is simply $\hat{L}_i^r \approx |S_r|(1 - 2\hat{p}_i)$. Substituting into the approximate relative loss of predictors,

$$\frac{\hat{L}_j^{r_j}}{\hat{L}_i^{r_i}} = \frac{\sum_{t=1}^{r_j} (1 - 2 \cdot \hat{p}_j)\ell_t}{\sum_{t=1}^{r_i} (1 - 2 \cdot \hat{p}_i)\ell_t},$$

yields:

$$|S_j| \approx |S_i|(1 + 2(\hat{p}_j - \hat{p}_i))$$

and

$$\frac{|S_j|}{|S_i|} \propto \frac{\hat{p}_j}{\hat{p}_i} + c.$$

For relevance at rank curves which are monotonically decreasing ($\ell_t > \ell_{t+1}$), the algorithm maintains a shallower band of sampling depths, and thus an even broader (more uniform) sampling rate across the predictors than in the uniform case.

Oversampling due to Predictor Bias

Having established that in the case of independent predictors sampling of an arbitrary predictor's list occurs at a rate proportional to the accuracy of the predictor, we may now address the question of the effect of oversampling of a portion of the predictor space due to correlation or bias among the predictors.

The oversampling rate in the worst case is easily defined. For a set of N predictors, with an initial uniform weighting we may have a worst case bias due to perfect correlation of $N-1$ predictors. Extending our previous discussion, we define the point at which the expected contribution of the highest ranked sample of predictor i is approximately equal to that of a set of $N-1 \approx N$ correlated predictors j to be:

$$\beta^{-\hat{L}_i^{r_i}} \ell_{r_i} \geq N \cdot \beta^{-\hat{L}_j^{r_j}} \ell_{r_j}$$

Again examining the uniform relevance curves ($\ell_{r_i} = \ell_{r_j}$) to determine worst case behavior of the greedy sampling strategy, we may define the maximum loss sustained by a correlated predictor j relative to independent predictor i as:

$$\hat{L}_j^{r_j} - \hat{L}_i^{r_i} \quad \leq \quad \ln_\beta N$$

Substituting the approximate loss for the uniform relevance curve ($\hat{L}_i^r \approx |S_r|(1-2\hat{p}_i)$), a simple expression for the rate of oversampling of the correlated predictors can be obtained for the case of equivalent accuracies $\hat{p}_j = \hat{p}_i$:

$$|S_j| - |S_i| \leq \frac{\ln_\beta N}{1 - 2 \cdot \hat{p}_{i,j}}$$

Since the quantity of oversampling of the correlated predictors is constant, the effect diminishes linearly with depth of sampling.

Chernoff/Hoeffding Bounds on Estimated Predictor Accuracy

We may diminish the effect of oversampling due to correlations in the predictor space by adopting an alternate technique for establishing the quality of predictors which applies Chernoff/Hoeffding bounds to the estimation of predictor accuracy. Rather than calculating the accuracy of the predictors via some common measure such as average precision on the sample pool, we may calculate the predictor accuracy directly, given a weighting of the relevance scores consistent with our relevance at rank curves. This technique has the advantage of providing confidence rated bounds for arbitrary sampling depths.

In the usual context, Chernoff/Hoeffding Bounds [51] are defined for a sequence of independent Bernoulli trials $X_1, \ldots, X_m$ with probability of success $\mathbf{E}[X_i] = p$, and $S = X_1 + \ldots + X_m$ corresponding to a random variable indicating the total number of successes, so that $\mathbf{E}[S] = pm$. However, in the current context, we shall be interested in the convergence of the variable $\hat{p}_i$ to its expected value with $\hat{p}_i$ defined as follows:

$$\hat{p}_i = \frac{\sum_{t=1..N} \operatorname{sgn}(\operatorname{rel}(s_i^t))\ell_t}{\sum_{t=1..N} \ell_t}$$

Thus, the variable of interest S_i shall be defined in terms of the aggregate of expected outcome over trials so that $m^t = \sum_{t=1..N} \ell_t$. Given this specification of S and m^t, the bounds may be applied directly.

In their usual form, multiplicative bounds for $\lambda_\pm$ s.t. $0 \leq \lambda_- \leq 1$ and $0 \leq \lambda_+$, are given by:

$$Pr[S > (1+\lambda_+)pm] \leq e^{-mp\lambda_+^2/3}$$

$$Pr[S < (1-\lambda_-)pm] \leq e^{-mp\lambda_-^2/2}$$

Note that we may equivalently consider the observed success probability $\hat{p} = S/m$ rather than the actual number of successes S, by eliminating m from the l.h.s.

First, we shall develop a joint bound defining *closeness* of the estimated quantity $\hat{p}$ to the actual value p in terms of λ, so that $p/c \leq \hat{p} \leq cp$, with confidence at least $1-\delta$. The joint bound may be stated, either in terms of λ_- or λ_+, noting that an arbitrary value of $c = 1+\lambda_+$ and $c^{-1} = 1-\lambda_-$ leads to the following relationships:

$$\lambda_- = \frac{\lambda_+}{1+\lambda_+}\ ,\ \lambda_+ = \frac{\lambda_-}{1-\lambda_-}$$

The joint bound is given by:

$$\begin{aligned} Pr[\hat{p} > (1+\lambda_+)p \vee \hat{p} < (1+\lambda_-)p] &= Pr[(\hat{p} > (1+\lambda_+)p] + Pr[\hat{p} < (1+\lambda_-)p] \\ &\leq e^{-mp\lambda_+/3} + e^{-mp\lambda_-/2} \\ &\leq 2e^{-mp\lambda_-/\eta} \end{aligned}$$

with $\eta = 3$ for $\{\lambda_- \leq 1 - \sqrt{2/3}$ and $\lambda_+ \leq \sqrt{3/2} - 1\}$, i.e. for $c = \sqrt{3/2}$, and $\eta = 2$ otherwise. Solving for $e^{-mp\lambda_-^2/\eta} \leq \delta$ gives an expression for the minimum number of samples required:

$$m(p, \lambda_-, \delta) \geq (\eta/(p\lambda_-^2)) \ln(2/\delta).$$

To apply this bound in our present context, we may define sampling bounds in terms of the accuracy of our best predictor, requiring sampling at a minimum rate $m(p_{\text{best}}, \lambda_-, \delta)$ for each predictor. Since the bound has a dependence on p of $O(1/p)$, by fixing the sampling rate across predictors in terms of the expected accuracy of the best predictor, we expect the relative quality of predictor evaluations to vary in direct proportion to the predictor accuracy. Note also that, to maintain constant values of m and δ, the parameter λ_- must vary inversely with the square root of the predictors' success ratios, so that $\frac{\lambda_2}{\lambda_1} \propto \sqrt{\frac{\hat{p}_1}{\hat{p}_2}}$.

4.4 Optimization of Beta

The Hedge algorithm of [37] establishes a single optimal value of β for the entire learning sequence based on the expected loss of the best underlying predictor L_{best} and the total number of predictors N. In the traditional Hedge setting, the sample pool is assumed to be drawn uniformly from the sample space with stationary assumptions of predictor accuracy. Although the evolution of the loss values with rank in Rankhedge suggest the possibility that our selection of β might likewise evolve as the ranked lists are descended, we shall limit our discussion to stationary selection of β and allow the decreasing probability of relevance of the samples to manifest directly in the loss function. Given this strategy, the proper definition of the loss of the best predictor L_{best} corresponds to the expected loss of the best predictor at the first sample point or rank.

The optimal value for β under these assumptions is equivalent to that of the original Hedge algorithm [37]. In the following Lemma, we provide a derivation of the expression for β which minimizes the worst case loss of the Hedge algorithm

Lemma 1 *The value of β which minimizes the worst case loss bound of the Hedge algorithm is given by:*

$$\beta = \frac{1}{1 + \sqrt{\frac{2 \ln N}{L_{\text{best}}}}} \tag{4.9}$$

Proof: *Briefly substituting $\tilde{L} = L_{\text{best}}$ and $\tilde{R} = \ln N$ and applying the approximation $-\ln \beta \leq (1 - \beta^2)/(2\beta)$ for $\beta \in (0, 1]$, we define the following worst case bound on Hedge behavior:*

$$\frac{-\tilde{L} \ln \beta + \tilde{R}}{1 - \beta} \leq \frac{\tilde{L}}{2\beta} + \frac{\tilde{L}}{2} + \frac{\tilde{R}}{(1 - \beta)} \tag{4.10}$$

Differentiation yields:

$$\frac{dL_{Hedge(\beta)}}{d\beta} \leq -\frac{\tilde{L}}{2\beta^2} + \frac{\tilde{R}}{(1 - \beta)^2} \tag{4.11}$$

To minimize, we set the r.h.s. to zero, resulting in:

$$\beta \geq \frac{1}{1 + \sqrt{\frac{2\tilde{R}}{\tilde{L}}}} = \frac{1}{1 + \sqrt{\frac{2 \ln N}{L_{\text{best}}}}} \tag{4.12}$$

The minimum choice of β which satisfies the equation results in the maximum *learning rate.* □

4.4.1 Sensitivity to Selection of Beta

Given this technique for selecting an optimal β, it is straightforward to establish bounds on sensitivity of the Hedge loss due to deviations in the parameter. We shall examine bounds on the effect of overestimation or underestimation of β by establishing the effect of errors in estimating the volume of the predictor set N or the accuracy of the best predictor L_{best}.

A value of β greater than optimal results in a slower than optimal convergence rate. This may be due to either underestimation of the accuracy (overestimation of the loss) of the best predictor or underestimation of the volume of the predictor set. A straightforward bound on this effect may be obtained by focusing on the denominator of the Hedge bound:

$$L_{\text{Hedge}_\beta} \leq \frac{-L_{\text{best}} \ln \beta + \ln N}{1-\beta}$$

We shall introduce the result of an inaccurate estimate of L_{best} or N as a function of $u = \frac{\ln N}{L_{\text{best}}}$, by defining the value γ to be the ratio of actual and estimated values of u, so that $\hat{u} = \gamma \cdot \frac{\ln N}{L_{\text{best}}}$. From Equation 4.9, we see that in the case of overestimation of β we are interested in values $0 \leq \gamma < 1$. Substituting the expression for optimal β from Equation 4.9 into the denominator, the Hedge bound may be rewritten:

$$\begin{aligned} L_{\text{Hedge}_\beta} &\leq \frac{-L_{\text{best}} \ln \beta + \ln N}{1-\beta} \\ &\leq \frac{-L_{\text{best}} \ln \beta + \ln N}{1 - \left(1 + \sqrt{\gamma \cdot \frac{2 \ln N}{L_{\text{best}}}}\right)^{-1}} \\ &= \Big(- L_{\text{best}} \ln \beta + \ln N \Big) \Big(1 + \Big(\gamma \cdot \frac{2 \ln N}{L_{\text{best}}} \Big)^{-\frac{1}{2}} \Big) \qquad (4.13) \end{aligned}$$

Thus, the change in the Hedge worst case bound induced by suboptimality of the β parameter may be written, in this case, as:

$$L_{\mathrm{Hedge}_\beta} - L_{\mathrm{Hedge}_{\beta_{\mathrm{opt}}}} \leq \Big(\sqrt{2}\cdot(\gamma^{1/2}-1)\Big)\cdot\Big(-L_{\mathrm{best}}\ln\beta+\ln N\Big)\Big(\frac{\ln N}{L_{\mathrm{best}}}\Big)^{-\frac{1}{2}} \quad (4.14)$$

making the substitution $\lambda = 1$ for $L_{\mathrm{Hedge}_{\beta_{\mathrm{opt}}}}$.

A similar expression is available in the case in which β is underestimated. In this situation, a convergence rate faster than optimal results either from an overestimation of accuracy of the best predictor (underestimation of the loss) or an overestimation of the volume of the predictor set. Here, convergence of the algorithm is slowed due to the excessive importance placed on each sample, which causes the Hedge algorithm to wander as it overcompensates for noise in the sample set. In the most extreme case, if the best predictor is assumed, erroneously, to be lossless ($L_{\mathrm{best}} = 0 \Rightarrow \beta = 0$), the algorithm may settle rapidly on an incorrect prediction, with an indeterminate effect on the bound.

For the case in which β is underestimated, we restrict γ so that $\gamma > 1$ and focus on the numerator of the worst case bound. We may further simplify by employing the approximation: $-\ln\beta \leq (1-\beta^2)/2\beta$ for $\beta \in (0,1]$.

$$\begin{aligned} L_{\mathrm{Hedge}_\beta} &\leq \frac{-L_{\mathrm{best}}\ln\beta}{1-\beta} + \frac{\ln N}{1-\beta} \\ &\leq L_{\mathrm{best}}\cdot\frac{(1+\beta)}{2\beta} + \frac{\ln N}{1-\beta} \\ &= L_{\mathrm{best}}\cdot\frac{1}{2}\Big(1+\sqrt{\gamma\cdot\frac{2\ln N}{L_{\mathrm{best}}}}\Big) + \frac{\ln N}{1-\beta} \qquad (4.15) \end{aligned}$$

Since the term related to the number of predictors is decreasing with diminished β, we may replace it with a constant, yielding a bound on the error due to underestimation of β:

$$L_{\mathrm{Hedge}_\beta} - L_{\mathrm{Hedge}_{\beta_{\mathrm{opt}}}} \leq \Big(\frac{1}{\sqrt{2}} \cdot (\gamma^{1/2} - 1)\Big) \cdot L_{\mathrm{best}} \cdot \Big(\frac{\ln N}{L_{\mathrm{best}}}\Big)^{\frac{1}{2}} \tag{4.16}$$

$$\leq \Big(\frac{1}{\sqrt{2}} \cdot (\gamma^{1/2} - 1)\Big) \cdot \Big(- L_{\mathrm{best}} \ln \beta + \ln N\Big) \cdot \Big(\frac{\ln N}{L_{\mathrm{best}}}\Big)^{\frac{1}{2}} \tag{4.17}$$

The relaxed bound in Equation 4.17 is included here for comparison with the earlier bound of Equation 4.14.

At this point, we may also make a statement about sensitivity of the worst case bounds to errors in estimation of the constant associated with the rate of decrease of the rank relevance curve described in Equation 4.7. As demonstrated in Section 4.2, variations in the parameter controlling rate of decay result in a proportional increase or decrease in the number of relevant documents discovered due to the properties of the inverse rank weighting (roughly $Pr(d \in S_r) \propto \frac{1}{c|S_r|}$ for document d drawn from document space S_r with rank scaling parameter c). Thus, errors in the estimation of c may be incorporated directly into the estimate of γ via L_{best}, with the resulting sensitivity, again, described by Equations 4.14 and 4.16.

Chapter 5

Hedge and Bayesian On-line Prediction

This chapter examines the Hedge algorithm in both Bayesian and information-theoretic contexts and demonstrates the intimate connection of the two perspectives. First, we consider the Hedge prediction rule as a Bayesian rule with the distribution over predictors corresponding to the best instantaneous prediction of posterior probability of relevance of documents. Analysis of the Bayesian rule as an optimal resource allocation (or gambling) strategy demonstrates an explicit connection between the Bayesian and information theoretic frameworks. In the Bayesian framework, the exponential update rule provides a continuous approximation to the discrete process defining the evolution of posterior probabilities associated with predictors. We demonstrate that, in the information theoretic context, the exponential update rule results in a maximum entropy (minimally constrained) distribution over the predictors, given a particular choice of β and constraints introduced by the labelled data. In addition, information theoretic analysis of the Hedge algorithm provides an exact loss bound in terms of the Kullback Liebler distance of the evolving weight distribution from that of the original prior. Finally, upper and lower bounds for the Hedge loss, derived in the context of the exact analysis, are proven to be consistent with the worst case Hedge loss bound of [37].

To adapt the original Rankhedge algorithm to the Bayesian framework, we modify the loss function to reflect the prior probability of relevance at rank, rather than the log prior (derived from the Average Precision measure) of our earlier experiments. Given the Bayesian interpretation of both the Hedge update rule and classification rule established in the following sections, we shall refer to the Hedge or Rankhedge algorithm with a loss function scaled to reflect the expected prior probability of relevance of the samples as a "pseudo-Bayesian" algorithm. Experimental results at the conclusion of the chapter clearly demonstrate the superiority of the "pseudo-Bayesian" Rankhedge to all other systems tested.

5.1 Bayesian On-line Prediction

Cesa-Bianchi, et al. [19] present a Bayesian method for sequentially predicting Boolean sequences via the linear combination of expert advice which is a close relative of the Hedge algorithm. In this Bayesian on-line setting, we are given an input sequence $\mathbf{x}^T$ drawn from the instance space $x \in X$ and a collection of noisy experts $\mathcal{E} = \{E_1, \ldots, E_N\}$. The goal is to predict the target binary sequence $\mathbf{y}^T$. The value of target instance y^T may be determined in Bayesian fashion by selecting an expert E_i, at round t, with probability defined by the prior distribution Q over the set $\mathcal{E}$ and then corrupting that prediction $E_i(x^t)$ according to the noise model associated with E_i. Distribution Q reflects the prior probability of accuracy of the individual predictors in set $\mathcal{E}$, and thus, the technique guarantees that the expectation of instances in the space of possible target sequences is equivalent to the actual posterior distribution over target sequences defined by Q and the accuracy of the noisy experts.

The Bayesian on-line prediction algorithm iteratively assigns labels in accordance with a Bayesian optimal classification strategy based on the posterior probabilities of the experts, given the observed sequence $\mathbf{y}^{t-1}$. The evolving posterior distribution over the expert set is defined by the posterior probability of relevance of E_i conditioned by an initial prior $P_0(E_i)$ and sequence of evidence $\mathbf{y}^{t-1}$. The initial prior $P_0(E_i)$, as an initial estimate of the maximum accuracy of E_i, may differ significantly from the actual prior Q, and, in the absence of evidence, the initial prior P_0 is often defined to be uniform over the predictors. The posterior probability $P(E_i|\mathbf{y}^{t-1})$ is computed in accordance with Bayes' rule (with likelihood $P_i(\mathbf{y}^{t-1}) = P(\mathbf{y}^{t-1}|E_i)$):

$$P(E_i|\mathbf{y}^{t-1}) = \frac{P_i(\mathbf{y}^{t-1})P_0(E_i)}{\sum_{j=1}^{N} P_j(\mathbf{y}^{t-1})P_0(E_j)} \tag{5.1}$$

with $P_i(\mathbf{y}^0) = 1$.

5.1.1 Classification Rules

Figure 5.1 presents the Bayesian On-line Prediction algorithm in its simplest form, with posterior probabilities associated with experts $P_i(\mathbf{y}^{t-1})$ defined implicitly. While the method of [19] deals primarily with the Bayes or "thresholded majority" classifier, it is worthwhile, in the current context, to discuss both the Bayes and Gibbs methods for classification. Under the naive assumption of independence of experts, we may express the probability $P(y^t|\mathbf{y}^{t-1})$ as a weighted sum of the expert predictions $P_i(y^t|\mathbf{y}^{t-1})$. Thus,

$$P(y^t|\mathbf{y}^{t-1}) = \sum_{i=1}^{N} P_i(y^t|\mathbf{y}^{t-1})P(E_i|\mathbf{y}^{t-1}) \tag{5.2}$$

```
General Bayesian On-line Predictor (Bayes Classifier):

Parameters:

        set of $N$ experts $E_1, \ldots, E_N$.
        noise model defining distributions $P_1, \ldots, P_N$ over $\{0,1\}^\infty$.

Do for $t = 1, 2, \ldots$

    1. If $\sum_{i=1}^N P_i(1|\mathbf{y}^{t-1})P_i(\mathbf{y}^{t-1}) > \sum_{i=1}^N P_i(0|\mathbf{y}^{t-1})P_i(\mathbf{y}^{t-1})$ predict 1,
       else if $\sum_{i=1}^N P_i(1|\mathbf{y}^{t-1})P_i(\mathbf{y}^{t-1}) < \sum_{i=1}^N P_i(0|\mathbf{y}^{t-1})P_i(\mathbf{y}^{t-1})$ predict 0,
       else flip an unbiased coin.
    2. Observe bit $y^t$.
```

Figure 5.1: General Bayesian On-line Prediction.

There are two obvious candidates for a classifier based on the posterior probability measure. The first method, employed in [19], is the *Bayes optimal classification rule*. This prediction method (shown in the algorithm of Figure 5.1) is a weighted majority voting method in which the label output by the algorithm is the thresholded majority label as voted by the experts. That is, in each round t, the Bayes rule simply outputs the label with the highest posterior probability, given $\mathbf{y}^{t-1}$:

$$\hat{y}^t = \arg \max_{y \in \{0,1\}} P(y|\mathbf{y}^{t-1}) \tag{5.3}$$

with the case $P(y=0|\mathbf{y}^{t-1}) = P(y=1|\mathbf{y}^{t-1})$ decided by an unbiased coin flip.

The Bayes classifier is appropriate to the current context. However, in many settings the Bayesian method suffers from the practical and philosophical drawback that the hypothesis used to predict $\hat{y}^t$ at some time t may not reside in the target class. This is of particular concern in the case of finite function classes. To alleviate these concerns, an alternate classification rule, the *Gibbs classification rule*, may be defined as follows:

- Given experts $\{E_1, \ldots, E_N\}$, randomly select an expert $\hat{E}$ according to the distribution over posterior probabilities $P(E_i|\mathbf{y}^{t-1})$.

- Predict $\hat{y}^t = \hat{E}(x^t)$.

If we consider the expectation of label values output by the Gibbs classifier, we see that the algorithm actually produces a distribution over the label set with expectation:

$$\bar{y}^t = \mathbf{E}_{y\in\{0,1\}}[y^t = y|\mathbf{y}^{t-1}] \tag{5.4}$$

5.2 Hedge vs. Bayesian On-line Prediction

Given the appropriate choice of loss function, the Hedge algorithm may be viewed as the bounding approximation of a Bayesian on-line algorithm, and in this section, we shall make an explicit connection to the Bayesian On-line method of Cesa Bianchi, et al. As we shall see in this and later sections, both the Bayesian and Hedge methods are grounded in the common framework of proportional (optimal) resource allocation, and thus connections of the algorithms and similar bounds on their behaviour proceed naturally.

