Journal of Machine Learning Research 3 (2003) 1025-1058 Submitted 4/02; Published 2/03

A Family of Additive Online Algorithms for Category Ranking

Koby Crammer KOBICS@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science & Engineering
Hebrew University, Jerusalem 91904, Israel

Editors: Jaz Kandola, Thomas Hofmann, Tomaso Poggio and John Shawe-Taylor

Abstract

We describe a new family of topic-ranking algorithms for multi-labeled documents. The motivation for the algorithms stem from recent advances in online learning algorithms. The algorithms are simple to implement and are also time and memory efficient. We provide a unified analysis of the family of algorithms in the mistake bound model. We then discuss experiments with the proposed family of topic-ranking algorithms on the Reuters-21578 corpus and the new corpus released by Reuters in 2000. On both corpora, the algorithms we present achieve state-of-the-art results and outperforms topic-ranking adaptations of Rocchio's algorithm and of the Perceptron algorithm.

1. Introduction

The focus of this paper is the problem of topic ranking for text documents. We use the Reuters[1] corpus release 2000 (often referred to as RCV1) as our running example. In this corpus, there are about a hundred different topic codes. Each document in the corpus is tagged with a set of topics which are relevant to its content. For instance, a document from late August 1996 discusses a bill by Bill Clinton to increase the minimum wage by $0.90. This document is associated with 9 topics, such as `labour`, `economics`, `unemployment`, and `retail sales`. This example shows that often there is a semantic overlap between the topics that are marked as relevant. Given a feed of documents, such as the Reuters newswire, the task of topic ranking is concerned with ordering the topics according to their relevance for each document independently. We cast topic-ranking as a supervised learning problem. That is, we receive a training set of documents where each document is provided with a set of relevant topics. Given the set of labeled documents, the goal of the learning algorithm is to find a topic-ranking function that outputs a ranking of the topics for an input document. The rankings of the topics should reflect their relevance according to the content of the document.

In the machine learning community this setting is often referred to as a *multi-label* classification problem. The motivation of most if not all of the machine learning algorithms for this problem stem from a decision theoretic view. Namely, the output of the algorithms is a predicted set of relevant topics and the quality of the predictions are measured by how successful we are in making a separate decision on whether each topic is relevant or non-relevant. In this paper we attempt to combine

1. See http://about.reuters.com/researchandstandards/corpus for information.

techniques from statistical learning theory with the more practical goals of information retrieval (IR) tasks. We do so by adapting and generalizing techniques from online prediction algorithms to the particular task of topic ranking. Our starting point is the Perceptron algorithm (Rosenblatt, 1958), which was originally designed for binary classification problems.

Despite (or because of) its age and simplicity, the Perceptron algorithm and its variants have proved to be surprisingly effective in a broad range of applications in machine learning and information retrieval (see for instance (Freund and Schapire, 1998, Ng et al., 1997) and the references therein). The original Perceptron algorithm was designed for binary classification problems. Although generalizations of the Perceptron algorithm to non-binary classification problems were outlined in Duda and Hart's book (Duda and Hart, 1973), it has been only recently that provably-correct extensions of the Perceptron algorithm to non-binary decision problems have been suggested (Crammer and Singer, 2001b,a). In this paper, we build on recent work and devise an efficient generalization of the Perceptron algorithm to the task of topic ranking given multiclass multi-label feedback. While this work borrows ideas from recent work on multiclass problems (Crammer and Singer, 2001b), the task of topic ranking is more involved and requires a somewhat different proof technique. Since all algorithms in the family are generalizations of the Perceptron algorithm to topic ranking using multiclass multi-label feedback, we refer to the various variants as the MMP algorithm.

A few learning algorithms for multi-labeled data have been devised in the machine learning community. Two notable examples are two multi-label versions of AdaBoost called AdaBoost.MH and AdaBoost.MR (Schapire and Singer, 1999) and a multi-label generalization of Vapnik's Support Vector Machines by Elisseeff and Weston (2001). These two multi-label algorithms take the same general approach by reducing a multi-label problem to multiple binary problems and comparing all pairs of labels (topics in our setting). Thus, they require time and space that scale quadratically with the total number of topics per example. The two papers above suggested computational shortcuts to overcome the quadratic complexity. These shortcuts, however, imposed additional technical difficulties. Our starting point is similar as we use an implicit reduction into pairs. Yet, using a simple pre-computation, as we describe in the sequel, the space complexity of MMP is *linear* in the number of topics and sub-quadratic in time. In addition, MMP can be combined with almost any general ranking-loss so long as the loss is bounded. Different losses may reflect different requirements from the topic-ranking rule. Thus the ability to incorporate losses into the learning algorithm adds flexibility that could help in achieving high accuracy with respect to losses that are less common in the machine learning community such as the *average precision* that is often used in information retrieval tasks.

The paper is organized as follows. We start with a formal description of the problem in Section 2. We then describe the family of topic-ranking algorithms called MMP. The algorithms in the family differ in the topic-ranking loss they attempt to decrease. We then describe experiments with two multi-labeled corpora: the Reuters-21578 corpus and a newer and much larger release by Reuters (release 2000). The results obtained on the new release of Reuters are among the first published results for this corpus. We compare the results achieved by MMP with an adaptation of the Perceptron algorithm to multi-label settings and also with a multi-label version of Rocchio's algorithm. The results obtained in the experiments indicate that MMP offers a viable alternative to existing algorithms that can be used with the largest text corpora that are currently available.