If the Hedge loss is defined directly in terms of expected relevance of the instances, i.e. in terms of relevance at rank, the algorithm may be interpreted as the bounding (continuous) approximation of the discrete Bayesian algorithm. Consider a new loss function based on the expected relevance of instances. With the posterior distribution over N experts denoted $\mathcal{P}^t$ and elements $\mathcal{P}_i^t = P(E_i|\mathbf{y}^{t-1})/(Z^t = \sum_{i=1}^N P(E_i|\mathbf{y}^{t-1}))$, the distribution $\mathcal{P}^t$ converges toward the actual posterior distribution given the evidence, with the exponential update rule establishing the maximal growth rate for a particular choice of $\beta = e^{-\alpha}$. Examining the evolution of the weight of a single predictor, we may define the instantaneous growth rate of the posterior probability incrementally in terms of discrete loss L, resulting in the following relationships:

$$\begin{aligned} P_t(E_i) &\approx P_{t-1}(E_i) \cdot (1 - \alpha L) \\ &\geq P_{t-1}(E_i) \cdot \lim_{n\to\infty} (1 - \frac{\alpha L}{n})^n \quad (5.5) \\ &= P_{t-1}(E_i) \cdot e^{-\alpha L} \quad (5.6) \end{aligned}$$

The exponential rule represents the infinitesimal limit of continuous growth over the interval spanned by total loss L, and thus, it is the bounding growth rate for all possible discretizations of the quantity L, given a particular choice of β. In terms of the partials associated with the posteriors, the exponential growth rate is proportional to the current estimate of the posterior probability associated with the expert: $\frac{\partial(P(E_i))}{\partial L} \approx P(E_i) \cdot \alpha$.

Hedge makes no assumption about the noise model associated with the experts, but instead defines the constant β in terms of the accuracy of the best expert. Given an appropriate choice of β, the Hedge approximation corresponds to a lower bound on the actual posterior probabilities assumed by the Bayes on-line algorithm:

$$\begin{aligned} P_i(\mathbf{y}^T) &= \frac{P_i(\mathbf{y}^0) \cdot \prod_{t=1}^{T-1} P_i(\mathbf{y}^t)}{Z^t} \\ &\geq \frac{P_i(\mathbf{y}^0) \cdot \prod_{t=1}^{T-1} e^{-\alpha L_i^t}}{Z^t} \end{aligned}$$

Here $L_i^t \in [0, 1]$ represents the loss assessed to expert E_i due to instance y^t.

Cesa Bianchi, et al. [19] provide worst case bounds for the Bayes on-line algorithm (with N corresponding to the number of predictors):

$$L_{\text{Bayes}}(\mathbf{y}^t) \leq \ln N + \min_{1\leq i\leq N} \ln \frac{1}{P_i(\mathbf{y}^t)}. \quad (5.7)$$

Substituting $\beta^{-L_{\text{best}}(\mathbf{y}^t)}$ for $\max_{1\leq i\leq N} P_i(\mathbf{y}^T)$ and assuming an initial uniform distribution over the predictors, we see that, unsurprisingly, the worst case analysis of Hedge (Equation 3.1):

$$L_{\mathrm{Hedge}_\beta}(\mathbf{y}^t) \quad \leq \quad \frac{\ln N + \ln \beta \cdot L_{\mathrm{best}}(\mathbf{y}^t)}{1-\beta} \tag{5.8}$$

is within a constant factor of the Bayes on-line bound.

In a later section we shall present experimental results which demonstrate superiority of a "pseudo-Bayesian" Rankhedge algorithm, that is, Rankhedge with a loss function based on prior relevance at rank. But for the moment, we shall examine the maximum entropy properties [49, 50] of the successive distributions produced by Hedge. The maximum entropy result likewise arises from the analysis of the exponential update method as a continuous proportional allocation method.

5.3 Information Theoretic Analysis

An information theoretic analysis of the prediction and exponential update rules of the Hedge algorithm reinforces the validity of the Bayesian interpretation of Hedge. In this section, we demonstrate, first, that prediction methods based on proportional allocation of resources (such as Hedge) are in fact optimal. Next, we show that the exponential update rule of Hedge is optimal, in the sense that it minimizes the change in distribution $\mathcal{P}^t$ over the predictors due to the additional constraint associated with the labelling of a new instance, given a particular choice of β. We extend the information-theoretic argument to generate an exact bound for the Hedge loss defined in terms of the evolution of the distribution $\mathcal{P}^t$. And finally, we provide an alternate proof, derived in this information-theoretic context, for both upper and lower bounds on the Hedge loss.

5.3.1 Proportional Resource Allocation

The prediction rules of both the Bayesian on-line algorithm and Hedge may be viewed as proportional allocation schemes which assign resources in direct proportion to the algorithms' best estimates of each predictor's posterior probability of success. Following the argument of Cover and Thomas [24], allocation of resources in quantities disproportionate to the expected reward, either due to mismanagement of the resources [40] or misunderstanding of the actual probability of reward [39], may diminish the reward accrued by the allocation scheme.

The resource allocation problem is naturally expressed in the gambling context. Given a set of predictors $\mathcal{E} = \{E_1, \ldots, E_N\}$, we define a *wealth function* S^t describing a gambler's wealth at round t. At each round, the gambling scheme allocates all resources to the predictors according to distribution (or portfolio) $\mathbf{b}^t$ with probability of success $\mathbf{p}^t$ and payoff $\mathbf{o}^t$. The wealth function, thus, is defined as:

$$S^t = \prod_{t=1}^{T} b^t \cdot o^t \tag{5.9}$$

The *wealth relative* is the factor $\hat{S}^t = \mathbf{b}^t\mathbf{o}^t$ which reflects the factor of increase of the gambler's wealth at round t. The *doubling rate* $W(\mathbf{b}^t, \mathbf{p}^t)$is defined as the expectation of the log of the wealth relative, i. e.

$$W(\mathbf{b}^t, \mathbf{p}^t) = E(\ln(\hat{S}^t)) = \sum_{k=1}^{N} p_k^t \ln b_k^t o_k^t \tag{5.10}$$

Focusing on a single round t, we may simplify the notation, so that we define the *optimum doubling rate* $W^*(\mathbf{p})$ as the maximum doubling rate over all choices of portfolio $\mathbf{b}$. That is,

$$W^*(\mathbf{p}) = \max_b W(\mathbf{b}, \mathbf{p}) = \max_{\mathbf{b}: b_i \geq 0, \sum_i b_i = 1} \sum_{i=1}^{N} p_i \ln b_i o_i \tag{5.11}$$

We wish to show that a proportional allocation of resources is an optimal prediction strategy, and we do so by demonstrating that a choice of portfolio $\mathbf{b} = \mathbf{p}$ maximizes the doubling rate. $W(\mathbf{b}, \mathbf{p})$ may be maximized as a function of $\mathbf{b}$ subject to the constraint $\sum_{i=1..N} b_i = 1$. As a functional with a Lagrange multiplier, this may be written:

$$J(\mathbf{b}) = \sum_{i=1}^{N} p_i \ln b_i o_i + \lambda \sum_{i=1}^{N} b_i. \tag{5.12}$$

Differentiating w.r.t. b_i we have:

$$\frac{\partial J}{\partial b_i} = \frac{p_i}{b_i} + \lambda, \qquad i = 1, 2, \ldots, N \tag{5.13}$$

Notice that the odds associated with the payoff of a predictor have no impact on the choice of optimal portfolio.

Setting the partial derivative to 0 we have:

$$b_i = -\frac{p_i}{\lambda} \tag{5.14}$$

and substitution into the constraint $\sum_i b_i = 1$ yields $\lambda = -1$ and $b_i = p_i$. Thus, we conclude that $\mathbf{b} = \mathbf{p}$ is a stationary point of the function $J(\mathbf{b})$.

Next, we employ the guess and verify method to demonstrate that the proportional allocation does, in fact, result in a *maximal* doubling rate.

Theorem 1 (Proportional Resource allocation is log-optimal) *The optimum doubling rate is given by:*

$$W^*(\mathbf{p}) = \sum_i p_i \ln o_i - H(\mathbf{p}) \tag{5.15}$$

and is achieved by the proportional gambling scheme $\mathbf{b}^* = \mathbf{p}$.

Proof: *Rewrite* $W(\mathbf{b}, \mathbf{p})$ *so that the maximum is obvious.*

$$\begin{aligned} W(\mathbf{b},\mathbf{p}) &= \sum_i p_i \ln b_i o_i \\ &= \sum_i p_i \ln(\frac{b_i}{p_i} p_i o_i) \\ &= \sum_i p_i \ln o_i - H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b}) && (5.16) \\ &\leq \sum_i p_i \ln o_i - H(\mathbf{p}) && (5.17) \end{aligned}$$

with equality iff $\mathbf{p} = \mathbf{b}$. □

We may specifically address the case of optimal sampling rates by defining the odds to be $o_i =$ one-for-one odds. That is, if we allocate a unit of sampling potential according to the distribution, a loss on predictor i results in a loss of p_i, while a success results in the return of the original p_i units of potential. With one-for-one odds, the doubling rate is given by $W(\mathbf{b},\mathbf{p}) = -H(\mathbf{p}) - D(\mathbf{p}||\mathbf{b})$.

5.3.2 The Exponential Update Rule

The preceding discussion established the optimality of proportional resource allocation as a basis for a prediction method, given the assumption of an accurate posterior distribution over predictors (the implicit assumption of the Bayes on-line prediction algorithm). We shall now demonstrate that the exponential update rule of Hedge is an optimal method for estimating the posterior distribution over predictors. This is accomplished via a technique due to Warmuth [97], by which we demonstrate that the exponential update rule minimizes the relative entropy between successive distributions consistent with the labelled constraints.

Given a distribution over a set of strategies $\mathbf{p}^t = \{p_1^t, \ldots, p_N^t\}$, and loss vector $\ell^t = \{\ell_1^t, \ldots, \ell_N^t\}$, we wish to show that application of the exponential update rule at time t produces an optimal distribution at time $t+1$. Although we measure optimality, in this instance, in terms of relative entropy, the technique works in the more general situation in which any Bregman divergence measure is available.

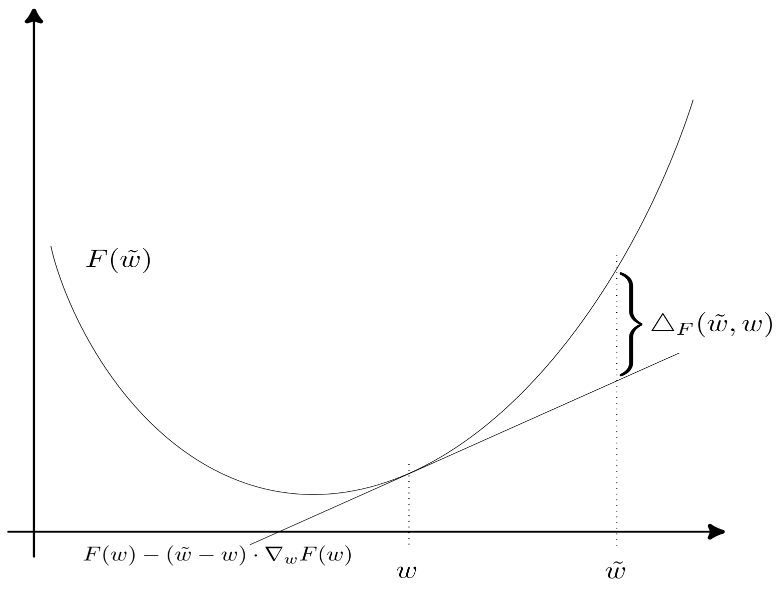

$F(\tilde{w})$
$\triangle_F(\tilde{w}, w)$
$F(w) - (\tilde{w} - w) \cdot \nabla_w F(w)$
w
$\tilde{w}$

Figure 5.2: Bregman Divergences

1. **$\triangle_F(\tilde{w}, w)$ is convex in $\tilde{w}$.**
2. **$\triangle_F(\tilde{w}, w) \geq 0$ with equality iff $\tilde{w} = w$.**
3. **$\nabla_{\tilde{w}} \triangle_F(\tilde{w}, w) = \nabla_{\tilde{w}} F(\tilde{w}) - \nabla_w F(w)$.**
4. **Usually not symmetric: $\triangle_F(\tilde{w}, w) \neq \triangle_F(w, \tilde{w})$.**
5. **Linearity (for $a \geq 0$): $\triangle_{F+aH}(\tilde{w}, w) = \triangle_F(\tilde{w}, w) + \triangle_{aH}(\tilde{w}, w)$.**
6. **Unaffected by linear terms ($a \in \mathbf{R}, b \in \mathbf{R}^n$): $\triangle_{H+a\tilde{w}+b}(\tilde{w}, w) = \triangle_H(\tilde{w}, w)$:**
7. **$\triangle_F(w_1, w_2) + \triangle_F(w_2, w_3) =$ $\triangle_F(w_1, w_3) + (w_1 - w_2) \cdot (\nabla_{w_3} F(w_3) - \nabla_{w_2} F(w_2))$.**

Figure 5.3: Simple Properties of Bregman divergences.

For a differentiable convex function F, we may define the Bregman divergence $\triangle_F(\tilde{w}, w)$ as follows (Figure 5.2):

$$\begin{aligned}\triangle_F(\tilde{w}, w) &= F(\tilde{w}) - F(w) - (\tilde{w} - w) \cdot \bigtriangledown_w F(w) \\ &= F(\tilde{w}) - \text{the supporting hyperplane through } (w, F(w))\end{aligned}$$

Some simple properties of Bregman divergences are given in Figure 5.3.

To prove optimality of the exponential update rule from first principles, we define the distance between distributions at consecutive iterations of the algorithm $\mathbf{p}^{t+1}$ and $\mathbf{p}^t$ in terms of a familiar Bregman divergence measure— the relative entropy or Kullback Liebler distance— measuring the informational loss due to sampling a distribution at the incorrect resolution. The KL distance between successive posterior distributions is defined as:

$$D(\mathbf{p}^{t+1}||\mathbf{p}^t) = \sum_{i=1}^{N} p_i^{t+1} \ln \frac{p_i^{t+1}}{p_i^t} \tag{5.18}$$

Once again using the technique of Langrangian minimization, we may establish that, for this choice of distance measure, the distribution $\mathbf{p}^{t+1}$ closest to current distribution $\mathbf{p}^t$ subject to a particular loss constraint ℓ^t is the one produced by the exponential update rule. We wish to minimize the following functional for choices of λ_1 and λ_2:

$$J(\mathbf{p}^{t+1}) = D(\mathbf{p}^{t+1}||\mathbf{p}^t) + \lambda_1 \sum_{i=1}^{N} p_i^{t+1} + \lambda_2 \sum_{i=1}^{N} p_i^t \ell_i^t \tag{5.19}$$

Partials of the Jacobian are given by:

$$\frac{dJ(\mathbf{p}^{t+1})}{dp_i^{t+1}} = 1 + \ln \frac{p_i^{t+1}}{p_i^t} + \lambda_1 + \lambda_2 \ell_i^t \tag{5.20}$$

We set Equation (5.20) equal to zero and after a bit of algebra arrive at a minimum:

$$p_i^{t+1} = p_i^t e^{-\lambda_2 \ell_i^t - 1 - \lambda_1} \tag{5.21}$$

$$= \frac{p_i^t e^{-\lambda_2 \ell_i^t}}{\sum_{i=1}^N p_i^t e^{-\lambda_2 \ell_i^t}} \tag{5.22}$$

Note that constant λ_1 is scaled so that the resulting vector $\mathbf{p}^{t+1}$ is a distribution. It is straightforward to verify that this is actually a minimum. In the next section, we shall verify that minimization of the relative entropy between successive distributions does, in fact, minimize the loss of the Hedge algorithm.

5.4 Information Theoretic Hedge

In this section, we continue the information-theoretic analysis of the Hedge update rule, deriving an exact form for the loss incurred by the algorithm through examination of the evolution of the weight distribution via the KL divergence. Given $\beta \in [0, 1]$, vector $\mathbf{w}^t$ describing the weight allocation at time t, and loss vector ℓ^t, the update rule is given by:

$$w_i^{t+1} = w_i^t \beta^{\ell_i^t} \tag{5.23}$$

The distribution $\mathbf{p}^t$ in round t is defined in terms of weight vector $\mathbf{w}^t$ and normalization factor Z^t so that

$$p_i^t = \frac{w_i^t}{\sum_{j=1}^t w_j^t} = \frac{w_i^t}{Z^t} \tag{5.24}$$

We shall interpret the posterior distribution $\mathbf{p}^{t+1}$ as reflecting the instantaneous best estimate of the *actual* prior probability of accuracy of the predictors. In this context, we may define the instantaneous loss incurred by the Hedge algorithm at round t in a manner consistent with our earlier information-theoretic discussions by examining the KL divergence $D(\mathbf{p}^{t+1}||\mathbf{p}^t)$ relating the *accurate* prior $\mathbf{p}^{t+1}$ and the current estimate $\mathbf{p}^t$.

Theorem 2 *The instantaneous Hedge loss due to round t is exactly:*

$$L^t_{Hedge} = \frac{D(\mathbf{p}^{t+1}||\mathbf{p}^t) - \ln\left(Z^t/Z^{t+1}\right)}{\ln\beta} \tag{5.25}$$

Proof:

$$\begin{aligned}
D(\mathbf{p}^{t+1}||\mathbf{p}^t) &= \sum_{i=1}^{N} p_i^{t+1} \ln \frac{p_i^{t+1}}{p_i^t} \\
&= \sum_{i=1}^{N} p_i^{t+1} \ln \frac{w_i^{t+1}/Z^{t+1}}{w_i^t/Z^t} \\
&= \sum_{i=1}^{N} p_i^{t+1} \ln \frac{w_i^t \beta^{\ell_i^t}/Z^{t+1}}{w_i^t/Z^t} \\
&= \sum_{i=1}^{N} p_i^{t+1} \ln\left(\beta^{\ell_i^t} \cdot (Z^t/Z^{t+1})\right) \\
&= \sum_{i=1}^{N} p_i^{t+1} \ell_i^t \ln\beta + \sum_{i=1}^{N} p_i^t \ln\left(Z^{t+1}/Z^t\right) \\
&= \ln\beta \sum_{i=1}^{N} p_i^{t+1} \ell_i^t + \ln\left(Z^t/Z^{t+1}\right) \sum_{i=1}^{N} p_i^{t+1} \\
&= \ln\beta \cdot L^t_{\text{Hedge}} - \ln\left(Z^{t+1}/Z^t\right) \qquad (5.26)
\end{aligned}$$

The result follows immediately. □

The advantage of this formulation of the Hedge bound is that it is defined for the entire range of $\beta \in [0, 1]$ including the lossless situation $\beta = 0$. However, it suffers from the fact that it reflects an oversampling of elements of the predictor set spanned by $\mathbf{p}^t$ which may not exist in the target distribution $\mathbf{p}^{t+1}$. Therefore, we shall derive another more practical form for the instantaneous loss of Hedge by reversing the order of terms $D(\mathbf{p}^t||\mathbf{p}^{t+1})$. The following instantiation of the theorem proves to be more intuitive than Theorem (2) and is applicable to all situations save the lossless case $\beta = 0$.

Theorem 3 *For $\beta \in (0, 1]$, the instantaneous Hedge loss due to round t is exactly:*

$$L^t_{Hedge} = \frac{D(\mathbf{p}^t||\mathbf{p}^{t+1}) - \ln{(Z^{t+1}/Z^t)}}{\ln{(1/\beta)}} \tag{5.27}$$

Proof:

$$\begin{aligned}
D(\mathbf{p}^t||\mathbf{p}^{t+1}) &= \sum_{i=1}^{N} p_i^t \ln \frac{p_i^t}{p_i^{t+1}} \\
&= \sum_{i=1}^{N} p_i^t \ln \frac{w_i^t/Z^t}{w_i^{t+1}/Z^{t+1}} \\
&= \sum_{i=1}^{N} p_i^t \ln \frac{w_i^t/Z^t}{w_i^t \beta^{\ell_i^t}/Z^{t+1}} \\
&= \sum_{i=1}^{N} p_i^t \ln \left((1/\beta)^{\ell_i^t} \cdot (Z^{t+1}/Z^t)\right) \\
&= \sum_{i=1}^{N} p_i^t \ell_i^t \ln{(1/\beta)} + \sum_{i=1}^{N} p_i^t \ln{(Z^{t+1}/Z^t)} \\
&= \ln{(1/\beta)} \sum_{i=1}^{N} p_i^t \ell_i^t + \ln{(Z^{t+1}/Z^t)} \sum_{i=1}^{N} p_i^t \\
&= \ln{(1/\beta)} \cdot L^t_{Hedge} + \ln Z^{t+1} - \ln Z^t \qquad (5.28)
\end{aligned}$$

The result follows immediately. □

A closed form expression for the cumulative loss of the Hedge algorithm may now be derived.

Corollary 1 *For $\beta \in (0, 1]$, the cumulative Hedge loss at round $T+1$ is exactly:*

$$L^{T+1}_{Hedge} = \frac{\sum_{t=1}^{T} D(\mathbf{p}^t||\mathbf{p}^{t+1}) - \ln Z^{T+1}}{\ln{(1/\beta)}} \tag{5.29}$$

Proof: *Given our expression for the instantaneous KL distance (5.28), consider* *the sum of KL distances over all rounds:*

$$\begin{aligned}\sum_{t=1}^{T} D(\mathbf{p}^t||\mathbf{p}^{t+1}) &= \sum_{t=1}^{T} \big(\ln(1/\beta)\cdot L^t_{Hedge} + \ln Z^{t+1} - \ln Z^t\big) \\ &= \ln(1/\beta)\sum_{t=1}^{T} L^t_{Hedge} + \ln Z^{T+1} - \ln Z^1 \\ &= \ln(1/\beta)\sum_{t=1}^{T} L^t_{Hedge} + \ln Z^{T+1} \qquad (5.30)\end{aligned}$$

The result follows immediately. □

Next we develop upper and lower bounds on the sum of KL distances— $\sum_{t=1}^{T} D(\mathbf{p}^t||\mathbf{p}^{t+1})$. The following proof relies, again, on convergence of the discrete approximation to the exponential:

$$e^{\pm\alpha L} = \lim_{n\to\infty}(1 \pm \frac{\alpha L}{n})^n. \qquad (5.31)$$

Corollary 2 *For $\beta \in (0,1]$, the cumulative KL distance at round $T+1$ is bounded by:*

$$0 \le \sum_{t=1}^{T} D(\mathbf{p}^t||\mathbf{p}^{t+1}) \le \big(1 - \frac{\ln(1/\beta)}{1-\beta}\big)\ln Z^{T+1} \qquad (5.32)$$

Proof: *The lower bound reflects the fact that the relative entropy is always positive. To demonstrate the upper bound, we define the quantities $Z^{T+1}_{\max}$ and $Z^{T+1}_{\min}$ corresponding to the values of Z^{T+1} for the bounding cases of maximum and minimum Hedge loss, respectively. From Equation (5.29), the minimum loss occurs with $\sum_{t=1}^{T} D(\mathbf{p}^t||\mathbf{p}^{t+1}) = 0$, yielding:*

$$\ln Z^{T+1}_{min} = -\ln\frac{1}{\beta}\cdot L^{T+1}_{Hedge} \qquad (5.33)$$

Substituting into equation (5.29), an upper bound on the aggregate relative en-

tropy in terms of $Z_{\max}^{T+1}$ *is:*

$$\sum_{t=1}^{T} D(\mathbf{p}^t||\mathbf{p}^{t+1}) \leq \ln Z_{\max}^{T+1} - \ln Z_{\min}^{T+1} \tag{5.34}$$

It can by shown, by a convexity argument that $\alpha^r \leq 1-(1-\alpha)r$*; therefore, the rate of change in weights for a particular choice of* β *is bounded above by* $\partial w_i^t/\partial \ell_i^t \leq -(1-\beta)w_i^t$*. Applying the maximum loss to the predictor distribution yields* $\partial Z^t/\partial L^t \leq -(1-\beta)Z^t$*. This suggests an exponential growth rate and the associated inequality:*

$$\big(1-\frac{\alpha L}{n}\big)^n \geq e^{\alpha L} \qquad \forall n \geq 0, \alpha \geq 0 \tag{5.35}$$

Choice of $\alpha = -(1-\beta)$ *yields:*

$$Z_{\max}^{T+1} \geq (\sum_{i=1}^{N} w_i^0)\big(1-\frac{(1-\beta)L_{Hedge}^{T+1}}{n}\big)^n \geq e^{-(1-\beta)L_{Hedge}^{T+1}}. \tag{5.36}$$

This provides a restatement of the Hedge worst case bound from [37]:

$$\ln Z_{\max}^{T+1} \geq -(1-\beta)L_{Hedge}^{T+1} \tag{5.37}$$

The relationship of the minimum and maximum partition entropies is, therefore:

$$\frac{\ln Z_{min}^{T+1}}{\ln Z_{\max}^{T+1}} = \frac{-\ln(1/\beta)\cdot L_{Hedge}^{T+1}}{-(1-\beta)\cdot L_{Hedge}^{T+1}} = \frac{\ln(1/\beta)}{1-\beta} \tag{5.38}$$

Substitution of (5.38) into equation (5.34) completes the proof. □

Finally, we note bounds on Hedge loss which are implicit in the previous discussion.

Corollary 3 *The cumulative Hedge loss at round* $T+1$ *is bounded above and* *below by:*

$$\frac{-\ln\left(\sum_{i=1}^{N} w_i^{T+1}\right)}{\ln(1/\beta)} \le L_{Hedge}^{T+1} \le \frac{-\ln\left(\sum_{i=1}^{N} w_i^{T+1}\right)}{1-\beta} \tag{5.39}$$

Proof: *The bounds proceed trivially from equations (5.33) and (5.37).* □

5.5 Results

Plots 5.4 to 5.9 present the results of experiments with the "pseudo-Bayesian" Rankhedge algorithm which has a loss function based on prior probability of relevance. Examining the topmost curves plotting rate of relevant documents retrieved, one notices immediately a qualitative improvement in performance over the previous Rankhedge incarnation. In these plots, the new Rankhedge closely tracks Rankhedge(Log Prior) in the early stages of the retrieval process where both demonstrate a significant advantage over the priority queue method. In the later stages of the process, when the previous algorithm tends to falter a bit due to excessive concentration on the more successful predictors, the new Rankhedge continues to retrieve relevant samples at a rate significantly superior to the algorithm of Cormack, et al. — the previous best in these regions. The advantage of a properly constructed loss function is most clearly seen in the results for the difficult TREC 9 data, where retrieval rates of the previous algorithm tailed off rapidly after the first 150 documents.

The system evaluation plots present no significant divergences in the behavior of the two Rankhedge algorithms or the priority queue method, although all three methods outstrip the evaluations of Depth-n pooling. The metasearch plots, however, prove more interesting. Note that in conferences 3, 6, 7 and 8 the Rankhedge(Log Prior) method yielded marginally better results than the newer Rankhedge. Taking the scale of the plots into consideration, the advantage of the early algorithm in the metasearch task is relatively small, with the biggest advantage appearing in TREC 8 where the MAP of metasearch lists of

Rankhedge(Log Prior) exceeds that of Rankhedge by approximately three percentage points after judgement of 200 documents. Close examination of the data for these tracks shows that the superior metasearch precision is due to a slight edge in the retrieval rates of Rankhedge(Log Prior) in the shoulder region of the curves, where the increased focus of the early algorithm on the most accurate systems provides a small advantage in total retrieved documents. The difference in performance in these regions is too subtle to be readily distinguished at the scale of the recall plots and does not alter the overall estimation of the qualitative superiority of the "pseudo-Bayesian" Rankhedge algorithm.

As one might expect, the newer Rankhedge surpasses the Rankhedge(Log Prior) results on the TREC 9 data, but, interestingly, neither achieves the precision of the best system in that track. Examination of the results for the actual TREC rankings for this track demonstrates that the best system is an outlier, with a mean average precision of 0.4425 versus the next best system score of 0.3499— a gap of more than 20%. The poor precision of the systems and sparsity of relevant documents in this track make it difficult for either system to recover the losses associated with the early stages of retrieval. Nevertheless, the metasearch lists of the newer Rankhedge method *approach* the quality of the best system and substantially exceed both combMNZ and Condorcet MAP scores as well as that of the second best system.

Finally, we include three dimensional plots taken from TREC 8 of the evolving weight histories of the Rankhedge distribution. The first Figure (5.10) shows the predictor weights averaged across all queries, while the second (5.11) demonstrates the range of behavior manifested on a per query basis. The sample axis of the plots is "pseudo-logarithmic," with sample intervals defined by discretization of the aggregate of the prior. The system axis is sorted according to net weight over the entire sample range, and the weights are scaled at each interval to fix the maximum at 1.0. Note that the plots of Figure 5.10 manifest relatively orderly behavior as the aggregate of the distributions across queries focuses fairly rapidly on the best predictors and maintains that focus throughout the course of the trial. A look at plots of Figure 5.11, however, reveals a range of behavior which is masked in the averaged results. In these plots, Query 407 is representative of the mean behavioral range of Rankhedge. Query 436 provides an example of a difficult query, with sparse relevant documents and lists of low precision returned by most predictors. Query 450 is an example of an easier query, with many relevant documents discovered and relatively good precision across a broad range of systems. Most notable in these plots (and readily seen in Queries 407 and 450) is the tendency of the algorithm to focus on the best predictors in the early stages, with a maximum contraction of the distribution in a transitional region roughly corresponding to the shoulder seen in the earlier plots (Figures 5.4 to 5.9). Beyond this shoulder, the histories demonstrate a broadening of the search for relevant documents as the early entries of the better lists are exhausted, and the algorithm shifts its attention to higher ranked documents in lists of lower precision. Clearly, the relative difficulty of individual queries results in significant qualitative differences in evolution of the on-line process, and the possibility of defining the loss function and β on a per query basis is an area for future research.

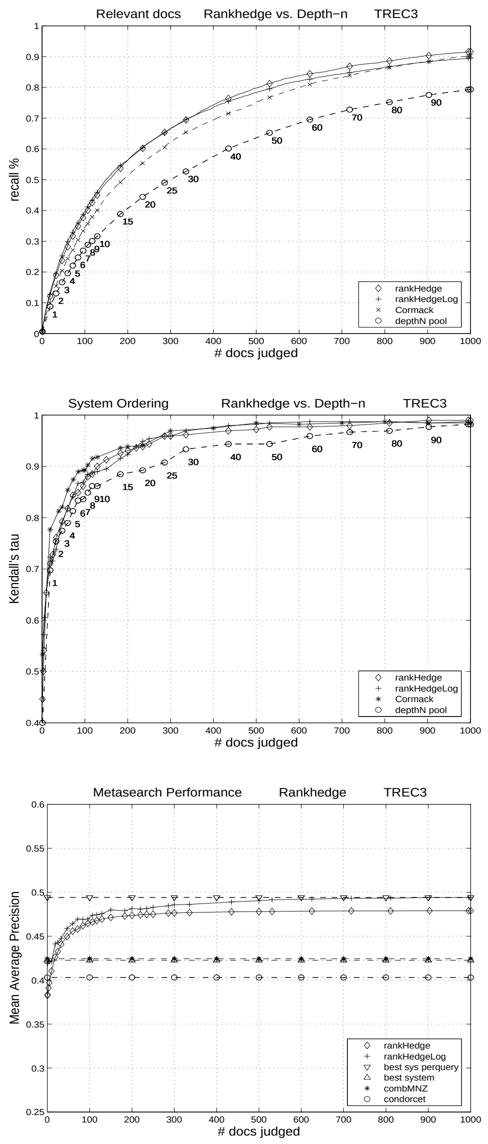
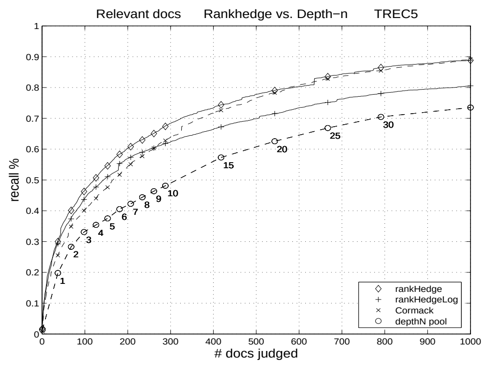
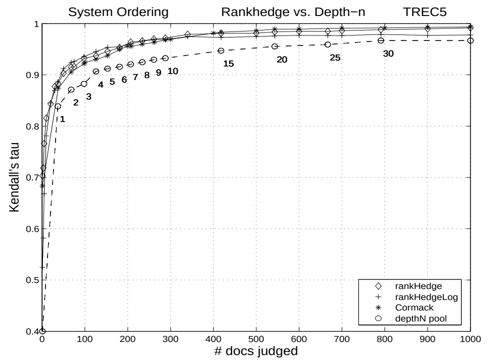


Figure 5.4: Trec 3. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision.





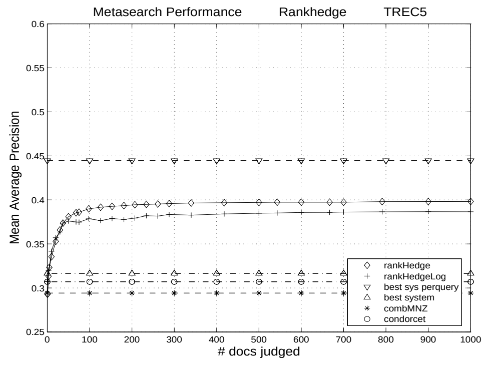


Figure 5.5: Trec 5. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision.

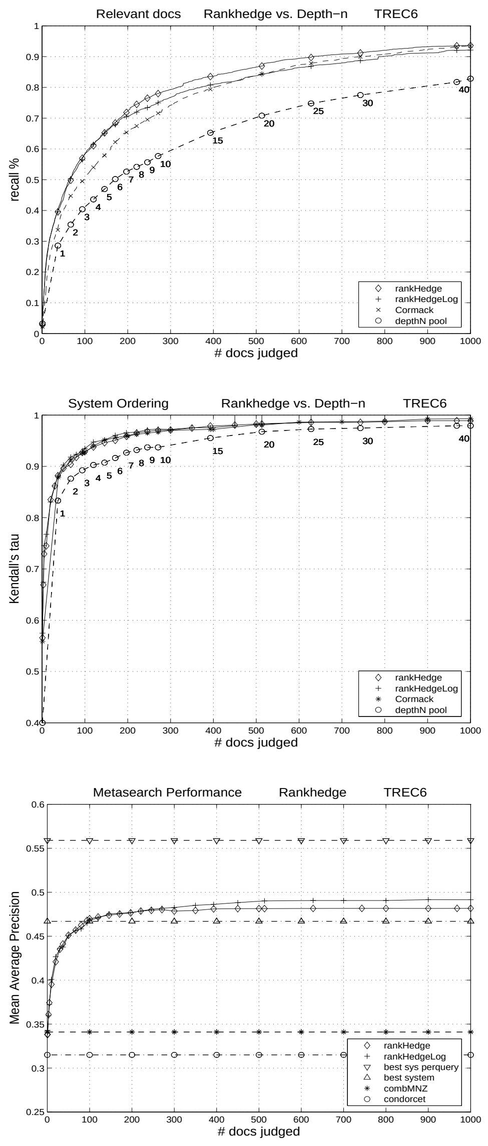


Figure 5.6: Trec 6. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision.

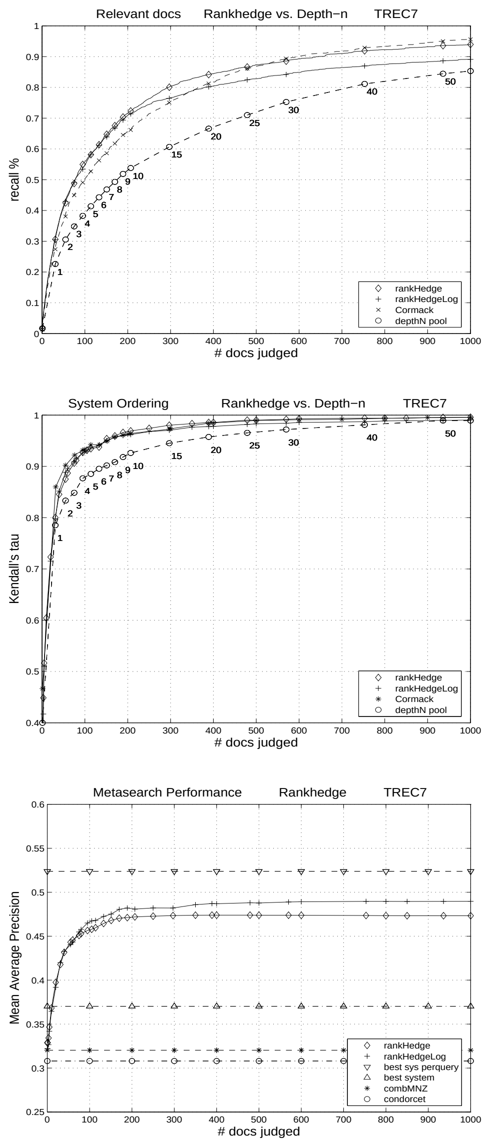


Figure 5.7: Trec 7. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision.

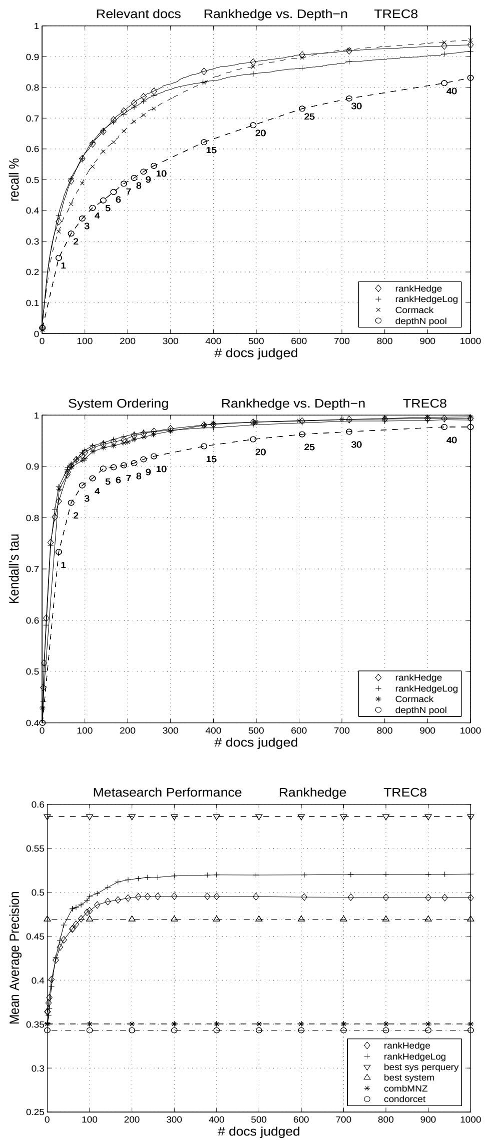


Figure 5.8: Trec 8. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ.(C) metasearch performance via Mean Average Precision.

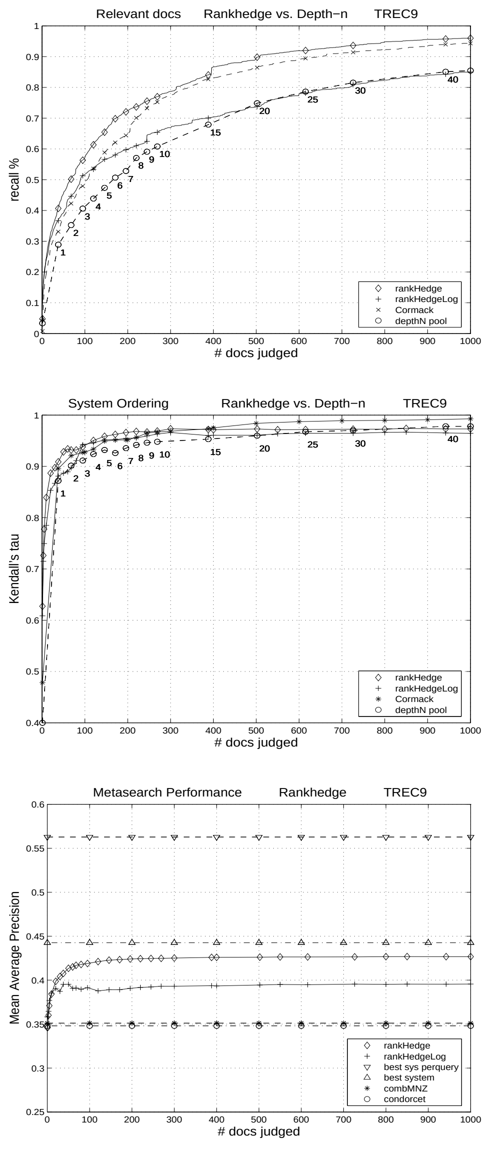
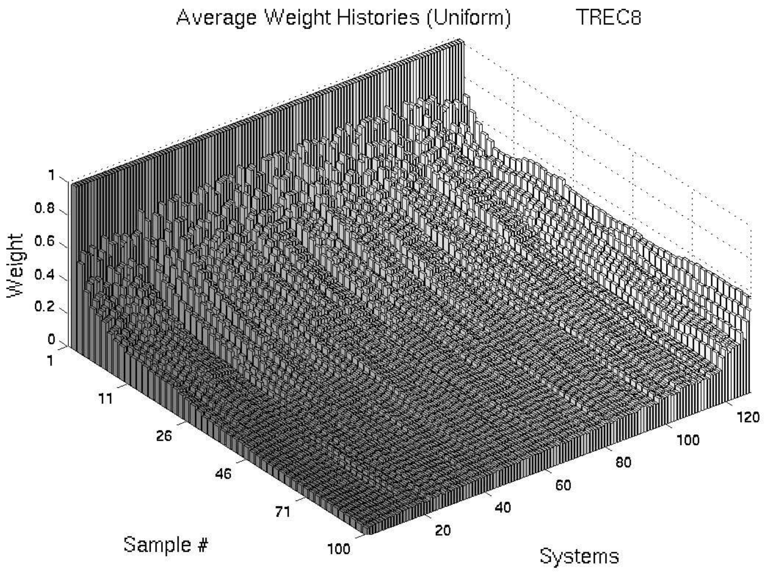


Figure 5.9: Trec 9. Comparison of Rankhedge— prior relevance at rank, rankHedge— log prior, Cormack pooling method, and Depth-n pooling under three performance measures: (A) percent of total relevant documents discovered. (B) system ordering via k-τ. (C) metasearch performance via Mean Average Precision.