2. Problem Setting

As discussed above, the text application that the paper is concerned with is topic-ranking of documents where each document can be associated with *multiple* relevant topics. In this problem we are given access to a stream of documents. Each document is associated with *zero* or more topics from a predefined set of topics. We denote the set of possible topics by $\mathcal{Y}$ and the number of different topics by k ($k = |\mathcal{Y}|$). In the new distribution of the Reuters corpus (which we refer to as Reuters-2000), there are 103 different topics[2], while Reuters-21578 consists of 91 different topics. Since there are semantic overlaps between the topics, a document is typically associated with more than one topic. More formally, a document is labeled with a set $y \subset \mathcal{Y}$ of relevant topics. In the Reuters-2000 corpus, the average size of y is 3.2 while in Reuters-21578 the average size of y is 1.24. We say that a topic r, also referred to in the machine learning community as a class or category, is *relevant* for a given document if r is in the set of relevant topics, $r \in y$.

There are numerous different information filtering and routing tasks that are of practical use for multi-label problems. The focus of this paper is the design, analysis, and implementation of a family of *topic-ranking* algorithms. That is, given a document, the algorithms we consider return a list of topics ranked according to their relevance. However, the feedback for each document is the set y. Thus, the topics of each document in the training corpus are not ranked but rather designated as relevant or non-relevant. Put another way, while the topic-ranking algorithm outputs a total ordering of the topics, the feedback can be viewed as a coarse partial ordering into two sets.

Each document in our study is represented using the vector space model (Salton, 1991) as a vector in $\mathbb{R}^n$. We thus denote a document by its vector representation $\bar{x} \in \mathbb{R}^n$. All the topic-ranking algorithms we discuss in this paper use the same mechanism: each algorithm maintains a set of k prototypes, $\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_k$. Analogous to the representation of documents, each prototype is a vector, $\bar{w}_r \in \mathbb{R}^n$. The specific vector-space representation we used is based on the pivoted length normalization of Singhal et al. (1996) whose description is deferred to Section 9.2. The family of algorithms we present constructs the prototype vectors from examples, i.e., from a corpus S containing T documents, each of which is associated with a set of relevant topics,

$$S = \left\{ (\bar{x}^t, y^t) \mid 1 \le t \le T, \bar{x}^t \in \mathbb{R}^n, y^t \subset \mathcal{Y} \right\} .$$

The set of prototypes induces a ranking on the topics according to their similarity to the vector representation of the document. That is, given a document $\bar{x}$ the inner-products $\bar{w}_1 \cdot \bar{x}, \bar{w}_2 \cdot \bar{x}, \ldots, \bar{w}_k \cdot \bar{x}$ induce an ordering according to the relevance level of each topic. We say that topic r is ranked higher than topic s if $\bar{w}_r \cdot \bar{x} > \bar{w}_s \cdot \bar{x}$. Given a feedback, i.e. the set of relevant topics y of a document $\bar{x}$, we say that the ranking induced by the prototypes is *perfect* if all the relevant topics are ranked higher than the non-relevant topics. More formally, in a perfect ranking for any pair of topics $r \in y$ and $s \notin y$ ($s \in \mathcal{Y} - y$) the relevance score induced by $\bar{w}_r$ is higher than that the one induced by $\bar{w}_s$: $\bar{w}_r \cdot \bar{x} > \bar{w}_s \cdot \bar{x}$. We measure the quality of a perfect topic-ranking using the following notion of margin[3] We define the *margin* to be the size of the gap between the lowest score among the relevant topics to the highest score among the non-relevant topics,

$$\min_{r \in y} \{\bar{w}_r \cdot \bar{x}\} - \max_{s \notin y} \{\bar{w}_s \cdot \bar{x}\} \quad . \tag{1}$$

2. The topics in this corpus are organized in a hierarchical structure. In this paper we do not exploit this hierarchical structure and treat all topics in the same manner.
3. The definition of margin for topic-ranking is a generalization of the margin definition for multiclass problems with single-labeled data (Crammer and Singer, 2001b).

Note that the margin of a document $\bar{x}$ depends both on the set of relevant topics y and the prototypes $\bar{w}_1,\ldots,\bar{w}_k$. Clearly, if the topic-ranking of a document is perfect then the document's margin is positive. The margin can also be computed when the prototypes do not induce a perfect ranking. In this case the margin is negative. An illustration of topic-ranking margins is given in Fig. 1. The illustration shows the margin in case of a perfect ranking (left) and an imperfect one (right). In both cases there are 9 different topics. The relevant topics in the illustration are marked with circles and the non-relevant topics with squares. The values of the margins are designated by the lengths of the arrows where a positive margin is denoted by an arrow pointing down and a negative margin by an arrow pointing up. The notion of margin is rather implicit in the algorithms we discuss in the paper. However, it plays an important role in the formal analysis of the algorithms.

The family of algorithms we discuss in the paper are online algorithms where the amount by which we update each prototype is proportional to the instantaneous loss suffered on each round. This loss reflects how the prototype-induced ranking is far from being perfect. Rather than discussing specific losses, we prescribe a family of algorithms that can be employed with general bounded losses that attain a value of zero when the predicted ranking is perfect. Formally, give an instance $\bar{x}$ and a set of prototypes $\bar{w}_1,\ldots,\bar{w}_k$ we define $rank(\bar{x},r)$ to be the ranking of the topic indexed by r as induced by the inner-products $\bar{x}\cdot\bar{w}_1,\ldots,\bar{x}\cdot\bar{w}_k$. That is, we set $rank(\bar{x},r)$ to i if $|\{s : \bar{x}\cdot\bar{w}_s > \bar{x}\cdot\bar{w}_r\}| = i$. (We break ties arbitrarily.) Put another way, the ranking of each topic is its index in the list of topics sorted in descending order according to the inner-products between the prototypes and the vector representation of a document.