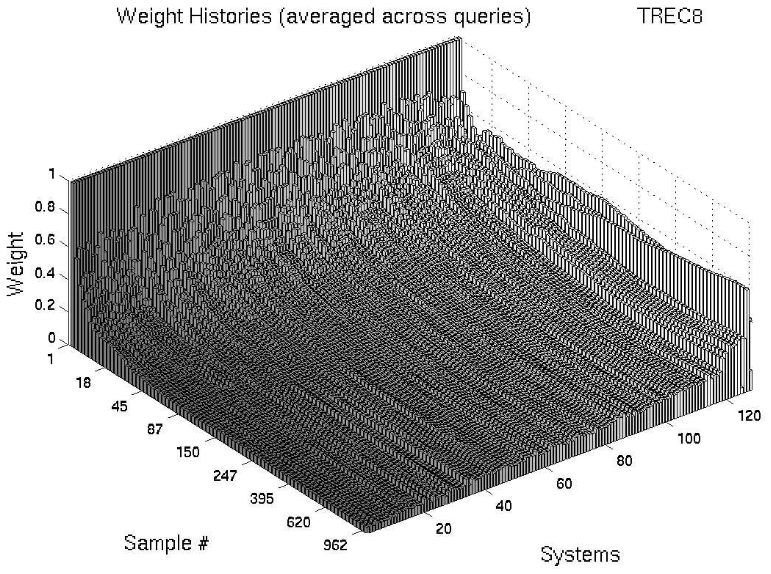
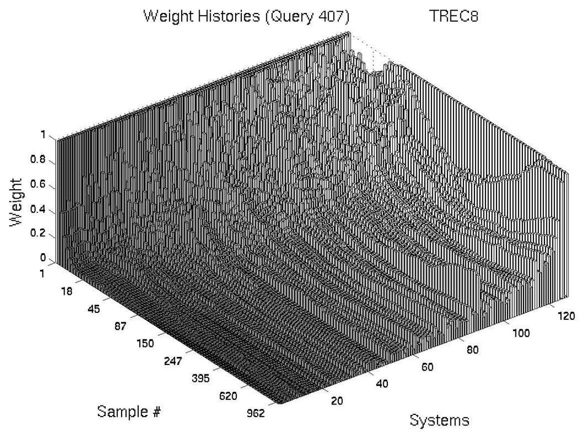
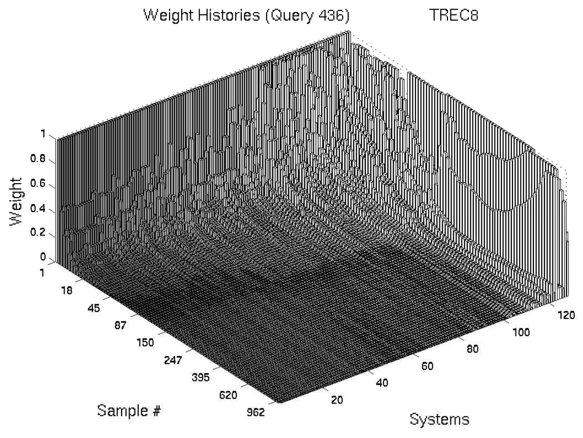


Figure 5.10: Trec 8: Evolution of predictor weights to depths 100 and 1000—averaged across 50 queries. Pseudo-logarithmic sample dimension.





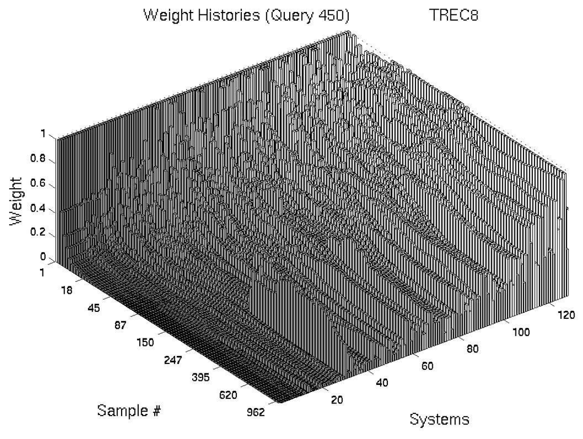


Figure 5.11: Trec 8: Evolution of predictor weights to depth 1000— Queries 407, 436, and 450. Pseudo-logarithmic sample dimension.

Chapter 6

Similarity Measures

Performance of the Hedge algorithm is dependent on both the number and quality of the predictors in the predictor set. In this chapter, we examine the independence assumption which is implicit in the usual choice of uniform initial distribution over predictors and present a theoretically grounded technique for improving Hedge performance by minimizing the volume of the predictor space via *debiasing* or decorrelation of the predictor set.

Bias error is a common term in IR for the effects of predictor correlation in a situation where independence among predictors is assumed. To accurately determine the amount of correlation among predictors, we must first define a similarity measure on the space of predictors. In the case of ranked lists, several options are available, some of which were presented in our discussion in Chapter Two of the Cranking method [55].

In the Rankhedge context, the availability of a prior estimation of relevance at rank allows us to define a similarity measure in local terms on the space of instances and to aggregate these local similarity measurements to define the global similarity score for a pair of predictors. This ability to localize the similarity measure to individual instances is a manifestation of a useful duality— of the local potentials (accuracy) of the predictions defined on the sample neighborhoods and the global accuracy of the predictors as approximated by the Hedge weight vector. We explore the implications of this duality and demonstrate the close relationship of Hedge's distribution over predictors to a well known entity in statistical physics— the Gibbs Distribution— the dual of which is the Markov Random Field (MRF).

We shall make extensive use of the dual properties of the sample/ predictor axes in the next chapter to develop a method for probabilistic directed sampling, but for the moment we shall restrict our attention to the implications for definition of a similarity measure in the predictor space. In the remainder of the chapter, we demonstrate that a concise information theoretic similarity measure arises naturally from the Hedge's implicit mapping of the predictor space. And we employ this measure to properly decorrelate the predictor set via reweighting of the initial distribution over predictors. In addition, we shall briefly revisit the experiments of Soboroff, et al. [84] and show that the success of attempts at blind evaluation of predictor quality discussed in the paper are, in fact, attributable to clustering of predictors in the similarity space.

6.1 Background

The Cranking algorithm [55] introduced in Chapter Two provided an interesting example of a learning algorithm defined explicitly in terms of a distance measure on the space of permutations (the symmetric group $\mathcal{S}_n$ of order n). Cranking is closely akin to Rankhedge in that the conditional probability model employed, the Mallows model, assumes an exponentially decaying influence function between predictors based on an arbitrary distance metric $d : \mathcal{S}_n \times \mathcal{S}_n \mapsto \mathbb{R}^+$, where $\mathcal{S}_n$ is a location parameter from the symmetric group of order n.

Given the evidence of a training set consisting of pairs $(\lambda^{(i)}, \sigma^{(i)})$ where $\lambda^{(i)}$

is a target ranking of instances i and $\sigma^{(i)}$ is the set of rankings returned by the underlying retrieval systems, the Mallows conditional ranking model is denoted $\mathcal{M}_d(\lambda, \theta, \sigma)$, where $\theta \in \mathbb{R}$ is a dispersion parameter and $\sigma \in \mathcal{S}_n$ is a location parameter which in the current context corresponds to a specific instance from the space of ranked lists. The model has the exponential form:

$$p(\lambda|\theta, \sigma) = e^{\theta d(\lambda,\sigma) - \psi(\theta,\sigma)} \tag{6.1}$$

In the Mallows model, the distance measure $d(\lambda, \theta)$ occupies the role of the aggregate loss term of Hedge and is defined relative to label sequence or permutation λ. The measure may be any one of a number of distance metrics $d : \mathcal{S}_n \times \mathcal{S}_n \mapsto \mathbb{R}^+$ such as Kendall's τ (the minimum number of adjacent transpositions needed to bring λ to σ), rank correlation ($R(\lambda, \sigma) = \sum_{i=1}^{n}(\lambda(i) - \sigma(i))^2$), or Spearman's foot rule ($F(\lambda, \sigma) = \sum_{i=1}^{n} |\lambda(i) - \sigma(i)|$). The denominator of the model corresponds to the partition energy with $Z(\theta, \sigma) = e^{\psi(\theta,\sigma)}$, where ψ is the cumulant function $\psi(\theta, \sigma) = \log \sum_{\lambda \in \mathcal{S}_n} \exp(\theta \cdot d(\lambda, \sigma))$.

6.1.1 Hedge and the Predictor Space

We refer to the Cranking algorithm and the Mallows model to introduce the notion of an exponentially decaying influence function on the predictor space. Connections to the Rankhedge algorithm are obvious— with a distance measure on a predictor pair $\theta d(\lambda, \sigma)$ defined as the projected aggregate loss function obtained by fixing either λ or σ as target and $\theta = \ln \beta$ as the rate of decay. In this section, we shall examine the dual relationship of the Hedge weighted predictor set and the field of predictions on instances which is implicit in the use of an exponential influence function. In the following section, an interpretation of the distance measure in the Rankhedge predictor space will be fully developed in the context of *information-theoretic similarity* measures.

The duality of the *local* representation of the state of the Rankhedge prediction on the space of instances and the *global* representation defined in terms of the distribution over the Hedge predictors is closely related to a well known result of statistical physics— the *Hammersfield-Clifford* equivalence property— which defines the relationship between the Gibbs Distribution and Markov Random Field representations of random fields on graphs.

6.1.2 Markov Random Fields

Connections between certain results in computational learning theory and statistical mechanics are examined in [72, 90, 98, 46]. We shall be concerned with a dual informational structure relating a distribution over a continuous basis of predictors and a random field defined on the instance space. In statistical mechanics, the distribution over configurations or predictors is called the Gibbs Distribution and its dual field on the instance space is the Markov Random Field. As noted in [16, 85] almost any multivariate distribution may be interpreted as a Markov Random Field, with the duality of the informational structures established by the *Hammersfield-Clifford* equivalence property [15].

The theory of Markov Random Fields on graphs is presented concisely in Geman and Geman [41]. An MRF is defined on a set of *sites* $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$, with $\mathcal{G} = \{\mathcal{G}_s : s \in S\}$ being a *neighborhood system* for S. A neighborhood system is defined as any collection of subsets of S for which $s \notin \mathcal{G}_s$ and $s \in \mathcal{G}_r \Leftrightarrow r \in \mathcal{G}_s$. Thus $\mathcal{G}_s$ is the set of *neighbors* of s and the pair $\{S, \mathcal{G}\}$ is a graph. A subset $C \subseteq S$ is a *clique* if all pairs of distinct sites in C are neighbors. The set of cliques will be denoted $\mathcal{C}$. Now, if we define $\mathcal{X} = \{X_s \big| s \in S\}$ to be *any* family of random variables indexed by S with a common state space Λ (in this context $\Lambda = [0, 1]$), we may define the set of all possible *configurations*:

$$\Omega = \{\omega = (x_{s_1}, \cdots, x_{s_N}) : x_{s_i} \in \Lambda, 1 \leq i \leq N\} \tag{6.2}$$

We may abbreviate the event $\{X_{s_1} = x_{s_1}, \cdots, X_{s_N} = x_{s_N}\}$ as $\{\mathcal{X} = \omega\}$. Now, $\mathcal{X}$ is a Markov Random Field with respect to $\mathcal{G}$ iff

$$P(\mathcal{X} = \omega) > 0 \qquad \forall \ \omega \in \Omega; \tag{6.3}$$

$$P(X_s = x_s | X_r = x_r, r \neq s) = P(X_s = x_s | X_r = x_r, r \in \mathcal{G}_s) \tag{6.4}$$

for every $s \in S$ and $(x_{s_1}, \cdots, x_{s_N}) \in \Omega$.

In other words, the pair $\{\mathcal{X}, P\}$ must satisfy equations (6.3) and (6.4) relative to some probability measure Ω. The collection of functions on the left hand side of (6.4) is referred to as the local characteristics of the MRF.

6.1.3 Gibbs Distributions

The dual of the Markov Random Field is the Gibbs distribution, which is defined, relative to $\{S, \mathcal{G}\}$, as a probability measure ψ on Ω with the following representation:

$$\psi(\omega) = \frac{1}{Z} e^{-U(\omega)/T} \tag{6.5}$$

where Z and T are constants and U is an *energy function* of the form

$$U(\omega) = \sum_{c \in \mathcal{C}} V_c(\omega) \tag{6.6}$$

Each V_c is a function on Ω with the property that $V_c(\omega)$ depends only on the coordinates x_s of ω for which $x_s \in C$. The family $\{V_c \mid c \in \mathcal{C}\}$ is called a *potential* and Z is the normalizing constant, also called the *partition function*:

$$Z = \sum_{\omega} e^{-U(\omega)/T} \tag{6.7}$$

The weights associated with the set of predictors which are maintained by the Hedge update scheme may be interpreted as a finite sampling of the probability measure $\psi(\omega)$ of Equation 6.5. With $\psi(\omega)$ reflecting the posterior probability of observing configuration ω in the space of configurations Ω, this is consistent with our results of the previous chapter which established the connection of Hedge and Bayesian online prediction. The weighted linear combination of predictions defined by Hedge on a particular instance c may be viewed as measuring the local potential $V_c(\omega)$ established by the MRF at instance c. Interpreted in this manner, Hedge's weighted linear combination of predictions defines a mean field estimate of the MRF potential on the space of instances, or in familiar Bayesian terms, the maximum a posteriori hypothesis given the evidence of the labelled samples and a uniform prior.

The values of $\psi(\omega)$ associated with the Gibbs Distribution define the distribution over the space Ω which has, of all distributions consistent with the constraints $\{V_c(\omega) \mid c \in \mathcal{C}\}$, a minimum KL distance (relative entropy) from uniform. Thus, the Gibbs Distribution may be viewed as a maximum entropy distribution on Ω, given the constraints. Equivalence of Gibbs distributions and MRF's are due to the Hammersfield-Clifford expansion (as developed in [15] which is stated explicitly in the following theorem [41]):

Theorem 4 (MRF Gibbs Equivalence) *Let $\mathcal{G}$ be a neighborhood system. Then $\mathcal{X}$ is an MRF with respect to $\mathcal{G}$ iff $\psi(\omega) = P(\mathcal{X} = \omega)$ is a Gibbs distribution with respect to $\mathcal{G}$.*

6.2 Information Theoretic Similarity Measures

We would now like to define a similarity measure on the space of predictors which is consistent with the probability measure associated with the Gibbs Distribution. We may examine the question of the similarity of two objects A and B or in this case two configurations ω_A and ω_B in information-theoretic terms utilizing the concept of an information-theoretic similarity measure IT-Sim(A, B), introduced by Lin in [59] and further developed by Aslam and Frost in [6]. The measure has a particularly simple interpretation in the Hedge/Gibbs context, with each site s of the sample space constituting an independent dimension or feature whose local potential $V_c(\omega)$ associated with configuration ω fixes the energy function $U(\omega)$ at site s. This local potential in turn fixes a portion of the probability measure $\psi(\omega)$. Given the exponential decay of the probability measure $\psi(\omega_A)$ with increasing deviations of configuration ω_A to a target configuration ω_B, it is straightforward to show that the appropriate information theoretic similarity measure on Ω devolves to a familiar measure of distance on the configuration space.

6.2.1 Background

Proceeding from the six basic assumptions listed in Figure 6.1, Lin [59] defines a general model of similarity which is applicable to any domain which has a probabilistic model. Aslam and Frost further develop the concept of information-theoretic object similarity in [6], focusing specifically on applications to pairwise document similarity.

The concept of information-theoretic similarity between two items A and B is defined in terms of the information content of the common features of A and B— $I(common(A,B))$— and the information content of the statement describing the set of both items— $I(description(A,B))$— as follows:

$$\text{IT-Sim}(A,B) = \frac{I(common(A,B))}{I(description(A,B))} = \frac{\log P(common(A,B))}{\log P(description(A,B))} \tag{6.8}$$

With the information content of a statement x defined by its self information $\log(1/\pi(x))$ where $\pi(x)$ is the probability of the statement within the world of objects, the set of objects which can be described by a set S of independent

1. $I(common(A,B))$ **measures information common to** A **and** B**.**
2. $I(description(A,B)) = I(common(A,B)) + I(difference(A,B))$**.**
3. $sim(A,B) = f(I(common(A,B)), I(description(A,B)))$**.**
4. $\forall x > 0, f(x,x) = 1$**.**
5. $\forall y > 0, f(0,y) = 0$**.**
6. $\forall x_1 \leq y_1, x_2 \leq y_2 : f(x_1 + x_2, y_1 + y_2) = \frac{y_1}{y_1+y_2} f(x_1, y_1) + \frac{y_2}{y_1+y_2} f(x_2, y_2)$

Figure 6.1: Six Assumptions of IT-Similarity.

features s possesses the following similiarity measure [6]:

$$\text{IT-Sim}(A,B) = \frac{2 \cdot \sum_{s \in A \cap B} \log \pi(s)}{\sum_{s \in A} \log \pi(s) + \sum_{s \in B} \log \pi(s)} \tag{6.9}$$

6.2.2 Similarity Measures on the Space of Predictors

In the Hedge/Gibbs context, two information-theoretic similarity measures arise intuitively from the exponential approximation of the posterior accuracy of the predictors. Since the assessment of predictor accuracy is dependent, not only on the value assigned to the instance or feature by the predictor, but also on the actual relevance score of the instance, we may define both an *extremal* similarity measure in which all samples are judged relevant and an *expected* measure with instances weighted by expected relevance of samples.

To define a similarity measure for a predictor space consistent with the distance measure on the space of predictors implicitly defined by the Gibbs Distribution, we associate neighborhood potentials $V_c(\omega)$ with features $\pi(s)$, possibly weighted by a constant factor ($\ln \beta$ in the Hedge context). Since the conditional probability of predictor ω_A given target configuration ω_B is defined by the exponential influence function $\psi(\omega_A|\omega_B)$, taking the log of the relationship simply results in a function of the original feature distance, as we shall demonstrate.

Noting that for $\pi_\omega(s) \in [0,1]$ with $\pi_\omega(s)$ defined as the expected label at s as fixed by predictor ω, the magnitude of overlap of the joint probability space $\pi_{\omega_{(A,B)}}(s) = \pi_{\omega_A}(s) \cap \pi_{\omega_B}(s)$ may be defined as follows:

$$\begin{aligned}
\pi_{\omega_{(A,B)}}(s) &= \min(\pi_{\omega_A}(s), \pi_{\omega_A}(s) \cdot \pi_{\omega_B}(s)) \\
&= \min(\pi_{\omega_B}(s), \pi_{\omega_B}(s) \cdot \pi_{\omega_A}(s)) \\
&= \pi_{\omega_A}(s) \cdot \pi_{\omega_B}(s).
\end{aligned}$$

The information-theoretic similarity measure for the Gibbs distribution is given by:

$$\begin{aligned}
\text{IT-Sim}_{\text{Gibbs}}(A,B) &= \frac{2 \cdot \sum_{s \in A \cap B} \log \pi(s)}{\sum_{s \in A} \log \pi(s) + \sum_{s \in B} \log \pi(s)} && (6.10) \\
&= \frac{2 \cdot \log(e^{\sum_{s \in S} \pi_{\omega_A}(s) \cdot \pi_{\omega_B}(s)})}{\log(e^{\sum_{s \in S} \pi_{\omega_A}(s)^2}) + \log(e^{\sum_{s \in S} \pi_{\omega_B}(s)^2})} && (6.11) \\
&= \frac{2 \cdot \sum_{s \in S} \pi_{\omega_A}(s) \cdot \pi_{\omega_B}(s)}{\sum_{s \in S} \pi_{\omega_A}(s)^2 + \sum_{s \in S} \pi_{\omega_B}(s)^2} && (6.12)
\end{aligned}$$

Note that the expression (6.12) corresponds to the familiar correlation coefficient for discrete events known as the *dice coefficient* [35].

The extremal measure is equivalent to IT-Sim$_{\text{Gibbs}}$, but with $\bar{\pi}_\phi(s)$ defined to reflect the set inclusion property:

$$\bar{\pi}_\phi(s) = \begin{cases} 0 & : \quad s \notin \phi \\ 1 & : \quad s \in \phi \end{cases}$$

Substituting into Equation 6.9, this yields a familiar form of set similarity:

$$\begin{aligned}\text{IT-Sim}_{\text{ext}}(A,B) &= \frac{2\cdot\sum_{s\in S}\bar{\pi}_A(s)\cdot\bar{\pi}_B(s)}{\sum_{s\in S}\bar{\pi}_A(s)^2+\sum_{s\in S}\bar{\pi}_B(s)^2} \\ &= \frac{2\,|A\cap B|}{|A|+|B|}\end{aligned}$$

6.3 Decorrelation of Predictors

Similarity measures in high dimensional feature spaces have been exploited in many practical settings to enable clustering and segmentation. Eigenvector methods on the $N \times N$ distance matrix spanning the instances have proven particularly successful at extracting useful correlations in image processing and web search [47, 71, 82, 99]. In the current context, we adopt an opposite viewpoint and consider inter-predictor similarity to be a potential source of bias error in the classification and system evaluation tasks. In this section, we apply the information-theoretic similarity measure to the simpler task of debiasing the initial distribution over predictors.

We may use the similarity measure $\text{IT-Sim}_{\text{Gibbs}}$ to decorrelate the initial distribution over predictors in a straightforward manner. A net similarity score for a predictor proceeds in an intuitive manner from consideration of the Laplacian [20, 74, 30] of the $N \times N$ similarity matrix (with N the number of systems) implicitly established by the imposition of the distance measure on the space of predictors. For a uniform initial distribution the net- similiarity is simply the sum of similarity scores on the neighborhoods, less the identity:

$$n{\times}n\_sim(A) = \sum_{B\in\mathcal{S}} \text{IT-Sim}(A,B) - \text{IT-Sim}(A,A) \tag{6.13}$$

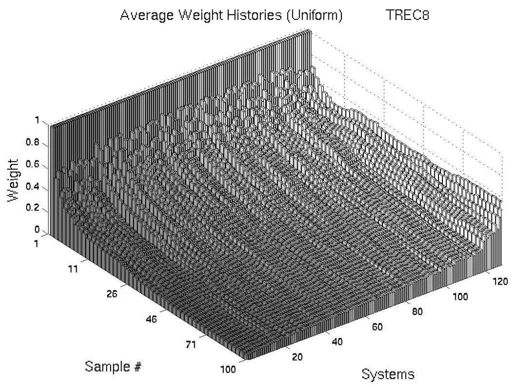
To debias a finite set of predictors, each predictor's initial weight is multiplied by the inverse of its *potential* in the similarity space, so that:

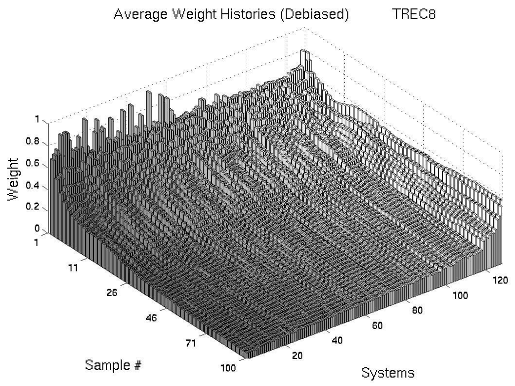
$$w_A = w_A^0 \cdot \frac{1}{1 + n \times n\_sim(A)} \tag{6.14}$$

6.3.1 Results

In experiments with the Rankhedge algorithm on TREC 8 data, the effect of decorrelation with net-similarity scores averaged across queries proved to be negligible for retrieval, system evaluation and metasearch, though it is possible that decorrelation on a per query basis would yield greater effect. A primary reason for the negligible difference in results is the fact that the Rankhedge algorithm naturally addresses the biased regions by actively sampling biased lists at rates approximately proportional to the amount of bias. The parallel weight histories of Figure 6.2 demonstrate the rapid convergence of the biased and unbiased weight distributions in response to relevance feedback. Figure 6.2 c) charts the average ratio of system weights (unbiased/biased) revealing convergence within the first twenty samples. This rapid convergence results from the combination of the exponential decay of Hedge update rule and the active concentration of sampling resources on the biased regions of the distribution due to the directed sampling method. Experiments with decorrelation of predictors on a per-query basis is an item for future research.

We also examine the use of the net system similarity measure in the context of the technique of "pseudoevaluation" [84] (system evaluation in the absence of relevance judgements). As demonstrated in [11], "pseudoevaluation" techniques rely on the assumption that agreement of systems on the quality of an instance is indicative of the actual probability of relevance. While this assumption holds true to a certain extent in the TREC context, it actually constitutes assignment of predictive import to a potentially malicious property of the predictor set. Correlation among poor predictors is, in fact, the worst case scenario which the Hedge algorithm was designed to address. The concluding Figures 6.3 and 6.4 clearly demonstrate the susceptibility of the “pseudoevalution” technique to flaws in its underlying assumptions. The plots in the left hand column compare the system rankings produced by “pseudoevaluation” with systems ranked in ascending order according to their net similarity scores. This method is effectively a system ranking based on probabilistic relevance scores defined by Rankhedge-0. It is comparable to the probabilistic sampling of [84] and produces similar results. As in our earlier discussion (Section 2.1.1) we note the characteristic tail associated with the highest ranked systems. This tail is particularly evident in TRECs 7, 8, and 9 and results from the fact that the best systems produce ranked lists which are significantly different than those of the generic predictors. The plots in the right hand column of Figures 6.3 and 6.4 correspond to the system evaluations produced by Rankhedge pools of size equivalent to Depth-1 pools for each TREC. Results for all TRECs demonstrate the effectiveness of Rankhedge’s use of on-line feedback not only to tighten the overall distribution of system rankings but also to rapidly correct the misclassification of the best systems.





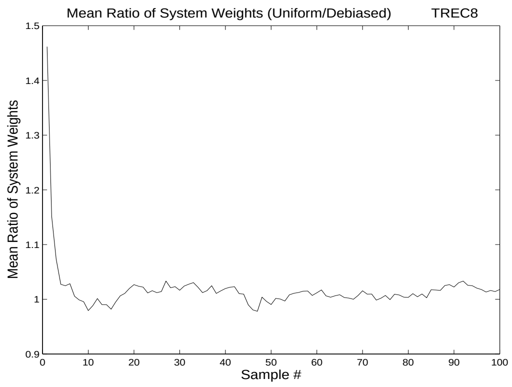


Figure 6.2: Trec 8: Rapid converge of Rankhedge weight history (uniform initial distribution) to corresponding history for initial debiased distribution. Figures a) and b) demonstrate qualitative similarity of evolving histories. Figure c) gives average ratio of system weights at sample depth.

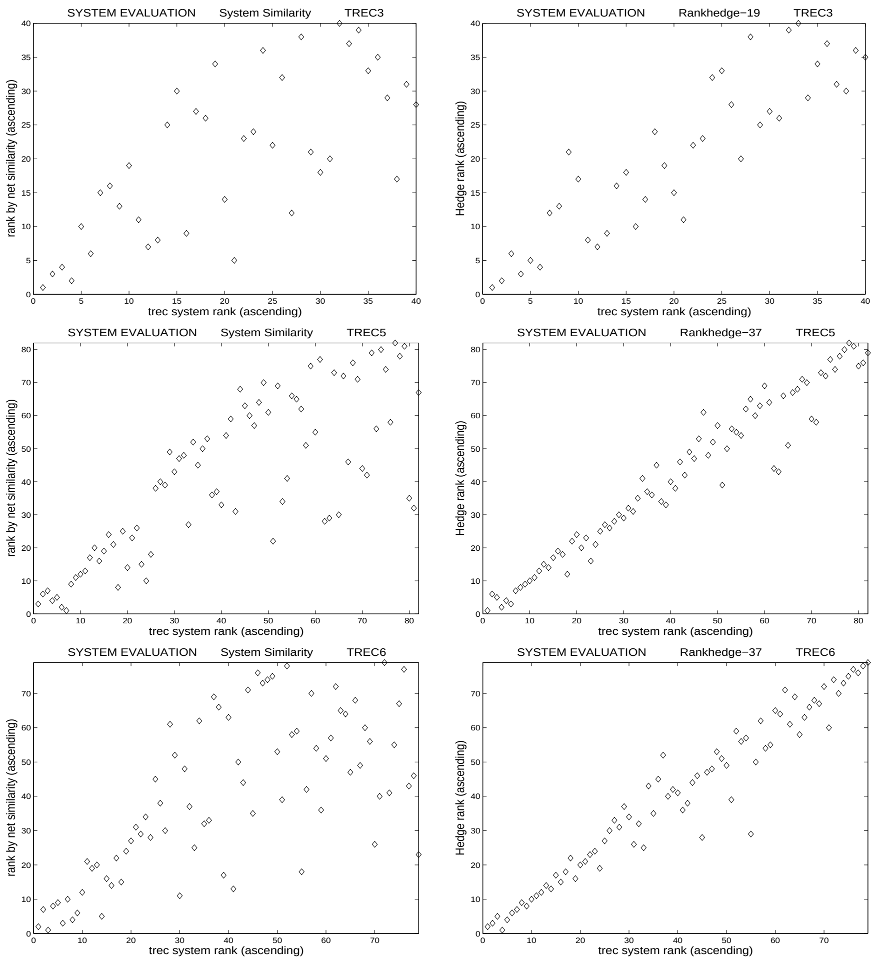


Figure 6.3: TRECs 3, 5, and 6: Column a) System ranking based on net system similarity vs. actual TREC ranks. Column b) Rankhedge-m ranking (pool size m equivalent to Depth-1) vs. actual TREC ranks.

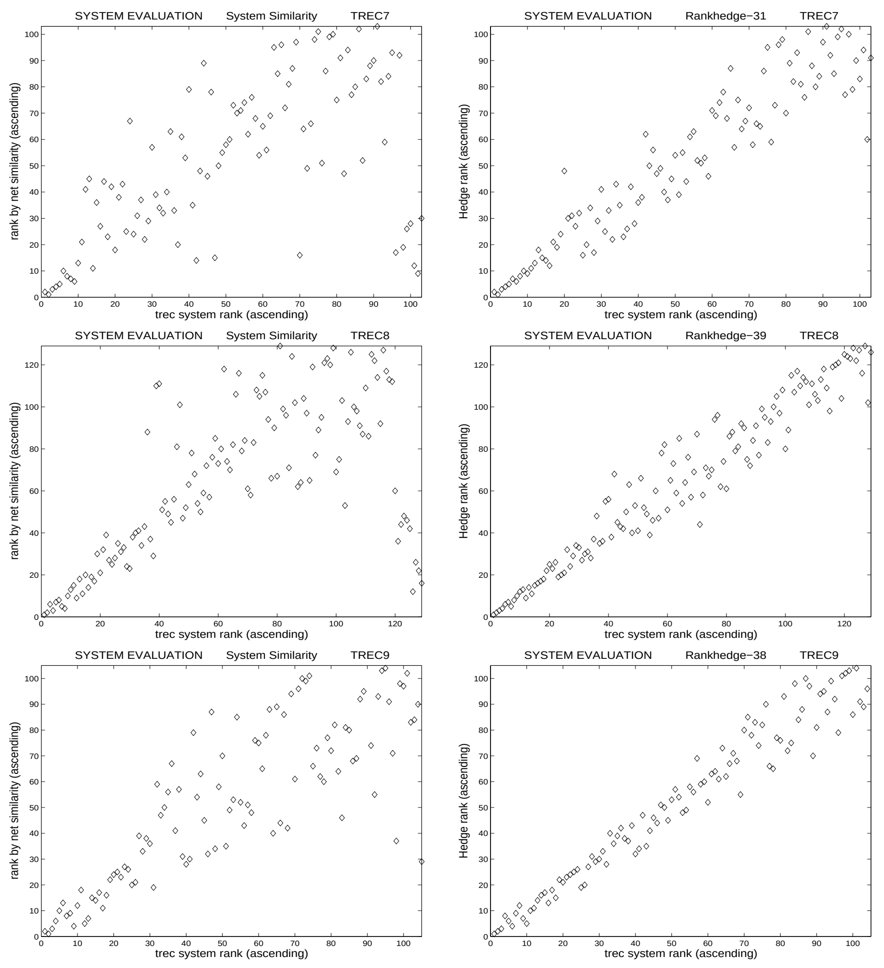


Figure 6.4: TRECs 7, 8, and 9: Column a) System ranking based on net system similarity vs. actual TREC ranks. Column b) Rankhedge-m ranking (pool size m equivalent to Depth-1) vs. actual TREC ranks.

Chapter 7

Active Sampling

In previous chapters, we have focused primarily on manipulations of the weight distribution over predictors via the Hedge algorithm, but to properly address the subject of active learning— the technique of directed (non-uniform) selection of samples to enhance learning rates— we must now fully define the concept of the informational complexity of a sample or an equivalent notion of the log of expected risk on the sample localities.

In Bayesian terms, the evolution of the distribution over predictors in Hedge reflects the iterative introduction of constraints associated with the labelled sample set, and we may view the evolution of Hedge weights as providing an estimate of the posterior accuracy of the individual predictors, given the constraints. The goal of active learning, then, becomes to select samples for labelling in a manner which optimizes convergence of the posterior distribution over predictors to a target distribution reflecting the actual accuracy of the predictors, given complete evidence.

In the following sections, we develop a probabilistic algorithm for selecting samples to enhance learning rates as well as a complementary strategy for maximization of the expected accuracy of the label prediction on the selected sample.

These strategies are based on a measure of the *risk* (expected loss) associated with the field of predictions. Since we may choose to minimize instantaneous error as well as maximize learning rates, we shall refer to the max/min strategy as active sampling rather than active learning.

As suggested by our earlier discussion of the connections of the Hedge algorithm to the dual Markov Random Field and Gibbs Distributions, we begin by demonstrating that a measure of the expected risk of the field restricted to the discrete sample set provides an estimate of the change of volume of the posterior distribution over predictors in response to the sample constraint. We then exploit this duality of expected risk and volume of constraints to define a pair of complementary strategies which effectively split the entropy of the unlabelled field into orthogonal components. By demonstrating that the entropy of the functional partition defined by the strategies is equivalent to the entropy of the uniformly sampled field, we establish the optimality of the probabilistic scheme.

7.1 Background

The question of optimal sample selection for learning has been pursued in diverse practical and theoretical explorations by the computational learning field since the early 90's. Cohn, et al. examine statistically optimal methods for mixture of gaussian models in [22], and Lewis and Gale present an interesting ad-hoc method for training text classifiers based on uncertainty sampling in [58]. Several implementations of committee based algorithms (described in detail below) may be found in [4, 28, 62]. Active learning implementations in the context of SVM's are given in [78, 88].

In an early examination of the prospects of active learning, Eisenberg and Rivest demonstrate a negative result in the PAC learning context. Counterexamples provided in [31] show that, for a natural set of concept classes which they refer to as “dense in themselves” directed queries of the sample set are essentially useless. That is, the learner cannot significantly reduce the total number of labelled samples required to learn a “dense in itself” concept class; because, if the learner observes only a small number of examples, either actively or passively, it cannot be sensitive to slight changes in both the target concept and the underlying distribution. Thus, an adversary can alter the distribution and the target in a way that will not alter the learner’s predicted hypothesis, but which will increase the error of the hypothesis in a significant way.

7.1.1 The Query by Committee Algorithm

Fortunately, the active learning technique proves to be viable given some relaxation of the problem context. Freund and Seung, et al. [80, 38] demonstrate that an active learning technique may improve learning performance, even of the “dense in themselves” concept classes, if it is allowed access to unlabelled as well as labelled examples. These papers introduce an ancestor of our probabilistic active learning technique known as “Query by Committee” or *QBC* which filters a stream of unlabelled samples, selecting samples for labelling which have maximum disagreement as judged by a “committee” of hypotheses randomly selected at each iteration from the surviving members of the concept-class. Examples of practical applications of the Query by Committee method, primarily to the problem of text classification, may be found in [4, 28, 62].

A simple version of the QBC sample selection algorithm may be described as follows:

1. Draw an unlabelled sample from the probability distribution of the sample space.

2. Select two hypotheses at random according to the prior probability distribution of the concept class— restricted to the set of currently consistent

concepts.

3. Select the example for training if the two hypotheses disagree. Else reject the sample and repeat.

The advantage which the QBC algorithm enjoys over the selection method examined by Eisenberg and Rivest may be ascribed to the extra information associated with the availability of predictions on the unlabelled examples. The quantity of disagreement between randomly selected hypotheses provides a crude estimate of the complexity of the field of predictions on sample localities, thus providing an indirect measure of the constraint volume to be expected from an unlabelled sample. Analysis of the QBC algorithm demonstrates that active learning provides improved learning rates even when limited to two-member committees.

Connections to our Hedge based sample selection algorithm are readily identifiable. With our predictors serving as the committee, we should attempt to select a sample at each round which maximizes (or perhaps minimizes) the amount of disagreement among the *weighted* predictions of the committee members. Rather than filtering the input sample stream one sample at a time, however, the probabilistic active sampling technique proposed in this Chapter takes advantage of the availability of predictions on multiple instances and establishes a more general sampling solution, generating an optimal distribution for probabilistic sampling of the space of unlabelled instances.

7.1.2 Semi-supervised Learning

Our probabilistic technique bases its sampling distribution on an estimate of the risk at each sample locality in the unlabelled sample space. A particularly elegant discussion of the utility of expected risk for active learning in a semi-supervised learning context is given in [101], in which Zhu and Lafferty, et al.

develop a method for active learning in the context of Gaussian kernels [53, 54]. While the method of determining the field on the sample set differs significantly from the straightforward labelling of the field via the predictor bundle in the Hedge case, the method for estimating risk, given the vector of field predictions on the samples is equivalent to that in our current context. As in most active learning methodologies, the sample selection process of the algorithm is restricted to the greedy selection of samples, requiring an exhaustive search of the unlabelled sample space to determine the expected reduction in risk on the remaining unlabelled field, given the constraints associated with a particular sample. We shall demonstrate that in the case where estimates of the risk on the unlabelled instance space are available a viable alternative strategy is to sample according to a distribution defined in proportion to the expected risk on these remaining instances.

In contrast to the semi-supervised learning method of [101], which must generate a new set of predictors at each round, the Hedge learning context maintains a fixed set of predictors and modifies the distribution which weights their contribution to predictions. Thus, constraint of the posterior function volume due to labelling of the selected instance serves as the mechanism driving the Hedge update process. In the following, we shall exploit the natural convergence properties of the Hedge method as well as the duality of the expected risk measurement and the convergence of the constrained distribution over predictors to avoid the need for an exhaustive search of the sample space.

7.2 Version Space and Risk

We begin with an examination of the relationship of risk, sample complexity and convergence of the volume of surviving functions in the Bayesian context as defined in an early presentation by Haussler, et al. [45]. The fundamental relationships of these quantities are defined in the lossless or error $\epsilon = 0$ context in which a predictor with zero error is assumed to exist in the hypothesis space. While active learning is not directly addressed, an algorithm for maximization of learning rates based on maximization of risk or sample complexity arises naturally from the framework established in the paper. We shall demonstrate that the Hedge algorithm may be seen as a generalization of the lossless Bayesian on-line algorithm developed in the paper, and we shall, in turn, incorporate the concepts of sample complexity and the posterior volume of surviving functions or *Version Space* into our development of the active sampling method.

7.2.1 Risk

Ideally an active learning technique seeks to minimize the risk to the unlabelled instances due to a particular classification method. Risk may be defined as the estimated generalization error of the classifer, and in the case of finite sample spaces may be computed exactly. In their examination of active learning in the semi-supervised framework, Zhu and Lafferty, et al. [101] provide a concise description of the risk function on the random field, given a Bayesian prediction scheme. This provides an intuitive place to begin our discussion.

Though the active learning method of Zhu, et al. is tailored to the context of Gaussian random fields, the estimated risk measure $\widehat{\mathcal{R}}(f)$, as presented in the paper, is equally applicable to the Hedge context. Let $\mathcal{F}$ be a family of functions defined on an instance space X. We may define the actual risk $\mathcal{R}_{\text{Bayes}}(f)$ of a Bayes classifier in terms of the mean prediction or "field" on the unlabelled portion of the sample set $x_1 \cdots x_n$ with associated labels $y_i \in \{0,1\}$ as follows:

$$\mathcal{R}_{\text{Bayes}}(f) = \sum_{u \in \mathbf{U}} \sum_{\hat{y} \in \{0,1\}} \sum_{f_u \in \mathcal{F}} [\text{sgn}(f_u) \neq \hat{y}] p^*(y_u = \hat{y} | \mathbf{L})] \tag{7.1}$$

$$\tag{7.2}$$

Here, the set of labelled instances is represented by $\mathbf{L} = x_1 \cdots x_\ell$. The unlabelled instances likewise correspond to $\mathbf{U} = x_{\ell+1} \cdots x_n$. As described in Chapter Five, the Bayesian decision rule is denoted $\text{sgn}(f_u)$ and is defined as $\text{sgn}(f_u) = 1$ for $f_u > 0.5$ and $\text{sgn}(f_u) = 0$ for $f_u < 0.5$, with $\text{sgn}(f_u) \in \{0, 1\}$ selected randomly with equal probability for $\text{sgn}(f_u) = 0.5$. The posterior distribution on the labels— $p^*(y_u|\mathbf{L})$— corresponds to the unknown *true* label distribution at node u, given the labelled data. To compute the *estimated risk* we can approximate the unknown distribution $p^*(y_u|\mathbf{L})$ using the mean of the predictions on x_u:

$$p^*(y_u = 1|\mathbf{L}) \approx \bar{f}_u \tag{7.3}$$

so that the estimated risk $\widehat{\mathcal{R}}_{\text{Bayes}}(f)$ may be written:

$$\begin{aligned}
\widehat{\mathcal{R}}_{Bayes}(f) &= \sum_{u \in \mathbf{U}} \sum_{\hat{y} \in \{0,1\}} [\text{sgn}(\bar{f}_u) \neq \hat{y}] \cdot p^*(y_u = \hat{y}|\mathbf{L}) \\
&= \sum_{u \in \mathbf{U}} \big([\text{sgn}(\bar{f}_u) \neq 0](1 - \bar{f}_u) + [\text{sgn}(\bar{f}_u) \neq 1] \cdot \bar{f}_u \big) \\
&= \sum_{u \in \mathbf{U}} \min(\bar{f}_u, 1 - \bar{f}_u) \qquad (7.4)
\end{aligned}$$

Likewise, we may develop an estimate for the risk of a Gibbs classifier, which probabilistically assigns label $\hat{y}_u \in \{0, 1\}$ in proportions defined by the mean field $\bar{f}_u$ on the sample. This results in an estimated risk:

$$\begin{aligned}
\widehat{\mathcal{R}}_{\text{Gibbs}}(f) &= \sum_{u \in \mathbf{U}} \sum_{\hat{y} \in \{0,1\}} (1 - p^*(y_u = \hat{y}|\mathbf{L})) \cdot p^*(y_u = \hat{y}|\mathbf{L}) \\
&= \sum_{u \in \mathbf{U}} \bar{f}_u(1 - \bar{f}_u) + (1 - \bar{f}_u)\bar{f}_u \\
&= \sum_{u \in \mathbf{U}} 2 \cdot \bar{f}_u(1 - \bar{f}_u) \qquad (7.5)
\end{aligned}$$

Exhaustive search of the unlabelled sample space for the instance whose inclusion in the labelled set results in minimum expected risk on the remaining unlabelled samples guarantees a maximum expected learning rate. However, in the Hedge context, it is possible to exploit the natural convergence properties of the algorithm to develop a more efficient probabilistic method for active selection. The Hedge learning process may be viewed as an exponential winnowing of the volume of inconsistent functions from the space of possible labellings of the instances, and the direct relationship of risk and constraint volumes associated with sample localities enables the development of an optimal probabilistic sampling strategy in the Hedge context.

7.2.2 Sample Complexity

Sample complexity corresponds to the entropy of the field of predictions defined on the individual samples, and it is naturally defined in terms of the expected log volume of the posterior distribution on the predictor space conditioned by the labelled sample constraints. The duality of sample complexity and the log volume of the surviving function bundle, as developed by Haussler, et al. in [45], may be exploited to develop an optimal on-line learning algorithm for the lossless case in which a predictor with zero loss is guaranteed to exist.

In the following discussion, we first establish the definition of sample complexity as described in [45] and its relationship to the expected error rates of Bayesian and Gibbsian prediction algorithms. This leads to a definition of expected risk in terms of the volume of surviving functions or Version Space in the lossless context of the Haussler paper, in which a function of error $\epsilon = 0$ relative to the labelled sequence is always assumed to exist. By generalizing the concept of the Version Space to allow for lossy situations in which the minimum function error may be $\epsilon \geq 0$, a lossy definition of sample complexity and Version Space may be applied in the context of the Hedge algorithm, and the appropriate active sampling algorithms for maximization or minimization of risk proceed naturally.