In our setting a ranking-loss for multilabled data is a function of the form,

$$loss \; : \; 2^{\mathcal{Y}} \times \pi_{|\mathcal{Y}|} \rightarrow \mathbb{R}_+ \; ,$$

where π_k denotes the set of all permutations over the k topics. We denote by

$$\mathcal{R} = (rank(\bar{x},1),\ldots,rank(\bar{x},k))$$

the topic-ranking obtained by applying $\bar{w}_1,\ldots,\bar{w}_k$ to $\bar{x}$. Using the above notation the loss that we suffer is $loss(y,\mathcal{R})$. Note that the ranking-loss functions are asymmetric due to the different role and form of the predicted ranking and the feedback. An example of such a loss is the ranking-loss employed by RankBoost (Freund et al., 1998). The ranking-loss employed by RankBoost counts the number of wrongly ordered pairs and, as we later see, it is one of the losses that we analyze and experiment with. We would like to note that our framework is somewhat more general than the one employed in (Freund et al., 1998), since the family of online algorithm we discuss subsequently can be used with other bounded losses.

3. Online Learning of Topic Ranking

The learning paradigm and the analysis that we use in this paper belongs to the mistake bound model for online learning. The algorithms we consider work in rounds. On round t an online learning algorithm receives a document $\bar{x}^t$. Given the document $\bar{x}^t$, the learning algorithm outputs a ranking $\mathcal{R}^t = (rank(\bar{x}^t,1),\ldots,rank(\bar{x}^t,k))$ which is induced by the inner-products $\bar{x}^t\cdot\bar{w}_1^t,\ldots,\bar{x}^t\cdot\bar{w}_k^t$. The algorithm then receives the (correct) set of relevant topics y^t. Given the feedback y^t and the predicted topic-ranking $\mathcal{R}^t$, the algorithm computes the associated loss $\ell^t = loss(y^t,\mathcal{R}^t)$. If ℓ^t is zero, the algorithm does not modify the set of prototypes it employs. Otherwise, it updates its topic-ranking rule by modifying the set of prototypes $\bar{w}_1^t,\ldots,\bar{w}_k^t$ in proportion to ℓ^t. As we discuss in

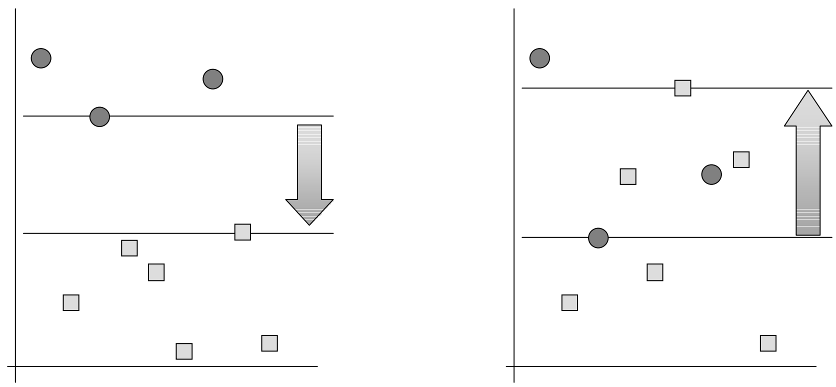

Figure 1: Illustration of the notion of the margin for a perfect ranking (left) and an imperfect ranking.

Section 5, the goal of the online topic-ranking algorithm is to suffer cumulative loss $(\sum_t \ell^t)$ that is competitive with a zero cumulative loss. (A zero cumulative loss is achieved by a set of prototypes $\bar{w}_1^*, \ldots, \bar{w}_k^*$ whose predicted rankings are always perfect.)

The family of algorithms we describe (and its corresponding analysis) employs a refined notion of mistake by examining all pairs of topics (labels). Whenever $\ell^t = loss(y^t, \mathcal{R}^t) > 0$ and the predicted ranking is not perfect, there must be at least one pair of topics (r,s) whose ordering according to the predicted ranking $\mathcal{R}^t$ disagrees with the feedback y^t, i.e., topic r is ranked not-above topic s $(\bar{x}^t \cdot \bar{w}_r^t \leq \bar{x}^t \cdot \bar{w}_s^t)$ but r is one of the relevant topics while s is not ($r \in y$, $s \notin y$). We therefore define the *error-set* of $(\bar{x}, y)$ as the set of all pairs whose predicted ranking disagrees with the feedback. The error-set is formally defined as,

$$E^t = \{(r,s) \in y \times (\mathcal{Y} - y) \ : \ \bar{w}_r^t \cdot \bar{x}^t \leq \bar{w}_s^t \cdot \bar{x}^t\} \ . \tag{2}$$

Many online algorithms update their prediction rule only on rounds on which they make a prediction error. Such algorithms are called *conservative*. We now give a definition that extends the notion of conservativeness to multi-label settings. This definition generalizes the definition from (Crammer and Singer, 2001b) used for multiclass problems with a uni-label feedback.

Definition 1 (Ultraconservative) *An online learning algorithm for topic (label) ranking from multi-label feedback that employs a set of prototypes $\bar{w}_1, \ldots, \bar{w}_k$ is ultraconservative if on round t it modifies only the prototypes corresponding to pairs from the error-set E^t.*

When the predicted ranking is perfect, the error-set is empty and therefore all the prototypes are left intact. Note also that the above definition does not require that *all* the prototypes from the error-set to be modified. Indeed, we discuss below one possible update scheme that modifies only a single pair from E^t.

4. A Family of Topic-Ranking Algorithms

To remind the reader, MMP is a descendant of the Perceptron algorithm and as such it is an *online* algorithm: It gets an example, outputs a ranking, and updates the hypothesis it maintains – the set of

Initialize: $\bar{w}_1^1 = \bar{w}_2^1 = \cdots = \bar{w}_k^1 = 0$
Loop: For $t = 1, 2, \ldots, T$