7.3 Error Bounds for On-line Algorithms

The Bayesian on-line learning process may be viewed as a resource allocation scheme which weights functions in a family of predictors in proportion to their posterior probabilities, given the evidence of a sample vector $\mathbf{x}$ drawn from sample space X. In this context, the *Version Space* corresponds to the posterior volume of functions consistent with the evidence of $\mathbf{x}$, and its associated informational *volume* may be defined.

Let $\mathcal{F}$ be a family of functions defined on an instance space X. We shall define $\mathcal{P}$ to be the prior probability distribution over $\mathcal{F}$ and $\mathcal{P}_m$ to be the mth posterior distribution given the evidence vector $x_1, \cdots x_m$, which restricts $\mathcal{P}$ to the mth Version Space $\mathcal{F}_m(f)$. $\mathcal{P}_m$ can be interpreted as the subjective probability distribution over various target concepts, given labels $f(x_1), \ldots, f(x_m)$ on the first m instances. Given a possibly infinite sequence of instances $\mathbf{x} = x_1, \ldots, x_m, x_{m+1}, \ldots$, the mth *volume* is defined as $\mathcal{V}_m^{\mathcal{P}}(\mathbf{x}, f)$, which is the probability volume of functions of $\mathcal{F}$ consistent with the first m samples. That is, $\mathcal{V}_m^{\mathcal{P}}(\mathbf{x}, f) = \mathcal{P}[\mathcal{F}_m(\mathbf{x}, f)]$, with

$$\mathcal{F}_m(\mathbf{x}, f) = \{\hat{f} \in \mathcal{F} : \hat{f}(x_1) = f(x_1), \ldots, \hat{f}(x_m) = f(x_m)\}.$$

Thus, the concept of a version space is defined in [45] in a lossless or exact context as the proportion of the distribution $\mathcal{P}$ over the function space which is exactly ($\epsilon = 0$) consistent with the evidence vector $\hat{f}(x_1, \ldots, x_m) = \hat{y}_1, \ldots, \hat{y}_m$.

To illustrate the utility of the concept of function volume, the *instantaneous* *information gain* of the $m+1$st label may be derived as follows:

$$\begin{aligned} \mathcal{I}^{\mathcal{P}}_{m+1}(\mathbf{x}, f) &= \mathcal{I}_{m+1}(f) & (7.6)\\ &= -\log \mathbf{E}_{\hat{f}\in\mathcal{P}_m}[\hat{f}(x_{m+1} = f(x_{m+1})|\hat{f}(x_i) = f(x_i), 1 \le i \le m] & (7.7)\\ &= -\log \frac{\mathcal{V}_{m+1}(f)}{\mathcal{V}_m(f)} & (7.8)\\ &= -\log \mathcal{X}_{m+1}(f) & (7.9) \end{aligned}$$

where the $m+1$st *volume ratio* is given by:

$$\mathcal{X}^{\mathcal{P}}_{m+1}(\mathbf{x}, f) = \mathcal{X}_{m+1}(f) = \mathcal{V}_{m+1}(f)/\mathcal{V}_m(f)$$

In addition, we may derive instantaneous expected losses for Bayes and Gibbs algorithms.

As discussed in Chapter Five, the Bayes classifier is a thresholded majority classifier. Therefore, since the volume $\mathcal{V}_{m+1}(f)$ of surviving functions corresponds to those elements in the function space whose predictions were consistent with the target function on instance x_m, the accuracy of the Bayes classifier may be defined in terms of the relationship of consecutive volumes $\mathcal{V}_m(f)$ and $\mathcal{V}_{m+1}(f)$. A mistake in predicting $f(x_{m+1})$ is made with probability 1 if $\mathcal{V}_{m+1}(f) < \frac{1}{2}\mathcal{V}_m(f)$, with probability $\frac{1}{2}$ if $\mathcal{V}_{m+1}(f) = \frac{1}{2}\mathcal{V}_m(f)$ and with probability 0 otherwise. Thus, the Bayes mistake probability on $f(x_{m+1})$ for fixed $\mathbf{x}$, f and $\mathcal{P}$ is given by:

$$\text{Bayes}^{\mathcal{P}}_{m+1}(\mathbf{x}, f) = \text{Bayes}_{m+1}(f) = \Theta\Big(\mathcal{X}_{m+1}(f)\Big)$$

where $\Theta(x) = 1$ if $x > \frac{1}{2}$, $\Theta(0) = \frac{1}{2}$, and $\Theta(x) = 0$ otherwise.

For the Gibbs algorithm, the prediction $f(x_{m+1})$ is accurate iff the randomly chosen hypothesis $\hat{f}$ is in $\mathcal{F}_{m+1}(f)$. Since $\mathcal{F}$ is chosen randomly according to $\mathcal{P}_m$, and the probability of $\mathcal{F}_{m+1}(f)$ under $\mathcal{P}_m$ is exactly $\mathcal{V}_{m+1}(f)/\mathcal{V}_m(f) = \mathcal{X}_{m+1}(f)$, the probability that $f(x_{m+1})$ is predicted incorrectly is:

$$\mathrm{Gibbs}^{\mathcal{P}}_{m+1}(\mathbf{x}, f) = \mathrm{Gibbs}_{m+1}(f) = 1 - \mathcal{X}_{m+1}(f).$$

for fixed $\mathbf{x}$, f and $\mathcal{P}$.

7.3.1 Expected Risk and Uncertainty

The risks of the Bayes and Gibbs algorithms on an arbitrary unlabelled sample correspond to the expected error of the predictive scheme when the true label is drawn in accordance with the posterior probability over predictors. Equivalence of the expected prediction and the volume of functions consistent with a prediction allows us to develop an alternate definition of the instantaneous risk of the Bayes and Gibbs algorithms in terms of the volume ratio.

As demonstrated in [45], the quantities of Bayes and Gibbs risk as well as the entropy of the field on a sample may be generalized by defining the expectations in terms of an arbitrary real-valued function of one argument. Haussler, et al. demonstrate that for any real valued function $\mathcal{G}(p)$, the expectation $\mathbf{E}_{f\in\mathcal{P}}[\mathcal{G}(\mathcal{X}_{m+1}(f))]$ is equivalent to:

$$\begin{aligned}\mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{G}(\mathcal{X}_{m+1}(f))\big] &= \mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{X}_{m+1}(f)\mathcal{G}(\mathcal{X}_{m+1}(f)) \\ &\quad + (1-\mathcal{X}_{m+1}(f))\mathcal{G}(1-\mathcal{X}_{m+1}(f))\big] \qquad (7.10)\end{aligned}$$

We are interested in the three forms of $\mathcal{G}$ we have been considering, $\mathcal{G}(p) = \Theta\big(\frac{1}{2} - p\big)$ for the Bayes predictor, $\mathcal{G}(p) = 1 - p$ for the Gibbs predictor, and $\mathcal{G}(p) = -\log p$ for the information gain. Substituting into Equation 7.10 we derive representations for expected risk and uncertainty.

The risk of the Bayesian predictor on sample x_{m+1} may be written:

$$\begin{aligned}
\widehat{\mathcal{R}}_{\text{Bayes}}(x_{m+1}) &= \mathbf{E}_{f\in\mathcal{P}}\big[\Theta\big(\frac{1}{2}-\mathcal{X}_{m+1}(f)\big)\big] && (7.11)\\
&= \mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{X}_{m+1}(f)\Theta\big(\frac{1}{2}-\mathcal{X}_{m+1}(f)\big) && (7.12)\\
&\qquad + (1-\mathcal{X}_{m+1}(f))\Theta\big(\frac{1}{2}-\mathcal{X}_{m+1}(f)\big)\big] && (7.13)\\
&= \mathbf{E}_{f\in\mathcal{P}}[\min(\mathcal{X}_{m+1}(f),1-\mathcal{X}_{m+1}(f))]. && (7.14)
\end{aligned}$$

The risk associated with the Gibbs predictor on x_{m+1} is:

$$\begin{aligned}
\widehat{\mathcal{R}}_{\text{Gibbs}}(x_{m+1}) &= \mathbf{E}_{f\in\mathcal{P}}\big[1-\mathcal{X}_{m+1}(f)\big] && (7.15)\\
&= \mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{X}_{m+1}(f)\cdot(1-\mathcal{X}_{m+1}(f)) && (7.16)\\
&\qquad + (1-\mathcal{X}_{m+1}(f))\cdot\mathcal{X}_{m+1}(f)] && (7.17)\\
&= \mathbf{E}_{f\in\mathcal{P}}[2\cdot\mathcal{X}_{m+1}(f)\cdot(1-\mathcal{X}_{m+1}(f))] && (7.18)
\end{aligned}$$

And the expected information gain due to x_{m+1} is given by:

$$\begin{aligned}
\mathbf{E}_{f\in\mathcal{P}}[\mathcal{I}_{m+1}(f)] &= \mathbf{E}_{f\in\mathcal{P}}[-\log\mathcal{X}_{m+1}(f)] && (7.19)\\
&= \mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{X}_{m+1}(f)\log(\mathcal{X}_{m+1}(f)) && (7.20)\\
&\qquad + (1-\mathcal{X}_{m+1}(f))\log(1-\mathcal{X}_{m+1}(f))\big] && (7.21)\\
&= \mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{H}_{\text{bin}}(\mathcal{X}_{m+1}(f))\big]. && (7.22)
\end{aligned}$$

where $\mathcal{H}_{\text{bin}}$ is the familiar binary entropy function.

Symmetry of the terms inside the expectation relative to the set of binary predictions allows us to move the expectation inside the brackets, replacing $\mathcal{X}_{m+1}(f)$ with its expected value, $\bar{f}_{m+1} = \mathbf{E}_{f\in\mathcal{P}}[\mathcal{X}_{m+1}(f)]$. This yields an estimated risk for Bayes and Gibbs algorithms consistent with the earlier derivations of Equations 7.4 and 7.5:

$$\widehat{\mathcal{R}}_{Bayes}(x_{m+1}) \quad = \quad \min(\bar{f}_{m+1}, 1-\bar{f}_{m+1}), \tag{7.23}$$

and

$$\widehat{\mathcal{R}}_{\text{Gibbs}}(x_{m+1}) \quad = \quad 2 \cdot \bar{f}_{m+1} \cdot (1-\bar{f}_{m+1}). \tag{7.24}$$

Likewise, we may define our expected uncertainty in terms of $\bar{f}$:

$$\begin{aligned} \mathbf{E}_{f\in\mathcal{P}}[\mathcal{I}_{m+1}(f)] &= -\bar{f}_{m+1}\log(\bar{f}_{m+1}) - (1-\bar{f}_{m+1})\log(1-\bar{f}_{m+1}) && (7.25) \\ &= \mathcal{H}_{\text{bin}}(\bar{f}_{m+1}) && (7.26) \end{aligned}$$

Finally, it is easy to verify that, for any $p \in [0,1]$, the following inequality holds:

$$\min(p, 1-p) \le 2p(1-p) \le \frac{1}{2}\mathcal{H}_{\text{bin}}(p),$$

so that:

$$\widehat{\mathcal{R}}_{\text{Bayes}}(x_{m+1}) \le \widehat{\mathcal{R}}_{\text{Gibbs}}(x_{m+1}) \le \frac{1}{2}\mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{H}_{\text{bin}}(\mathcal{X}_{m+1}(f))\big] = \frac{1}{2}\mathbf{E}_{f\in\mathcal{P}}\big[\mathcal{I}_{m+1}(f)\big]$$

Thus, the expected information gain (or entropy reduction) associated with selection of the new instance bounds both Bayes and Gibbs risks.

7.3.2 Cumulative Upper Bounds

We may also establish cumulative bounds on the number of mistakes incurred by the on-line strategies. While the expressions for the cumulative mistakes by the Bayes and Gibbs algorithms are difficult to analyze due to lack of a closed form expression, we may develop a bound based on the cumulative information gain in m trials. Invoking additivity of information and recalling that $V_0(f) = 1$:

$$\begin{aligned}
\mathbf{E}_{f\in\mathcal{P}}\big[\sum_{i=1}^{m}\mathcal{I}_i(f)\big] &= \mathbf{E}_{f\in\mathcal{P}}\big[\sum_{i=1}^{m}-\log\mathcal{X}_i(f)\big] \\
&= \mathbf{E}_{f\in\mathcal{P}}[\sum_{i=1}^{m}\big(\log\mathcal{V}_{i-1}(f)-(\log\mathcal{V}_i(f)\big)] \\
&= \mathbf{E}_{f\in\mathcal{P}}\big[-\log\mathcal{V}_m(f)\big] \qquad (7.27)
\end{aligned}$$

Again, following the Haussler argument, if we consider the weighting scheme $P_m(\mathbf{x}, f)$ over the function space $\mathcal{F}$ at trial m to represent a probabilistic labelling scheme with function values f_m assigned to each of two labels $\{0, 1\}$ in proportion to the weight p_f^m, then the first m instances $x_1, \ldots, x_m$ define a partition of the function space into equivalence classes on the space of functional configurations on $\mathbf{x}_m$: $\prod_m^{\mathcal{F}}(\mathbf{x}) = \prod_m^{\mathcal{F}} = \{\mathcal{F}_m(\mathbf{x}, f) : f \in \mathcal{F}\}$, with the entropy of the partition naturally defined in terms of prior $\mathcal{P}$ as:

$$\mathcal{H}_{\text{bin}}^{\mathcal{P}}(\mathbf{x}_m) = \mathbf{E}_{f\in\mathcal{P}}\big[-\log\mathcal{V}_m(f)\big] = -\sum_{\pi\in\prod_m^{\mathcal{F}}}\mathcal{P}[\pi]\log\mathcal{P}[\pi] \qquad (7.28)$$

Thus, the expected cumulative information gain from the labels $\mathbf{x}_m = x_1, \ldots, x_m$ is simply the entropy of partition $\prod_m^{\mathcal{F}}$ under $\mathcal{P}$.

This analysis leads to straightforward bounds on the expected cumulative loss of the Gibbs and Bayes algorithms:

$$\begin{aligned}
\mathbf{E}_{f\in\mathcal{P}}[\sum_{i=1}^{m}\text{Bayes}_i(f)] &\leq \mathbf{E}_{f\in\mathcal{P}}[\sum_{i=1}^{m}\text{Gibbs}_i(f)] \qquad (7.29) \\
&\leq \frac{1}{2}\mathbf{E}_{f\in\mathcal{P}}[-\log V_m(f)] \qquad (7.30) \\
&= \frac{1}{2}\mathcal{H}_{\text{bin}}^{\mathcal{P}}(\mathbf{x}_m) \qquad (7.31)
\end{aligned}$$

7.3.3 Hedge and the Version Space

Next, we would like to examine the more general context in which the maximum error rate of the best predictor may be greater than zero. We shall refer to this as the lossy context. A cursory examination of the Hedge update rule $w_i^{m+1} = w_i^m \cdot \beta^{\ell_i^m}$ demonstrates that it may be viewed as a generalization of the Bayes or Gibbs on-line learning algorithms (depending on our choice of thresholded or probabilistic classification method) to the lossy context. Conversely, the lossless on-line algorithms of the previous section may be viewed as instances of the Hedge algorithm with β restricted to 0 to maximally punish functions for incorrect predictions.

The Hedge algorithm may be readily incorporated into the sample complexity framework of [45] by establishing a bound on the instantaneous loss of Hedge in terms of the volume of the distribution over predictors:

$$\mathcal{X}_{m+1} = \frac{\mathcal{V}_{m+1}(f)}{\mathcal{V}_m(f)} = \frac{\sum_{i=1}^N w_i^{m+1}}{\sum_{i=1}^N w_i^m}.$$

Recalling the following inequality from our Hedge discussion:

$$\alpha^r \leq 1 - (1 - \alpha)r \tag{7.32}$$

with $\alpha \geq 0$ and $r \in [0, 1]$, we may interpret the distribution of weights over the predictor space as an estimate of the posterior accuracy of the individual predictors and derive an upper bound for the instantaneous loss due to evidence of item x_m in a manner consistent with the lossless analysis of the previous sections. For an arbitrary β and with total predictors N and distribution over predictors at round m, $\mathbf{p}^m$, we may establish the following relationship between the Hedge loss ℓ_i^m assessed to predictor i at round m and the function volume at m and $m+1$:

$$\sum_{i=1}^{N} w_i^{m+1} = \sum_{i=1}^{N} w_i^m \beta^{\ell_i^m} \leq \sum_{i=1}^{N} w_i^m (1-(1-\beta)\ell_i^m) = \big(\sum_{i=1}^{N} w_i^m\big)(1-(1-\beta)\mathbf{p}^m \cdot \ell^m) \tag{7.33}$$

with $\mathbf{p}^m = \mathbf{w}^m / \sum_{i=1}^N w_i^m$.

This leads to a description of the worst case Hedge loss or risk for arbitrary β in terms of the volume ratio:

$$\begin{aligned}
L^m_{\text{Hedge}_\beta}(f) &= \sum_{i=1}^{N} \mathbf{p}^m \cdot \ell^m && (7.34)\\
&\leq \frac{\sum_{i=1}^N w_i^m - \sum_{i=1}^N w_i^{m+1}}{(1-\beta)\cdot \sum_{i=1}^N w_i^m} && (7.35)\\
&= \frac{\mathcal{V}_m(f) - \mathcal{V}_{m+1}(f)}{(1-\beta)\cdot \mathcal{V}_m(f)} && (7.36)\\
&= \frac{1-\mathcal{X}_{m+1}(f)}{1-\beta} && (7.37)\\
&= \frac{\text{Gibbs}^{\mathcal{P}}_{m+1}(\mathbf{x}, f)}{1-\beta} && (7.38)
\end{aligned}$$

We may also establish bounds on expected risk of the Hedge algorithm for arbitrary β as in the earlier lossless examples, but now with expectations taken over the non-uniform distribution defined by the Hedge weights on the predictor set, rather than, as in the lossless case, from the uniformly weighted volume of surviving functions:

$$\widehat{\mathcal{R}}_{\text{Hedge}_\beta}(x_m) \leq \mathbf{E}_{f\sim\mathcal{P}_m}[\frac{1-\mathcal{X}_{m+1}(f)}{1-\beta}] \leq \mathbf{E}_{f\sim\mathcal{P}_m}[-\frac{\log \mathcal{X}_{m+1}(f)}{1-\beta}] \tag{7.39}$$

or in terms of the expected field $\bar{f}_m$ on sample x_m:

$$\widehat{\mathcal{R}}_{\text{Hedge}_\beta}(x_m) \leq \frac{\widehat{\mathcal{R}}_{\text{Gibbs}}(x_m)}{1-\beta} = \frac{2\bar{f}_m(1-\bar{f}_m)}{1-\beta} \leq \frac{1}{2}\frac{\mathcal{H}_{\text{bin}}(\bar{f}_m)}{\big(1-\beta\big)} \tag{7.40}$$

Finally, for cumulative expected loss, we have:

$$L^m_{\text{Hedge}_\beta}(f) \leq \frac{1}{2}\mathbf{E}_{f\sim\mathcal{P}_0}[-\frac{\log V_{m+1}(f)}{1-\beta}] = \frac{1}{2}\frac{\mathcal{H}^{\mathcal{P}_0}_{\text{bin}}(\mathbf{x}_{m+1})}{\big(1-\beta\big)} \tag{7.41}$$

corresponding to the expectation taken over the prior distribution over predictors of our usual worst case loss bound for Hedge, with $V_{m+1}(f) = \sum_{i=1..N} w_i^{m+1}$.

7.4 Active Sampling

The duality of the expected risk on an unlabelled sample and the expected diminution of the posterior distribution due to labelling of the sample may be exploited to define a probabilistic method for active sample selection. When the goal is to enhance learning, the technique seeks to minimize the estimated risk on the unlabelled sample set by employing a probabilistic technique to maximize expected risk of the selected sample. Equivalently, the technique attempts to maximize the rate of convergence of the posterior volume to the optimal distribution based on labelling of all constraints.

In most cases the goal of an active sampling method is to select a sample sequence which has an enhanced learning rate relative to a uniform sampling of the unlabelled instances. However, we shall demonstrate that a sampling strategy seeking to maximize the probability of accurately labelling the selected sample (i.e. selecting the sample with minimum risk) is equally viable, and that the two strategies naturally split the entropy of the unlabelled sample field into orthogonal components. These orthogonal components correspond to a max/min strategy for sampling the space and are properly addressed within a single unified information-theoretic framework.

We shall develop the technique of active sampling in the lossless context using the terms for expected risk and entropy defined in the previous sections. The sampling techniques apply without modification to the lossy situation, since the upper bounds of the Gibbs and Hedge techniques differ only by a constant factor.

7.4.1 A Probabilistic Method

In order to maximize the expected convergence rate of the posterior volume per sample or alternatively to minimize the immediate expected risk on the selected sample, we shall define a probabilistic sampling method based on the proportional resource allocation methods examined in Chapter Five. Our method may be contrasted with typical greedy methods of sample selection such as in [75, 101]. In these methods, a set of samples is drawn from the sample space and an exhaustive examination of the expected effect of labelling of each sample on the posterior risk over remaining samples is conducted. The algorithm then greedily selects the sample whose label is expected to result in the greatest dimunition in risk over the remaining samples. This method tends to avoid some of the worst case behavior of other active learning techniques (such as the susceptibility to risky but irrelevant samples), by directly measuring the expected risk on the remaining sample field. Our probabilistic algorithm is designed to limit exposure to worst case behavior without an exhaustive examination of the risk of the sample field, by applying hedge principles on the sample axis.