- Get a new instance $\bar{x}^t \in \mathbb{R}^n$.
- Compute $\bar{w}_1^t \cdot \bar{x}^t, \ldots, \bar{w}_k^t \cdot \bar{x}^t$.
- Set predicted ranking: $\mathcal{R}^t = (rank(\bar{x}^t, 1), \ldots, rank(\bar{x}^t, k))$
- Get feedback: $y^t \subset \mathcal{Y}$.
- Compute loss: $\ell^t = loss(y^t, \mathcal{R}^t)$.
- If $\ell^t > 0$ update the prototypes (otherwise $\forall p \;\; \bar{w}_p^{t+1} = \bar{w}_p^t$):
  1. Set $E^t = \{(r,s) \mid r \in y^t,\; s \notin y^t,\; \bar{w}_r^t \cdot \bar{x}^t \leq \bar{w}_s^t \cdot \bar{x}^t\}$
  2. Form a set of parameters $\alpha_{r,s}^t$ that satisfies the following:
     (a) $\alpha_{r,s}^t \geq 0$.
     (b) If $(r,s) \notin E^t$ then $\alpha_{r,s}^t = 0$.
     (c) $\sum_{r \in y} \sum_{s \notin y} \alpha_{r,s}^t = 1$.
  3. Set for $p = 1, 2, \ldots, k$:
  $$\tau_p^t = \begin{cases} \ell^t \sum_{s \notin y} \alpha_{p,s}^t & p \in y^t \\ \\ -\ell^t \sum_{r \in y} \alpha_{r,p}^t & p \in \mathcal{Y} - y^t \end{cases}$$
  4. For $p = 1, 2, \ldots, k$ update:
  $$\bar{w}_p^{t+1} \leftarrow \bar{w}_p^t + \tau_p^t \bar{x}^t$$

Output : $H(\bar{x}) =$ Topic ranking function of $\mathcal{Y}$ using $\{\bar{w}_p^{T+1} \cdot \bar{x}\}$.

Figure 2: Pseudocode of the MMP algorithm for topic ranking.

prototypes $\bar{w}_1, \ldots, \bar{w}_k$. Online algorithms become especially handy when the training corpus is very large, since they require minimal amounts of memory. In the case of batch training (i.e. when the training corpus is provided in advance and not example by example) we need to augment the online algorithm with a wrapper. Several approaches have been proposed for adapting online algorithms for batch settings. A detailed discussion is given in (Freund and Schapire, 1998). The approach we take in this paper is the simplest to implement. We run the algorithm in an online fashion on the provided training corpus and use the *final* set of topic prototypes, which is obtained after a single pass through the data. This set is used to rank the topics of new documents. We do however describe the results of experiments that cycle through the data multiple times. More sophisticated prediction schemes such as voting and averaging with inner-product kernels (Freund and Schapire, 1998) were not tested in this study due to the vast amounts of memory that such schemes would require for large corpora such as RCV1. In the following description, we omit the index t of the document and its set of relevant topics and denote them as $\bar{x}$ and y, respectively.

The core of MMP is a parametric family of possible updates of the prototypes it maintains. Rather than describing a few possible variants of update schemes, we first describe the parametric form of the update in general by casting a rather flexible set of constraints. We later discuss possible realizations that satisfy these constraints and report experiments with a different update schemes for the prototypes. We also do not need to specify a particular topic-ranking loss for the algorithm. As we show in our analysis, it suffices to assume that the loss suffered by an imperfect ranking is bounded. For concreteness and simplicity, let us assume for now that we suffer a unit loss on each document for which the predicted ranking is not perfect (and otherwise the loss is zero).

The error-set defined in Equation (2) plays a major role in updating the prototypes. Generalizing the Perceptron's update which moves the (single) separating hyperplane toward the example $\bar{x}$, we would like in our setting to move each prototype representing a relevant topic toward $\bar{x}$ and, analogously, each prototype representing a non-relevant topic away from $\bar{x}$. However, there might be quite a few topics which are ranked correctly. These topics include all of the relevant topics that are ranked at the top of the list of topics (above all of the non-relevant topics) and the non-relevant topics ranked at the bottom of the list (below any relevant topic). By definition, the indices of wrongly ordered pairs of topics constitute the error-set. For each pair (r,s) of relevant and non-relevant topics in the error-set, we assign a weight denoted $\alpha_{r,s}$. We set the weights $\alpha_{r,s}$ for topic-pairs not in E to zero. We impose two, rather general, constraints on $\alpha_{r,s}$. The first requires that each $\alpha_{r,s}$ is non-negative. The second constraint confines the total sum of $\alpha_{r,s}$ to be 1. Given a specific set of values that satisfy the constraints, we now define the amount by which the prototype of each relevant topic in E is moved toward $\bar{x}$ and analogously the amount by which each non-relevant topic in E is moved away from $\bar{x}$. For a relevant topic indexed r $(r \in y)$ we define τ_r to be $\sum_{s \notin y} \alpha_{r,s}$ and add to $\bar{w}_r$ the input instance $\bar{x}$ scaled by τ_r: $\bar{w}_r \leftarrow \bar{w}_r + \tau_r \bar{x}$. Therefore, relevant topics whose predicted rank is low are likely to be moved more aggressively toward $\bar{x}$ compared to relevant topics whose predicted rank is relatively high for which there are only a few non-relevant topics listed above them. Similarly, the non-relevant topics are pushed away from $\bar{x}$ in different proportions, depending on how high they are (wrongly) ranked. The general form of this update increases the value of inner-products between $\bar{x}$ and each of the prototypes in the subset of relevant topics in E and similarly decreases the inner-product values between $\bar{x}$ and the non-relevant topics in E.

To further illustrate the update, let us a look at a specific realization of the values $\alpha_{r,s}$ which satisfy the constraints. Concretely, let us assume that all the values are equal. We can therefore assume that $\alpha_{r,s} = 1$ for $(r,s) \in E$ and at the end scale all the values of τ_r by a constant. In this case, the amount by which we move each prototype directly depends on its location in the ranked list. The value τ_r of a relevant topic is equal to the number of non-relevant topics ranked above it and the value $-\tau_s$ of a non-relevant topic s is equal to the number of relevant topics ranked below topic s. Finally, the normalization of the values $\alpha_{r,s}$ ensures that total amount by which we change to the prototypes in terms of $\bar{x}$ is fixed and does not differ between examples.

It remains to describe the more general case when the loss on each round is not constant and depends on the quality of the predicted ranking. The modification for general bounded losses is rather simple and intuitive. We multiply each of the values τ_r and τ_s by the instantaneous loss ℓ^t. Therefore, the total amount by which we modify the prototypes depends on how good (or bad) the predicted ranking is. We modify the prototypes more aggressively on rounds on which the predicted rankings are rather poor. The pseudo code describing the family of topic algorithms for bounded ranking-losses is given in Fig. 2. We next discuss the formal properties of the algorithm.

5. Analysis

In Section 7 we describe a few possible ranking-losses we experimented with. To analyze the algorithm in the mistake bound model it suffices however to assume that the loss suffered on each round with imperfect predicted ranking is bounded above by a constant A. We later provide a more refined analysis for one of the losses we experimented with. The following theorem states that the cumulative ranking loss MMP suffers is bounded. Concretely, it is inversely proportional to the generalized notion of margin as given in Equation (1) and proportional to radius of the sphere enclosing all the examples. This type of result is common to Perceptron-like algorithms and implies that large ranking-margin yields good performance. We would like to note in passing that using the technique of Helmbold and Warmuth (1995) it is possible to derive a generalization bound on the performance of the algorithms from the online mistake bound below.

Theorem 2 (Mistake bound) *Let $(\bar{x}^1, y^1) \ldots (\bar{x}^T, y^T)$ be an input sequence for MMP where $\bar{x}^t \in \mathbb{R}^n$ and $y^t \in \{1 \ldots k\}$. Denote by $B = \max_t \|\bar{x}^t\|$. Assume that there exists a set of prototypes $\bar{w}_1^*, \ldots, \bar{w}_k^*$ of a unit norm ($\sum_r (\bar{w}_r^*)^2 = 1$) whose predicted rankings on entire sequence are all perfect with a margin,*

$$\gamma = \min_t \left\{ \min_{r \in y^t} \{\bar{w}_r^* \cdot \bar{x}^t\} - \max_{s \in \mathcal{Y} - y^t} \{\bar{w}_s^* \cdot \bar{x}^t\} \right\} > 0 \ .$$

If MMP is run with a loss function such that for all t, $\ell^t \leq A$ then its cumulative loss is bounded by,

$$\sum_{t=1}^{T} \ell^t \leq 2A \frac{B^2}{\gamma^2} \ .$$

Proof Let us fix an example $(\bar{x}^t, y^t)$ which the algorithm received on round t. By construction we have that $\bar{w}_p^{t+1} = \bar{w}_p^t + \tau_p^t \bar{x}^t$ for $p = 1, \ldots, k$. We compute the mistake bound of the algorithm by bounding the term $\sum_p \|\bar{w}_p^{T+1}\|^2$ from above and below. We first compute the lower bound by bounding the quantity $\sum_p \bar{w}_p^{t+1} \cdot \bar{w}_p^*$. If no error occurred on the tth round then $\bar{w}_p^{t+1} = \bar{w}_p^t$ for $p = 1, \ldots, k$, and therefore we have that $\sum_p \bar{w}_p^{t+1} \cdot \bar{w}_p^* = \sum_p \bar{w}_p^t \cdot \bar{w}_p^*$. We can therefore assume that MMP ranking was imperfect and thus the set $E^t = \{(r,s) \in y^t \times (\mathcal{Y} - y^t) \ : \ \bar{w}_r^t \cdot \bar{x}^t \leq \bar{w}_s^t \cdot \bar{x}^t\}$ is not empty. By definition we also have that,

$$\sum_p \bar{w}_p^{t+1} \cdot \bar{w}_p^* = \sum_p \bar{w}_p^t \cdot \bar{w}_p^* + \sum_p \tau_p^t \left(\bar{w}_p^* \cdot \bar{x}^t\right) \ . \tag{3}$$

We further bound the right term of Equation (3) from below. We substitute the values of the coefficients τ_p^t using the values of τ_r^t and τ_s^t as defined in Fig. 2 and get,

$$\begin{aligned}
\sum_p \tau_p^t \left(\bar{w}_p^* \cdot \bar{x}^t\right) &= \sum_{r \in y^t} \left[\ell^t \sum_{s \notin y^t} \alpha_{r,s}^t\right] \left(\bar{w}_r^* \cdot \bar{x}^t\right) + \sum_{s \in \mathcal{Y} - y^t} \left[-\ell^t \sum_{r \in y} \alpha_{r,s}^t\right] \left(\bar{w}_s^* \cdot \bar{x}^t\right) \\
&= \ell^t \sum_{r \in y^t} \sum_{s \notin y^t} \alpha_{r,s}^t \left(\bar{w}_r^* \cdot \bar{x}^t\right) - \ell^t \sum_{r \in y} \sum_{s \in \mathcal{Y} - y^t} \alpha_{r,s}^t \left(\bar{w}_s^* \cdot \bar{x}^t\right) \\
&= \ell^t \sum_{r \in y^t} \sum_{s \in \mathcal{Y} - y^t} \alpha_{r,s}^t \left(\bar{w}_r^* \cdot \bar{x}^t - \bar{w}_s^* \cdot \bar{x}^t\right) \ . \qquad (4)
\end{aligned}$$

Given the assumption that $\bar{w}_r^* \cdot \vec{x}^t - \bar{w}_s^* \cdot \vec{x}^t \geq \gamma$ for $r \in y^t$ and $s \in \mathcal{Y} - y^t$ we get,

$$\begin{aligned} \sum_p \tau_p^t \left(\bar{w}_p^* \cdot \vec{x}^t\right) &\geq \ell^t \sum_{r \in y^t} \sum_{s \in \mathcal{Y} - y^t} \alpha_{r,s}^t \gamma \\ &= \gamma \ell^t \sum_{r \in y^t} \sum_{s \in \mathcal{Y} - y^t} \alpha_{r,s}^t \ . \end{aligned}$$