As in our earlier discussion, we denote the labelled and unlabelled sample sets as $\mathbf{L} = \{x_1, \ldots, x_\ell\}$ and $\mathbf{U} = \{x_{\ell+1}, \ldots, x_N\}$, respectively. The probabilistic method for active learning draws samples from a distribution over the unlabelled sample set based on the expected risk under a Gibbsian prediction strategy, so that:

$$\mathbf{Pr}(x_{m+1} = x_u \in \mathbf{U}) = \widehat{\mathcal{R}}_{\text{Gibbs}}(x_u)/Z^- \tag{7.42}$$

with $Z^- = \sum_{x_u \in \mathbf{U}} \widehat{\mathcal{R}}_{\text{Gibbs}}(x_u) = \sum_{x_u \in \mathbf{U}} 2\bar{f}_u(1 - \bar{f}_u)$

Alternatively, we might sample at a rate intended to maximize the instantaneous expected return or probability of success of the labelled sample:

$$\mathbf{Pr}(x_{m+1} = x_u \in \mathbf{U}) = (1 - \widehat{\mathcal{R}}_{\text{Gibbs}}(x_u))/Z^+ \tag{7.43}$$

with $Z^+ = \sum_{x_u \in \mathbf{U}} (1 - \widehat{\mathcal{R}}_{\text{Gibbs}}(x_u)) = \sum_{x_u \in \mathbf{U}} \bar{f}_u^2 + (1 - \bar{f}_u)^2$.

As noted in Equation 7.15, these sampling distributions have an equivalent interpretation in terms of the expected volume of the respective posterior function bundles. With $\mathcal{X}_u^{\pm}$ corresponding to the volume ratios of the function bundles through $u \in \mathbf{U}$ which are respectively consistent or inconsistent with the predictions of the bundle, and $\mathcal{P}_\ell$ the distribution over predictors after labelling the set of instances $x_{1..\ell}$, the expected volume ratios associated with predicted risk and gain on sample u are, by definition: $\mathbf{E}_{f \sim \mathcal{P}_\ell}[\mathcal{X}_u^-(f)] = \widehat{\mathcal{R}}_{\text{Gibbs}}(x_u)$ and $\mathbf{E}_{f \sim \mathcal{P}_\ell}[\mathcal{X}_u^+(f)] = (1 - \widehat{\mathcal{R}}_{\text{Gibbs}}(x_u))$.

The Joint Predictor/Label Distribution

The probabilistic sampling strategy allows us to allocate sampling resources optimally on both the predictor and the sample axis. Introduction of the extra degree of freedom (associated with resource allocation on the sample axis) allows us to optimally partition the posterior function bundle on the unlabelled sample set into orthogonal components— with one of these components corresponding to the portion of the predictor space consistent with the mean field assignment on the samples and the complementary component corresponding to the expected error bundle given the mean field assignment.

We shall examine the sampling scheme on the unlabelled samples in terms of the joint distribution of the expected labels (the expected field) and the probabilistic predictions of the individual hypotheses. In the Hedge context, functions are restricted to a finite function bundle $\mathbf{F} \in \mathcal{F}$. We introduce the random variables $\bar{h}_x = h(x)$, corresponding to the weights attributed to labels

$\{0, 1\}$ by unweighted hypothesis $h \in \mathbf{F}$, and $\bar{f}_x = \mathbf{E}_{f \sim \mathcal{P}_\ell}\big[f(x)\big]$, corresponding to the respective weights of labels $\{0, 1\}$ in the field defined by $f \in \mathbf{F}$ at x. As in previous sections, $\mathcal{P}^\ell$ refers to the posterior distribution over predictors $f \in \mathbf{F}$ after labelling of the ℓth (last in the sequence) sample in the labelled set $\mathbf{L}$.

Let $\mathbf{K}$ be a matrix of dimension $|\mathbf{F}| \times |\mathbf{U}| \times 2 \times 2 \mapsto [0, 1]$, such that $\mathbf{K}_{(h,u)}$ references the joint distribution of the unweighted prediction $\bar{h}_u$ and expected field $\bar{f}_u$ on $x_u \in \mathbf{U}$:

$$\mathbf{K}_{(h,u)} = \mathbf{E}_{f \sim \mathcal{P}^\ell}\big[h(x_u) = \{0, 1\}, f(x_u) = \{0, 1\}\big] \tag{7.44}$$

$$= \bar{h}_u \times \bar{f}_u \tag{7.45}$$

An example distribution, shown in Table 7.1, reflects a hypothetical positive prediction $\bar{h}_u = 1$ with probability 1/4 and a mean field prediction $\bar{f}_u = 1$ with probability 5/8.

Table 7.1: Joint distribution of hypotheses $\bar{h}_x = 1/4$ and field $\bar{f}_x = 5/8$

$\bar{h}$, $\bar{f}$	0	1
0	9/32	15/32
1	3/32	5/32

A second matrix $\mathbf{M}$ will serve as the actual resource allocation matrix and corresponds to the joint probability matrix $\mathbf{K}$ weighted by the posterior distribution over hypotheses. With $\mathbf{w}^\ell$ the posterior distribution over hypotheses after constraint by instance ℓ, the entries of $\mathbf{M}$ are given by $\mathbf{M}_{(u,h)} = \mathbf{w}^\ell_h \mathbf{K}_{(u,h)}$. We shall refer to the sum of entries of the subspaces of the matrix $\mathbf{M}$ or $\mathbf{K}$ corresponding to the case in which the expected label is consistent with the hypothesis h (the sum of diagonals of a cell $\mathbf{M}_{(u,h)}$) as $\mathbf{M}^+$ and $\mathbf{K}^+$, respectively. Likewise the case in which the expected label is inconsistent with h (the sum of skew terms) is denoted $\mathbf{M}^-$ and $\mathbf{K}^-$.

In matrix form, the expected volume ratio for an arbitrary sample column u after labelling of the ℓth instance is given by:

$$\mathbf{M}^{+}_{u} = \mathbf{E}_{f\sim\mathcal{P}^{\ell}}[\mathcal{X}^{+}_{u}(f)] = \mathbf{E}_{h\sim\mathcal{P}^{\ell},f\sim\mathcal{P}_{\ell}}\big[h(x_u) = f(x_u)\big]$$

And an equivalent formulation for the volume ratio associated with the inconsistent samples is:

$$\mathbf{M}^{-}_{u} = \mathbf{E}_{f\sim\mathcal{P}^{\ell}}[\mathcal{X}^{-}_{u}(f)] = \mathbf{E}_{h\sim\mathcal{P}^{\ell},f\sim\mathcal{P}^{\ell}}\big[h(x_u) \neq f(x_u)\big].$$

We wish to define a pair of weighting strategies on the unlabelled sample set to maximize either the expected gain or risk of the selected sample. Consistent with the discussions of Chapter Five concerning optimality of proportional gambling strategies, we specify a probabilistic scheme which samples the unlabelled instances at a rate proportional to the expected gain or risk of the samples. The respective sampling vectors $\mathbf{m}^{\pm}_{u}$ are given by:

$$\mathbf{m}^{\pm}_{u} = \frac{\sum_{h\in\mathbf{F}}\mathbf{M}^{\pm}_{(h,u)}}{Z^{\pm} = \sum_{u\in\mathbf{U},h\in\mathbf{F}}\mathbf{M}^{\pm}_{(h,u)}}$$

Optimal Sampling on the Predictor Axis

The probabilistic sampling scheme defined in two dimensions apportions the sampling space to individual predictions $h(x_u)$ at a rate proportional to the joint posterior probability of gain or risk associated with the prediction on the sample, and arguments for optimality of proportional resource allocation presented in Chapter Five apply without modification to either dimension of the matrix $\mathbf{M}$. That is, if we choose to aggregate the sampling resources along the hypothesis dimension rather than the sample dimension, we find that the sampling method allocates resources to an arbitrary h in proportion to the expectation that the hypothesis is consistent with either the target $(\hat{f})$ or error

($f_{\text{err}} = 1 - \hat{f}$), depending on our choice of sampling strategy. The probabilistic strategy is, therefore, an optimal sampling scheme in terms of both the sample and predictor axes.

For either the strategy maximizing risk or reward, we make the assumption that the distribution over hypotheses $\mathbf{w}^{\ell}$ is a current best approximation of the true posterior distribution: $\tilde{P}(h|\mathbf{L}) \approx w_h^{\ell}$. This leads to an expression, consistent with Bayes rule, for the expected posterior probability of hypothesis $\hat{h}$ given labelled set $\mathbf{L}$ and an arbitrary unlabelled sample u:

$$\tilde{P}(\hat{h}|\mathbf{L} + u) \approx \frac{\tilde{P}(\hat{h}|\mathbf{L}) \cdot \mathbf{K}^{\pm}_{(\hat{h},u)}}{\sum_{h \in \mathbf{F}} \tilde{P}(h|\mathbf{L}) \cdot \mathbf{K}^{\pm}_{(h,u)}} \tag{7.46}$$

Summing over the unlabelled samples, the quantity of the resource allocation vector associated with hypothesis h corresponds to:

$$\mathbf{m}^{\pm}_{h} \propto \mathbf{E}_{u \in \mathbf{U}}[\tilde{P}(h|\mathbf{L} + u)] \tag{7.47}$$

Entropy of the Field on Unlabelled Samples

As in the earlier sections on proportional gambling, the growth of resources (here the volume of positive or negative constraints) exhibits an expected log growth rate which is best defined in terms of the entropy of the vector of probabilities on the unlabelled samples. Assuming, for the moment, the prior distribution over predictors is correct, so that the sampling vector on $\mathbf{U}$ ($\mathbf{m}^{\pm}$) is directly proportional to the actual probabilities, $\mathbf{p}^{+}$ or $\mathbf{1} - \mathbf{p}^{+} = \mathbf{p}^{-}$, we may concisely describe the expected log success rate $\mathbf{W}$ of the sampling scheme for maximizing gain. Since the active sampling routine will be drawing the probability mass equivalent of one sample from the total probability mass over the unlabeled samples ($Z^{\pm} = \sum_{u \in \mathbf{U}} p^{\pm}_{u}$) with expected payoff $p^{\pm}_{u} = Z^{\pm} m^{\pm}_{u}$, the expected log success rate is given by:

$$\begin{aligned}
\mathbf{W}_{\mathbf{U}}(\mathbf{m}^{+}, \mathbf{p}^{+}) &= (Z^{+})^{-1} \sum_{u \in \mathbf{U}} p_u^{+} \log\left(Z^{+} m_u^{+}\right) && (7.48) \\
&= (Z^{+})^{-1} \sum_{u \in \mathbf{U}} p_u^{+} \log p_u^{+} && (7.49)
\end{aligned}$$

Likewise, the success of the sampling scheme to maximize risk corresponds to:

$$\begin{aligned}
\mathbf{W}_{\mathbf{U}}(\mathbf{m}^{-}, \mathbf{p}^{-}) &= (Z^{-})^{-1} \sum_{u \in \mathbf{U}} p_u^{-} \log\left(Z^{-} m_u^{-}\right) && (7.50) \\
&= (Z^{-})^{-1} \sum_{u \in \mathbf{U}} (1 - p_u^{+}) \log(1 - p_u^{+}) && (7.51)
\end{aligned}$$

Examining the collective sampling rates of a combined max/min strategy, we see that a mixture of strategies preserving the proportion of expected return for each strategy($S_{Z^{\pm}} \propto sZ^{\pm}$) exactly reproduces the binary entropy of the unlabelled sample set:

$$\begin{aligned}
\sum_{u \in \mathbf{U}} \sum_{\{\mathbf{m},\mathbf{p}\}^{\pm}} \mathbf{W}_{\mathbf{U}}(\mathbf{m}_u, \mathbf{p}_u) &\approx S_{Z^{+}} Z +^{-1} \sum_{u \in \mathbf{U}} \left(p_u^{+} \log p_u^{+}\right) + \\
&\qquad S_{Z^{-}} (Z^{-})^{-1} \sum_{u \in \mathbf{U}} (1 - p_u^{+}) \log(1 - p_u^{+}) && (7.52) \\
&= s \sum_{u \in \mathbf{U}} \left(\left(p_u^{+} \log p_u^{+} + (1 - p_u^{+}) \log(1 - p_u^{+})\right)\right) && (7.53) \\
&= s \sum_{u \in \mathbf{U}} \sum_{\mathbf{p} \in \{\mathbf{p}^{\pm}\}} \mathcal{H}(p_u) && (7.54) \\
&= s \sum_{u \in \mathbf{U}} \frac{1}{2} \sum_{p_u \in \{p_u^{\pm}\}} \mathcal{H}(p_u) && (7.55)
\end{aligned}$$

This gives a maximum expected log growth rate on the individual samples of $\mathbf{W}(\mathbf{m}_u^{\pm}, \mathbf{p}_u^{\pm}) = -\mathcal{H}_{\text{bin}}(\mathbf{p}_u^{\pm})$. Thus, the combination of risk maximization and minimization strategies may be seen as an optimal partitioning of the probability space with maximal expected growth properties.

The loss to our strategies due to incorrect prior distributions over predictors may also be calculated in the usual way, with log gain of the sampling scheme defined in the general case:

$$\mathbf{W}(\mathbf{m}^{\pm}, \mathbf{p}^{\pm}) = -\mathcal{H}(\mathbf{p}^{\pm}) - \mathcal{D}(\mathbf{p}^{\pm} || Z^{\pm}\mathbf{m}^{\pm})$$

with $\mathbf{m}^{\pm}$ being the sample vectors defined according to the strategies' expectations of gain or error volumes and $\mathbf{p}^{\pm}$ the correct vector.

Note that the algorithm fails to make gains only in the extreme case of $\mathbf{m}^{\pm} \perp \mathbf{p}^{\pm}$, a situation we refer to as the "Bush corollary" in which minimal understanding of the priors leads to maximal confidence in the quality of the predictions, resulting in all sampling resources being concentrated in null locations. Fortunately, in the theoretical setting, learning occurs in this case.

7.4.2 Controlling Sampling

By effectively splitting the function volume on the unlabelled sample set into orthogonal components corresponding to the expected risk and gain, this max/min strategy provides an intuitive informational framework for examining the active sampling component of a semi-supervised strategy. Orthogonality of the risk and gain strategies guarantees that maximization of agreement of predictor and label on the selected sample results in maximization of expected risk on the remaining field. Likewise, minimization of agreement of the label with the prediction leads to minimization of expected risk on the remaining field. Methods for more refined control of the active sampling method such as informed alternation of strategies through measurement of informational quantities is a matter for further study.

Due to the penalties associated with delay of discovery of relevant documents in the Rankhedge setting and the monotonically decreasing probability of relevance at rank, the probabilistic method is not naturally applicable in the current context. However, the informational approach presented here should prove insightful in other contexts, and we note that the complementary sampling strategies bear strong resemblance to other familiar max/min strategies such as the "exploration" and "exploitation" paradigm of reinforcement learning.

In the final section, we present a simple application of the probabilistic sampling method in a familiar "exploration" and "exploitation" context as a demonstration of the method's effectiveness in hedging against the pitfalls associated with a naive greedy selection strategy. The data set in this example is quite simple and designed to illustrate a single mode of behavior, nevertheless an interesting transition naturally occurs between the early "exploration" phase in which probabilistic sampling ranges over a broad sampling distribution and a later "exploitation" phase where the distribution has focused on the selections of the single best predictor.

7.5 A Search Problem

Our Rankhedge algorithm relies on the assumption that relevance is a monotonically decreasing function of rank. This severely restricts the potential for adversarial placement of samples to thwart the naive greedy selection algorithm employed in our methodology. However, in many other contexts, the greedy sampling method is easily led astray by fairly straight forward strategies designed to divert attention of the sampling method from relevant regions. In the absence of exhaustive search to find the sample whose labelling will maximally diminish the expected risk or maximally increase the expected reward as measured over the remaining sample field, the probabilistic sampling method may provide a hedge against malicous risk or reward distributions on the unlabelled instances. Correlations among samples which influence the potential payoff of the sampling algorithm and which are ignored by a naive greedy strategy (selecting a sample with maximum *instantaneous risk or reward*) are naturally captured by the probabilistic method. Thus, the probabilistic method serves as an intermediate strategy between the extremes of naive greedy selection (maximizing instantaneous reward/risk) and optimal greedy selection (maximizing reward or minimizing risk via exhaustive search).

In Figure 7.1, we provide the results of an application of the probabilistic technique to a fairly common situation in which an adversary has placed a number of decoys in the sample field to obscure a correlated region of samples with high potential relevance. The test was defined in a straightforward manner, with $F = 100$ predictors providing binary classifications for $T = 1000$ instances. Relevance is likewise binary, resulting in a test space of size: $|F = 100| \times |T = 1000| \times \{0,1\} \mapsto \{0,1\}$.

The predictors' classifications of the sample space are produced as follows. For 90% of the instance space, each predictor produces an expected rate of positive classifications ($F_i = 1$) of 40%. These positive predictions are uniformly distributed throughout this portion of the space. A sole predictor F_{best} is defined to have an accuracy of 80% on its positive classifications as well as a 13% false negative rate.

The remaining 10% of instances serve as a decoy set. Decoys induce an expected 70% rate of positive classification uniformly across predictors including F_{best}. The relevance of decoy samples is zero. The value of all constants used in the test were arbitrarily selected and may vary over a very wide range with qualitatively similar results.

We employ the Hedge algorithm with two different active sampling strategies to attempt to identify relevant samples in the instance space. Since our goal is simply to compare sampling strategies, an optimal value for β is established via examination of the exact loss of the best predictor. Both sampling strategies select samples according to expected relevance as defined by the w.l.c. of classifications by the predictors.

The results of our test (Fig. 7.1) clearly demonstrate the potential pitfalls associated with a naive greedy strategy. In plots 7.1a), we see that, in the probabilistic strategy, the precision or average number of relevant samples in the selected pool rapidly approaches the limiting 80% defined by the accuracy of the positive classifications of the best predictor. The naive greedy algorithm, on the other hand, clearly shows the risk associated with pursuit of the decoy samples— 70% of which must be exhausted before relevant samples begin to be discovered in significant numbers.

Plots 7.1b) likewise demonstrate the probabilistic sampling method's superior ability to focus on the best predictor. As expected, the weight history of the best predictor exhibits a fairly continuous and rapid convergence of the Hedge distribution to the eventual target distribution in which all weight is focused on the best predictor. As expected, the weight history for the greedy method exhibits sporadic jumps toward the target distribution only after the supply of decoys dwindles.

Plots 7.1c) are provided for reference. They display the w.l.c. of Hedge predictions for a sample at the moment of selection. In plots for both methods, we see that, as the Hedge distribution focuses on the best predictor, the sampling distribution naturally focuses on that predictors' positive classifications. This will remain the steady state until the supply of positive classifications is exhausted. In the case of the probabilistic algorithm, the transition from broad to focused sampling distributions proceeds in a particularly natural manner, further supporting our intuition that the probabilistic sampling method offers the possibility of naturally integrating the "exploration" and "exploitation" phases of learning methods in diverse contexts in a consistent Bayesian framework.

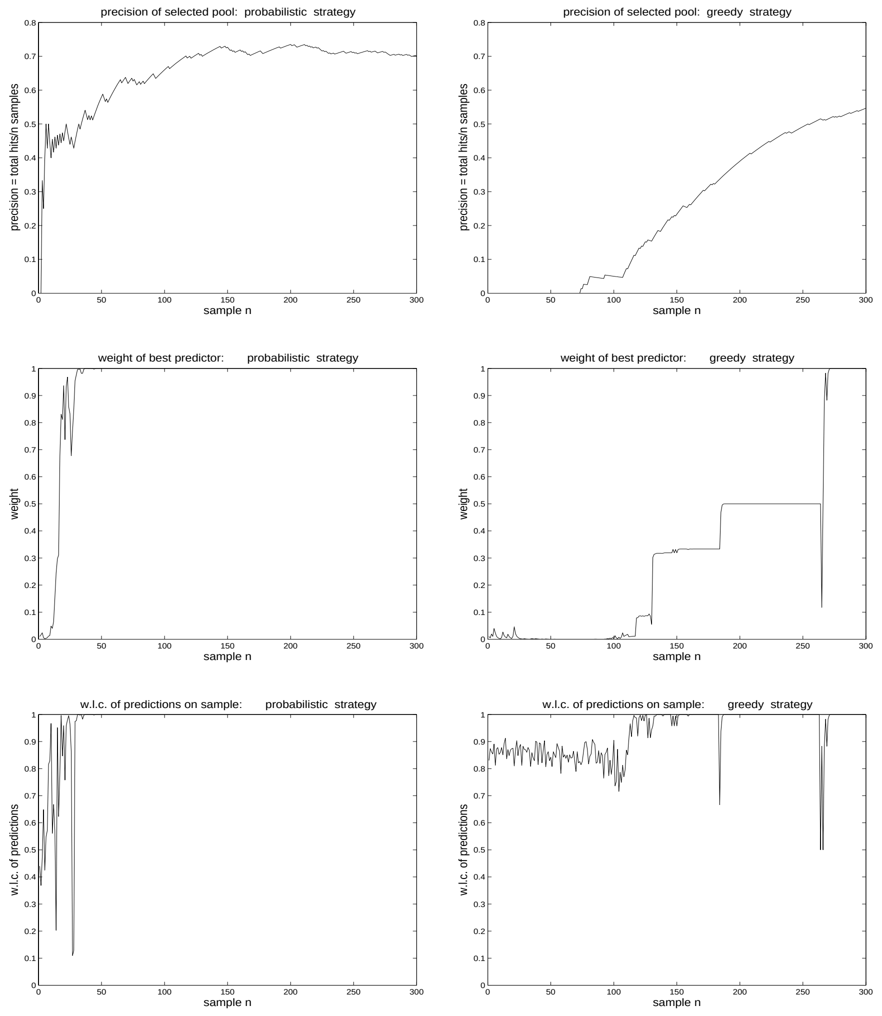


Figure 7.1: Probabilistic vs. Greedy Sampling Strategy: a) precision of selected sample pool, b) weight history of best predictor, and c) w.l.c. of predictions on samples.

Chapter 8

Conclusion

This thesis has presented a new algorithm, Rankhedge for the simultaneous solution of metasearch, pooling, and system evaluation in the IR context. The algorithm is the first to integrate these three tasks, previously addressed individually in the IR literature, into a singe unified model. In addition, by recasting the three problems in the context of the Hedge algorithm for on-line learning, the solution achieves results in all three fundamental tasks with proveable worst case bounds under certain measures of performance.