Using the third constraint imposed by MMP ($\sum_{r,s} \alpha_{r,s}^t = 1$) we finally get,

$$\sum_p \tau_p^t \left(\bar{w}_p^* \cdot \vec{x}^t\right) \geq \gamma \ell^t \ . \tag{5}$$

Unfolding the sum over t of Equation (3) using Equation (5) we have,

$$\sum_p \bar{w}_p^{T+1} \cdot \bar{w}_p^* \geq \gamma \sum_{t=1}^{T} \ell^t \ .$$

Applying Cauchy-Schwartz inequality and using the assumption $\sum_r \left(\bar{w}_r^*\right)^2 = 1$ we get the desired lower bound,

$$\sum_p \|\bar{w}_p^{T+1}\|^2 \geq \gamma^2 \left(\sum_{t=1}^{T} \ell^t\right)^2 \ . \tag{6}$$

Next, we bound the term $\sum_p \|\bar{w}_p^{T+1}\|^2$ from above. As before, assume that the topics of example $(\vec{x}^t, y^t)$ were ordered using the set of prototypes $(\bar{w}_1^t, \ldots, \bar{w}_k^t)$ and denote by $(\bar{w}_1^{t+1}, \ldots, \bar{w}_k^{t+1})$ the rule after round t. Again, if the predicted ranking is perfect then $\bar{w}_p^{t+1} = \bar{w}_p^t$ for $p = 1, \ldots, k$. We thus assume again that the predicted ranking is imperfect. We now rewrite the term $\sum_p \|\bar{w}_p^{t+1}\|^2$ in a more explicit form.

$$\begin{aligned} \sum_p \|\bar{w}_p^{t+1}\|^2 &= \sum_p \|\bar{w}_p^t + \tau_p^t \vec{x}^t\|^2 \\ &= \sum_p \|\bar{w}_p^t\|^2 + 2 \sum_p \tau_p^t \left(\bar{w}_p^t \cdot \vec{x}^t\right) + \sum_p \left(\tau_p^t\right)^2 \|\vec{x}^t\|^2 \end{aligned} \tag{7}$$

We first bound the middle term of Equation (7) from above. Using the same derivation as of Equation (4) we get,

$$\sum_p \tau_p^t \left(\bar{w}_p^t \cdot \vec{x}^t\right) = \ell^t \sum_{r \in y^t} \sum_{s \in \mathcal{Y} - y^t} \alpha_{r,s}^t \left(\bar{w}_r^t \cdot \vec{x}^t - \bar{w}_s^t \cdot \vec{x}^t\right) \ .$$

From the MMP's first constraint on $\alpha_{r,s}^t$ we have $\alpha_{r,s}^t \geq 0$ and from the second constraint we get that $\alpha_{r,s}^t = 0$ if $\bar{w}_r^t \cdot \vec{x}^t - \bar{w}_s^t \cdot \vec{x}^t \geq 0$. Since $\ell^t \geq 0$, we can bound the middle term from the right hand side of Equation (7) by,

$$\sum_p \tau_p^t \left(\bar{w}_p^t \cdot \vec{x}^t\right) \leq 0 \ . \tag{8}$$

For the right term of Equation (7) we use the bound on the losses $\ell^t \leq A$ and that $\sum_{s \notin y^t} \alpha_{r,s}^t, \sum_{r \in y} \alpha_{r,s}^t \leq 1$ (due to the first and third conditions) and get

$$\sum_p \left(\tau_p^t\right)^2 = \sum_{r \in y^t} \left(\tau_r^t\right)^2 + \sum_{s \in \mathcal{Y} - y^t} \left(\tau_s^t\right)^2$$

$$
\begin{aligned}
&= \sum_{r \in y^t} \left(\ell^t \sum_{s \notin y^t} \alpha_{r,s}^t \right)^2 + \sum_{s \in \mathcal{Y} - y^t} \left(\ell^t \sum_{r \in y} \alpha_{r,s}^t \right)^2 \\
&\leq A \ell^t \left[\sum_{r \in y^t} \left(\sum_{s \notin y^t} \alpha_{r,s}^t \right) + \sum_{s \in \mathcal{Y} - y^t} \left(\sum_{r \in y} \alpha_{r,s}^t \right) \right] . \qquad (9)
\end{aligned}
$$

Using the third constraint of the algorithm described in Fig. 2 we get,

$$\sum_p \left(\tau_p^t\right)^2 \leq A\,\ell^t (1+1) = 2A\,\ell^t . \qquad (10)$$

Using the bounds from Equations (10) and (8) in Equation (7) with the bound $\|\bar{x}^t\|^2 \leq B^2$ we get,

$$\sum_p \|\bar{w}_p^{t+1}\|^2 \quad \leq \quad \sum_p \|\bar{w}_p^t\|^2 + 2A\,B^2\,\ell^t .$$

Unfolding the sum over t we get the desired upper bound,

$$\sum_p \|\bar{w}_p^{T+1}\|^2 \leq 2A\,B^2 \sum_{t=1}^{T} \ell^t . \qquad (11)$$

Finally, combining Equations (6) and (11) we get,

$$\gamma^2 \left(\sum_{t=1}^{T} \ell^t \right)^2 \leq \sum_p \|\bar{w}_p^{T+1}\|^2 \leq 2A\,B^2 \sum_{t=1}^{T} \ell^t ,$$

which gives the desired bound. ■

To conclude this section, we would like to note that in the simple case where the instantaneous loss is either zero or one. For this simple loss, the ranking-loss bound of the theorem above reduces to a bound on the number of rounds on which the predicted rank was not perfect, which is simply $2B^2/\gamma^2$. We refer below to this type of loss as IsErr.

We would also like to note that when all the examples are uni-labeled then using MMP with IsErr as its loss results in an update that is equivalent to the additive algorithm described in (Crammer and Singer, 2001b).

6. Prototype Update Schemes

The family of additive online algorithm prescribes a set of constraints. Any *conservative* update that satisfies the constraints attains the mistake bound given in Thm. 2. We now describe four different schemes for updating the parameters that satisfy the constraints. Experiments with the different updates are reported in Section 9.

Uniform Update: The uniform update allocates the same weight for each wrongly ordered pair $(r,s) \in E$, setting $\alpha_{r,s} = 1/|E|$. In addition, we need to set the weights of all pairs not in the error to 0. Clearly, this update satisfies the constraints on $\alpha_{r,s}$. Note that although each pair $(r,s) \in E$ is allocated the same weight of $1/|E|$, the weights τ_r are different. Qualitatively, the lower a relevant topic r is ranked, the more it will be "pushed" toward $\bar{x}$ by setting τ_r to a large value. Analogously, a highly ranked topic s which is not relevant will be "pulled" away from $\bar{x}$ strongly since τ_s is going to be negative with a large absolute value. We use the abbreviation Uniform for this update.

Loss	Range	Bound
IsErr	$\{0,1\}$	1
ErrSetSize	$\{0,1,,\ldots,\lvert y\rvert\lvert k-y\rvert\}$	$k^2/4$
OneErr	$\{0,1\}$	1
1-AvgP	$\{0,1/2,\ldots,B(\lvert y\rvert,k)\}$	$(k-1)/k$
maxF_1	$\{0,\ldots,1\}$	1

Table 1: Summary of the losses used in the experiments.

Max Update: In this update, we modify only two prototypes: the prototype corresponding to the lowest ranked relevant topic and the prototype corresponding to the highest ranked non-relevant topic. Formally, let $\tilde{r}$ be the index of the relevant topic for which $\bar{w}_{\tilde{r}} \cdot \bar{x} \leq \bar{w}_r \cdot \bar{x}$ ($r \in y$) and similarly $\tilde{s}$ is the index of the non-relevant topic $\bar{w}_{\tilde{s}} \cdot \bar{x} \geq \bar{w}_s \cdot \bar{x}$ ($s \notin y$). Then, $\alpha_{\tilde{r},\tilde{s}}$ is set to 1 and we set the rest of the weights $\alpha_{r,s}$ to zero. We therefore concentrate on a single pair from E. Thus, $\tau_{\tilde{r}} = 1$ and $\tau_{\tilde{s}} = -1$ and we update only two prototypes. We use the abbreviation Max for this update.

Margin-proportional Update: This update is a variation on the uniform update that takes the ranking margin into account. In this update, the weight $\alpha_{r,s}$ is proportional to the difference between the inner-products of $\bar{x}$ with $\bar{w}_r$ and $\bar{w}_s$. Let $[z]_+$ denote the hinge function which equals to the argument z if $z > 0$ and is zero otherwise, $[z]_+ = \max\{z, 0\}$. Then, the margin proportional update is defined to be,

$$\alpha^t_{r,s} = \frac{[\bar{w}^t_s \cdot \bar{x} - \bar{w}^t_r \cdot \bar{x}]_+}{\sum_{(p,q)\in y\times\mathcal{Y}-y}[\bar{w}^t_p \cdot \bar{x} - \bar{w}^t_q \cdot \bar{x}]_+} \; .$$

On the first round we have by construction that $\bar{w}^t_p \cdot \bar{x} = 0$ and thus the value of $\alpha_{r,s}$ is ill-defined. We therefore set $\alpha_{r,s}$ to some arbitrary feasible values for the first round of the algorithm.

Randomized Update: This update is a randomized version of the uniform update. In this update, we pick at random a vector from the simplex $\Delta_{|E|}$ as follows. We choose for each pair $(r,s) \in E$ a value in $[0,1]$ uniformly at random. We then normalize the weights $\alpha_{r,s}$ so that they sum to one. The results is a distribution over the simplex $\Delta_{|E|}$ that is dense in the middle of the simplex and gets sparser as we move toward the vertices of the simplex.

7. Ranking losses

We have implemented and evaluated the MMP algorithm with five different ranking-losses. The ranking-losses we employed constitute some of the common ranking-losses used in machine learning and information retrieval. A topic ranking algorithm that performs well with respect to all of the ranking-losses is likely to be effective in various applications requiring ranking of topics. We describe below the ranking-losses we used. All of the losses are additive and thus each loss is described for a single document. In assessing the performance on a whole corpus, we simply compute the empirical expectation of each loss.

IsErr stands for Is-Error. This ranking-loss is simply the indicator of whether the induced ranking is perfect or not. Formally, IsErr is 1 if $|E| > 0$ and 0 otherwise. IsErr is one of the commonly used performance criteria in the analysis of learning algorithms.

ErrSetSize measures the size of the error-set $|E|$ for an induced ranking. It is therefore zero if the induced topic ranking is perfect. In the worst case, when all the relevant topics are ranked *below* the non-relevant topics, ErrSetSize is $|y|(k - |y|)$ and thus ErrSetSize can be as large as $k^2/4$.

OneErr is an abbreviation for the term one-error introduced by Schapire and Singer (1999). The OneErr of a single document is defined to be 1 if the top ranked topic is *not* one of its relevant topics. It is defined to be 0 otherwise. Formally, OneErr is equal to 1 if $\forall r \in y \ : \ rank(\bar{x}, r) \geq 1$ and is 0 otherwise. Therefore, the average OneErr of a corpus reflects the fraction of times the top ranked topic was a non-relevant topic.