An early incarnation of the Rankhedge algorithm, Rankhedge(Log Prior), with a loss function designed to reflect an individual sample's contribution to the familiar Average Precision measure of system accuracy produced excellent results relative to the standard Depth-n method for TREC pooling. The Rankhedge(Log Prior) method also proved superior to the priority queue pooling method of Cormack et al. in the critical early stages of the retrieval process, but tended to falter in the latter stages, demonstrating a tendency to overfocus on the most accurate predictors.

This tendency to overfocus proved to be a natural result of the logarithmic nature of the loss function, with losses approximately proportional to the log prior of relevance at rank. The second incarnation of Rankhedge, employing a loss function based on an accurate prior probability of relevance, was introduced to corrected this inefficiency and performance of the second Rankhedge algorithm in all three tasks proved to be equivalent and in most cases significantly superior to that of previous algorithms including the priority queue method. In addition, the exact form of the loss function based on inverse rank reflects the expansion of the set of instances spanned by rank and is therefore a property of the support of the rank function. Thus, rather than being an arbitrary relevance curve fit specifically to the TREC data, the form of prior relevance at rank employed by Rankhedge has potential applicability to the general context of ranked lists.

Practical success of the Rankhedge algorithm proceeds from the sound theoretical underpinnings of the Hedge method. The thesis demonstrated, via information-theoretic examination of the exponential update rule of the Rankhedge algorithm that the Hedge technique for maintaining the weight distribution over predictors is, in fact, a maximum entropy method (with minimum KL distance between successive distributions given the constraints of the labelled instances). Further, as a classification method, Rankhedge was shown to be closely related to other methods for Bayesian on-line prediction, with the distribution over predictors reflecting the best estimate of the relative posterior probabilities of predictor accuracy. In conjunction with the exponential update rule, the Hedge classification method may be interpreted as an adaptation of Bayes optimal on-line classification methods for the lossless situation (error $\epsilon = 0$) to the lossy context in which the best predictor has an error rate greater than zero.

Continuing with the information-theoretic analysis of the Hedge environment, two different extensions of the Hedge related methods were also defined. In the first, information-theoretic similarity measures on the space of predictors were developed to provide a method for decorrelating the initial Hedge weight distribution. The information-theoretic similarity measure on a predictor pair was demonstrated to correspond roughly to the sum of log accuracies of each predictor with its counterpart defined to be the "target." A probabilistic measure incorporating the prior relevance was shown to correspond to the dice coefficient on the sets of instances. An extremal measure, in which each ranked instance was assigned an expected relevance of one, corresponded, likewise, to a straightforward expression of set similarity.

The probabilistic similarity measure was applied to decorrelate the initial distribution on the predictor set with little effect on the evolution of the Rankhedge learning process due to the rapidity with which the exponential update method of Rankhedge corrects for the bias error. Comparison of the weight histories proceeding from uniform and decorrelated initial distributions demonstrated rapid converge due to the joint effect of the exponential update rule and the active learning mechanism of Rankhedge. The availability of a similarity measure on the predictor space has implications beyond the immediate implementation of Rankhedge, however, and these will be discussed in the next section.

Finally, a second extension of the method of Rankhedge was suggested by the optimality of proportional resource allocation. An alternative to the greedy active learning method proceeds naturally from the original gambling scenario by allowing sampling resources to be allocated probabilistically on the sample axis. By extending the proportional resource allocation scheme to a second dimension, the active sampling method may allocate resources at rates which reflect the expected posterior distribution over hypotheses as measured over the remaining unlabelled samples. As a generic method for sample selection with optimal expected retrieval rates, the probabilistic technique has the potential for application to a broad variety of contexts.

8.1 Future Work

Several questions remain regarding the application of Rankhedge to the TREC conference data. As demonstrated in Figure 5.11, substantial variability in precision exists when the conference data is examined on a query by query basis, and the possibility of manipulating the loss function to reflect this per-query variability is an open question. The applicability of the prior relevance assumptions to metasearch and system evaluation contexts beyond the controlled environment of the TREC conference is also a matter for further exploration.

In addition, substantially more differentiation exists in quantities of predictor bias on a per-query basis than in the average values used in our test of the decorrelation technique. It is possible that a concentration on the bias characteristics of individual queries may yield greater rewards for the decorrelation method.

In the broader range of applications beyond the TREC conference, the definition of a similarity measure on the predictor space opens the possibility of adding a layer of contextual control to the retrieval process. Search engines may vary greatly in accuracy, depending on the context of the queries. For example a medical query will likely be handled in a significantly different manner by search engines with domain specific algorithms or knowledge than by the more generic engines examined in the TREC retrieval tracks. A similiarity measure on the space of predictors provides an opportunity for clustering and segmentation of the predictor space and suggests the possibility of contextualized confidence weighting of predictors depending on their proximity in predictor space to a set of reference queries.

The probabilistic active sampling method also holds the promise of applicability to a range of applications beyond the limits of the TREC conference. The probabilistic method was developed with a distributed context in mind, and it has potential application to problems of functional surface reconstruction via manipulation of sampling rates by hierarchically coordinated agents. Problems defined via a Markov or Gaussian random field [100, 101] on an $n \times n$ distance matrix connecting the discrete elements of the sample space are particularly suited to this method of sampling, as are the distributed problem definitions in the spatial aggregation framework [67, 68]. As noted in Chapter Seven, the paradigm of exploration and exploitation modes defined in the reinforcement learning context also bears strong resemblance to the orthogonal partitioning of risk and relevance sampling in our probabilistic method.

8.2 Beyond Traditional Metasearch

A central theme of this thesis has been the theoretical situation of the three problems from IR— metasearch, system evaluation and pooling— within the generic context of computational learning theory. By establishing connections to various incarnations of information-theoretically grounded resource allocation and prediction methods— particularly the Hedge algorithm and on-line Bayesian prediction— these three problems from IR serve to illuminate the broader applicability of the information-theoretic methods to the development and analysis of learning and search techniques in a broad range of contexts. When viewed through the unifying lense of optimal resource allocation, the techniques described in the thesis suggest a natural method for generalization of the search and retrieval process along multiple dimensions. Duality of the proportional allocation model defined on the predictor and sample axes suggests a straightforward generalization of the methods to the distributed retrieval situation in which context specific behavior— such as access of independent data sets by the retrieval systems— indicates the need for context sensitive segmentation and control in the predictor space. Generalization of the problem definition along the temporal dimension leads to possible applications such as filtering—

the sampling and prediction of and adaptation to a (possibly non-stationary) stream of instances. Here again, application of the techniques to the filtering context should proceed naturally from the resource allocation framework by introducing a new (temporal) degree of freedom. Finally, the close relationship of the Hedge algorithm to the MRF/Gibbs informational structures from statistical physics indicates that the information-theoretic techniques of the thesis have applicability to myriad physical situations beyond the context of IR. Since it is, at its foundation, a maximal entropy method, the optimal resource allocation and sampling paradigm may potentially serve not only as a basis for developing efficient algorithms for these contexts but also as an internal organizational principle of the physical systems themselves.

Bibliography

[1] *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1995.

[2] *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, Louisiana, USA, Sept. 2001. ACM Press, New York.

[3] *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Totonto, CA, Aug. 2003. ACM Press, New York.

[4] S. Argamon-Engelson and I. Dagan. Committee-based sample selection for probabilistic classifiers. *Journal of Artificial Intelligence Research*, 11:335–360, 1999.

[5] J. Aslam and M. Montague. Bayes optimal metasearch: A probabilistic model for combining the results of multiple retrieval systems. In N. J. Belkin, P. Ingwersen, and M.-K. Leong, editors, *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 379–381. ACM Press, July 2000.

[6] J. A. Aslam and M. Frost. An information-theoretic measure for document similarity. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [3].

[7] J. A. Aslam and M. Montague. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [2], pages 276–284.

[8] J. A. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch and the efficient evaluation of retrieval systems via the hedge algorithm. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [3].

[9] J. A. Aslam, V. Pavlu, and R. Savell. A unified model for metasearch, pooling, and system evaluation. In *Proceedings of the 2003 ACM CIKM: Twelfth International Conference on Information and Knowledge Management*, New Orleans LA, USA, Nov. 2003.

[10] J. A. Aslam, V. Pavlu, and E. Yilmaz. Measure-based metasearch. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM Press, August 2005. To appear.

[11] J. A. Aslam and R. Savell. On the effectiveness of evaluating retrieval systems in the absence of relevance judgements. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [3].

[12] B. T. Bartell. *Optimizing Ranking Functions: A Connectionist Approach to Adaptive Information Retrieval*. PhD thesis, University of California, San Diego, 1994.

[13] B. T. Bartell, G. W. Cottrell, and R. K. Belew. Automatic combination of multiple ranked retrieval systems. In W. B. Croft and C. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR*

Conference on Research and Development in Information Retrieval, pages 173–181, Dublin, Ireland, July 1994. Springer-Verlag, London.

[14] N. Belkin, P. Kantor, C. Cool, and R. Quatrain. Combining evidence for information retrieval. In Harman [42], pages 35–43.

[15] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, series B*, 36(2):192–236, 1974.

[16] J. Besag and P. J. Green. Spatial statistics and Bayesian computation. *Journal of the Royal Statistical Society, series B*, 16(2):395–407, 1993.

[17] D. Black. *Theory of Comittees and Elections.* Cambridge University Press, 1958.

[18] J. C.Borda. *Memoire sur les elections au scrutin.* In Histoire de l'Academie Royale des Sciences, 1781.

[19] N. Cesa-Bianchi, D. P. Helmbold, and S. Panizza. On bayes methods for on-line boolean prediction. In *Proceedings of the ninth annual conference on Computational learning theory*, pages 314–324. ACM Press, 1996.

[20] F. Chung. Spectral graph theory. *Regional Conference Series in Mathematics, American Mathematical Society*, 92:1–212, 1997.

[21] W. W. Cohen, R. E. Schapire, and Y. Singer. Learning to order things. In *Proceedings of the 1997 conference on Advances in neural information processing systems 10*, pages 451–457. MIT Press, 1998.

[22] D. A. Cohn, Z. Ghahramani, and M. I. Jordan. Active learning with statistical models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 705–712. The MIT Press, 1995.

[23] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In Croft et al. [27], pages 282–289.

[24] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley Series in Telecommunications. John Wiley & Sons, New York, NY, USA, 1991.

[25] N. Craswell, D. Hawking, and P. Thistlewaite. Merging results from isolated search engines. In *Proceedings of the Tenth Australasian Database Conference*, Aukland, New Zealand, Jan. 1999. Springer-Verlag.

[26] W. B. Croft. Combining approaches to information retrieval. In W. B. Croft, editor, *Advances in Information Retrieval: Recent Research from the Center for Intelligent Information Retrieval*, chapter 1. Kluwer Academic Publishers, 2000.

[27] W. B. Croft, A. Moffat, C. J. van Rijsbergen, R. Wilkinson, and J. Zobel, editors. *Proceedings of the 21th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, Aug. 1998. ACM Press, New York.

[28] I. Dagan and S. P. Engelson. Committee-based sampling for training probabilistic classifiers. In *International Conference on Machine Learning*, pages 150–157, 1995.

[29] M. de Condorcet. *Essai surl'application de l'analyse a la probabilitie des decisions rendues a la pluralite des voix.* 1785.

[30] P. Diaconis. A Generalization of Spectral Analysis with Application to Ranked Data. *The Annals of Statistics*, 17(3):949–979, 1989.

[31] B. Eisenberg and R. L. Rivest. On the sample complexity of pac-learning using random and chosen examples. In *COLT '90: Proceedings of the*

third annual workshop on Computational learning theory, pages 154–162. Morgan Kaufmann Publishers Inc., 1990.

[32] H. L. Fisher and D. R. Elchesen. Effectiveness of combining title words and index terms in machine retrieval searches. *Nature*, 238:109–110, July 1972.

[33] E. A. Fox, M. P. Koushik, J. Shaw, , R. Modlin, and D. Rao. Combining evidence from multiple searches. In D. Harman, editor, *The First Text REtrieval Conference (TREC-1)*, pages 319–328, Gaithersburg, MD, USA, Mar. 1993. U.S. Government Printing Office, Washington D.C.

[34] E. A. Fox and J. A. Shaw. Combination of multiple searches. In Harman [42], pages 243–249.

[35] W. B. Frakes and R. A. Baeza-Yates, editors. *Information Retrieval: Data Structures & Algorithms*. Prentice-Hall, 1992.

[36] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969, 2003.

[37] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, Aug. 1997.

[38] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Mach. Learn.*, 28(2-3):133–168, 1997.

[39] T. Geisel. *Green Eggs and Ham*. Random House, NY, 1960.

[40] T. Geisel. *The Lorax*. Random House, NY, 1971.

[41] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions, and the bayesian restoration of images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[42] D. Harman, editor. *The Second Text REtrieval Conference (TREC-2)*, Gaithersburg, MD, USA, Mar. 1994. U.S. Government Printing Office, Washington D.C.

[43] D. Harman. Overview of the third text REtreival conference (TREC-3). In Harman [44], pages 1–19.

[44] D. Harman, editor. *Overview of the Third Text REtrieval Conference (TREC-3)*, Gaithersburg, MD, USA, Apr. 1995. U.S. Government Printing Office, Washington D.C.

[45] D. Haussler, M. Kearns, and R. Schapire. Bounds on the sample complexity of bayesian learning using information theory and the vc dimension. In *Proceedings of the Fourth Annual Workshop on Computational Learning Theory*. Morgan Kaufmann, 1991.

[46] D. Haussler, M. Kearns, H. S. Seung, and N. Tishby. Rigorous learning curve bounds from statistical mechanics. *Mach. Learn.*, 25(2-3):195–236, 1996.

[47] T. Haveliwala and S. Kamvar. The second eigenvalue of the google matrix, 2003.

[48] D. A. Hull, J. O. Pedersen, and H. Schütze. Method combination for document filtering. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 279–287, Zurich, Switzerland, 1996. ACM Press, New York.

[49] E. T. Jaynes. On the Rationale of Maximum Entropy Methods. In *Proc. IEEE, Volume 70, p. 939-952*, pages 939–952, 1982.

[50] E. T. Jaynes. *Prior Probabilities*, pages 114–130. D. Reidel Publishing Company, Boston, Massachusetts, 1983.

[51] M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[52] J. S. Kelly. *Social Choice Theory: An Introduction*. Springer-Verlag, 1988.

[53] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 315–322. Morgan Kaufmann Publishers Inc., 2002.

[54] J. Lafferty and G. Lebanon. Information diffusion kernels. In S. T. S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 375–382. MIT Press, Cambridge, MA, 2003.

[55] G. Lebanon and J. Lafferty. Cranking: Combining rankings using conditional probability models on permutations. In *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002)*, Sydney, AU, June 2002.

[56] J. H. Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1], pages 180–188.

[57] J. H. Lee. Analyses of multiple evidence combination. In N. J. Belkin, A. D. Narasimhalu, and P. Willett, editors, *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 267–275, Philadelphia, Pennsylvania, USA, July 1997. ACM Press, New York.

[58] D. D. Lewis and W. A. Gale. A sequential algorithm for training text classifiers. In W. B. Croft and C. J. van Rijsbergen, editors, *Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval*, pages 3–12, Dublin, IE, 1994. Springer Verlag, Heidelberg, DE.

[59] D. Lin. An information-theoretic definition of similarity. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 296–304. Morgan Kaufmann Publishers Inc., 1998.

[60] N. Littlestone and M. Warmuth. The weighted majority algorithm. *Information and Computation*, 108(2):212–261, 1994.

[61] R. Manmatha, T. Rath, and F. Feng. Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [2], pages 267–275.

[62] A. K. McCallum and K. Nigam. Employing EM in pool-based active learning for text classification. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 350–358, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[63] M. Montague and J. A. Aslam. Metasearch consistency. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [2], pages 386–387.

[64] M. Montague and J. A. Aslam. Relevance score normalization for metasearch. In *Proceedings of the 2001 ACM CIKM: Tenth International Conference on Information and Knowledge Management*, pages 427–433, Atlanta, Georgia, USA, Nov. 2001.

[65] M. Montague and J. A. Aslam. Condorcet fusion for improved retrieval. In K. Kalpakis, N. Goharian, and D. Grossman, editors, *Proceedings of the Eleventh International Conference on Information and Knowledge Management*, pages 538–548. ACM Press, November 2002.

[66] H. Moulin. *Axioms of Cooperative Decision Making.* Cambridge University Press, 1988.

[67] N. Ramakrishnan, C. Bailey-Kellogg, S. Tadepalli, and V.N. Pandey. Gaussian processes for active data mining of spatial aggregates. In *Proc. 18th International Workshop on Qualitative Reasoning, Chicago, Il.*, 2004.

[68] N. Ramakrishnan, C. Bailey-Kellogg, S. Tadepalli, and V.N. Pandey. Gaussian processes for active data mining of spatial aggregates. In *to appear in Proc. SIAM Data Mining Conference*, 2005.

[69] K. B. Ng and P. B. Kantor. An investigation of the preconditions for effective data fusion in ir: A pilot study. In *Proceedings of the 61th Annual Meeting of the American Society for Information Science*, 1998.

[70] K. B. Ng and P. B. Kantor. Predicting the effectiveness of naive data fusion on the basis of system characteristics. *Journal of the American Society for Information Science*, 51(13):1177–1189, 2000.

[71] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford Digital Library Technologies Project, 1998.

[72] D. Pollard. *Convergence of Stochastic Processes.* Springer-Verlag, 1984.

[73] W. H. Riker. *Liberalism Against Populism.* Waveland Press, 1982.

[74] D. Rockmore. Some applications of generalized fft's. In *Proceedings of the DIMACS Workshop on Groups and Computation June 7-10 1995*, pages 329–369. American Mathematical Society, 1997.

[75] N. Roy and A. McCallum. Toward optimal active learning through sampling estimation of error reduction. In *Proc. 18th International Conf. on Machine Learning*, pages 441–448. Morgan Kaufmann, San Francisco, CA, 2001.

[76] D. G. Saari. Explaining all three-alternative voting outcomes. *Journal of Economic Theory*, 87(2):313–355, Aug. 1999.

[77] D. G. Saari. The mathematics of voting: Democratic symmetry. *The Economist*, page 83, Mar. 2000.

[78] G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. In *Proc. 17th International Conf. on Machine Learning*, pages 839–846. Morgan Kaufmann, San Francisco, CA, 2000.

[79] E. W. Selberg. *Towards Comprehensive Web Search*. PhD thesis, University of Washington, 1999.

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

[81] J. A. Shaw and E. A. Fox. Combination of multiple searches. In Harman [44], pages 105–108.

[82] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, page 731. IEEE Computer Society, 1997.

[83] B. Shu and S. Kak. A neural network-based intelligent metasearch engine. *Information Sciences*, 120:1–11, 1999.

[84] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [2], pages 66–73.

[85] A. D. Sokal. Mone Carlo Methods in statistical mechanics: foundations and new algorithms. *Cours de Troisieme Cycle de la Physique en Suisse Romande*, 1989.

[86] P. Thompson. A combination of expert opinion approach to probabilistic information retrieval, part 1: the conceptual model. *Information Processing and Management*, 26(3):371–382, 1990.

[87] P. Thompson. A combination of expert opinion approach to probabilistic information retrieval, part 2: mathematical treatment of CEO model 3. *Information Processing and Management*, 26(3):383–394, 1990.

[88] S. Tong and D. Koller. Support vector machine active learning with applications to text classification. In P. Langley, editor, *Proceedings of ICML-00, 17th International Conference on Machine Learning*, pages 999–1006, Stanford, US, 2000. Morgan Kaufmann Publishers, San Francisco, US.

[89] G. G. Towell, E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection FUsion strategies for information retrieval. In *International Conference on Machine Learning*, pages 540–548, 1995.

[90] V. Vapnik. *The Nature of Statistical Learning Theory.* Springer, NYl, 1995.

[91] C. C. Vogt. *Adaptive Combination of Evidence for Information Retrieval.* PhD thesis, University of California, San Diego, 1999.

[92] C. C. Vogt. How much more is better? Characterizing the effects of adding more IR systems to a combination. In *Content-Based Multimedia Information Access (RIAO)*, pages 457–475, Paris, France, Apr. 2000.

[93] C. C. Vogt and G. W. Cottrell. Fusion via a linear combination of scores. *Information Retrieval*, 1(3):151–173, Oct. 1999.

[94] C. C. Vogt, G. W. Cottrell, R. K.Belew, and B. T. Bartell. Using relevance to train a linear mixture of experts. In E. Voorhees and D. Harman, editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 503–515, Gaithersburg, MD, USA, 1997. U.S. Government Printing Office, Washington D.C.

[95] E. Voorhees and D. Harman. Overview of the Eighth Text REtrieval Conference (TREC-8). In D. Harman, editor, *The Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, USA, 2000. U.S. Government Printing Office, Washington D.C.

[96] E. M. Voorhees, N. K. Gupta, and B. Johnson-Laird. Learning collection fusion strategies. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* [1], pages 172–179.

[97] M. Warmuth and C. Gentile. Proving relative loss bounds for on-line learning algorithms using bregman divergences. Presented at Thirteenth Annual Conference on Computational Learning Theory. Stanford University, June 2000.

[98] T. L. H. Watkin, A. Rau, and M. Biehl. The statistical mechanics of learning a rule. *Reviews of Modern Physics*, 65:499–556, 1993.

[99] Y. Weiss. Segmentation using eigenvectors: A unifying view. In *Proceedings of the International Conference on Computer Vision-Volume 2*, page 975. IEEE Computer Society, 1999.

[100] J. D. L. X. Zhu, Z. Ghahramani. Semi-supervised learning using gaussian fields and harmonic functions. In *Proc. 20th International Conference on Machine Learning (ICML'03)*, pages 912–919. AAAI Press, January 2003.

[101] X. Zhu, J. Lafferty, and Z. Ghahramani. Combining active learning and semi-supervised learning using gaussian fields and harmonic functions. In *Proc. of the ICML Workshop on the Continuum from Labeled to Unlabeled Data*, pages 58–65. AAAI Press, 2003.

[102] J. Zobel. How reliable are the results of large-scale retrieval experiments? In Croft et al. [27], pages 307–314.