AvgP is an abbreviation of average-precision. AvgP is a commonly used ranking-loss in information retrieval tasks such as the ad hoc document retrieval. AvgP measures the average proportion of relevant topics in a ranked list of topics. The average is taken over all of the positions of the relevant topics in the ranked list and is formally defined as,

$$\text{AvgP} = \frac{1}{|y|} \sum_{r \in y} \frac{|\{r' \in y \ : \ rank(\bar{x}, r') \leq rank(\bar{x}, r)\}|}{rank(\bar{x}, r)} \ .$$

A perfect ranking thus attains an AvgP of 1. Since in our evaluation we employ both learning-theoretic and IR ranking-losses, we would like to use the same range for all ranking-losses. Therefore, we use $1 - \text{AvgP}$ as the ranking-loss for average precision since it attains a value of 0 for a perfect ranking. However, in reporting performance results we use the more intuitive and common measure AvgP. Whenever it is clear from the context, we also refer to $1 - \text{AvgP}$ as the average-precision. For a ranking rule which wrongly ranks all the relevant topics below the non-relevant ones $1 - \text{AvgP}$ can be as large as

$$B(|y|, k) \stackrel{\text{def}}{=} 1 - \sum_{i=1}^{|y|} \frac{i}{k - |y| + i} \ .$$

The largest value of $B(|y|, k)$, and thus the largest loss for AvgP, is attained when there is a single relevant topic that is wrongly ranked at the bottom of the list. In this case the value of $1 - \text{AvgP} = B(1, k)$ is $1 - \frac{1}{k} = \frac{k-1}{k}$.

maxF_1 is derived from the F_1 ranking-loss which is also common in performance evaluations of IR tasks. Given a ranked list of topics, recall-at-r is defined as the fraction of relevant topics down to position r out of the total number of relevant topics for the document. Precision-at-r is the fraction of relevant topics in the top r positions. We denote the two values as $Recall(r)$ and $Precision(r)$. The F_1 value at r is defined as,

$$F_1(r) = \frac{2\,\text{Recall}(r)\,\text{Precision}(r)}{\text{Recall}(r) + \text{Precision}(r)} \ .$$

For more information and further motivation for using F_1 as a ranking-loss, see (van Rijsbergen, 1979). The maxF_1 is defined as the maximum over r of the values that F_1 can attain. As in the case of average precision, we use $1 - \max F_1$ as the ranking-loss employed in training.

The properties of the various losses are summarized in Table 1. For each each ranking-loss, we give its range and an attainable upper bound on its value.

8. Refined Analysis for ErrSetSize

To conclude the formal part of the paper, we give a refined analysis for the case when the ranking-loss is ErrSetSize with the Uniform update. Examining Table 7, we see that the maximum loss for ErrSetSize is $k^2/4$. Therefore, using Thm. 2 straightforwardly we get a bound which is quadratic in the number of topics. We now show how to improve the bound to be linear in the number of topics. We follow the same line of proof until we get to Equation (9). We now modify the proof as follows. Note that for the specific choices of ranking-loss and update scheme we have that $\ell^t = |E^t|$. In addition, the sum $\sum_{s\notin y^t} \alpha^t_{r,s}$ is proportional to the number of non-relevant topics ($s \notin y^t$) which are ranked higher than the relevant topic indexed r. Assuming that for each document there is at least one relevant topic and one non-relevant topic, the above sum is bounded by,

$$\sum_{s\notin y^t} \alpha^t_{r,s} \leq \frac{k-1}{|E^t|} = \frac{k-1}{\ell^t} \ . \tag{12}$$

A similar argument holds for the sum $\sum_{r\in y} \alpha^t_{r,s}$ and we thus get,

$$\sum_{r\in y} \alpha^t_{r,s} \leq \frac{k-1}{\ell^t} \ . \tag{13}$$

We now replace Equation (9) with,

$$\begin{aligned}
\sum_p \left(\tau^t_p\right)^2 &= \sum_{r\in y^t} \left(\tau^t_r\right)^2 + \sum_{s\in\mathcal{Y}-y^t} \left(\tau^t_s\right)^2 \\
&= \sum_{r\in y^t} \left(\ell^t \sum_{s\notin y^t} \alpha^t_{r,s}\right)^2 + \sum_{s\in\mathcal{Y}-y^t} \left(\ell^t \sum_{r\in y} \alpha^t_{r,s}\right)^2 \\
&= \sum_{r\in y^t} \left(\ell^t \sum_{s\notin y^t} \alpha^t_{r,s}\right)\left(\ell^t \sum_{s\notin y^t} \alpha^t_{r,s}\right) + \sum_{s\in\mathcal{Y}-y^t} \left(\ell^t \sum_{r\in y} \alpha^t_{r,s}\right)\left(\ell^t \sum_{r\in y} \alpha^t_{r,s}\right)
\end{aligned}$$

Using Equations (12) and (13) we get

$$\begin{aligned}
\sum_p \left(\tau^t_p\right)^2 &\leq \sum_{r\in y^t} (k-1) \left(\ell^t \sum_{s\notin y^t} \alpha^t_{r,s}\right) + \sum_{s\in\mathcal{Y}-y^t} (k-1) \left(\ell^t \sum_{r\in y} \alpha^t_{r,s}\right) \\
&= (k-1)\ell^t \left[\sum_{r\in y^t} \left(\sum_{s\notin y^t} \alpha^t_{r,s}\right) + \sum_{s\in\mathcal{Y}-y^t} \left(\sum_{r\in y} \alpha^t_{r,s}\right)\right] \\
&= 2(k-1)\ell^t \ ,
\end{aligned}$$

where the last equality follows from the third condition ($\sum_{r,s} \alpha^t_{r,s} = 1$) of the algorithm. We thus have replaced Equation (10) with a tighter bound of $2(k-1)\ell^t$. We now proceed with the rest of the proof as before and get the following corollary.

Corollary 3 *Let $(\bar{x}^1, y^1) \ldots (\bar{x}^T, y^T)$ be an input sequence for MMP where $\bar{x}^t \in \mathbb{R}^n$ and $y^t \in \{1 \ldots k\}$. Denote by $B = \max_t \|\bar{x}^t\|$. Assume that there exists a set of prototypes $\bar{w}^*_1, \ldots, \bar{w}^*_k$ of a unit norm ($\sum_r (\bar{w}^*_r)^2 = 1$) whose predicted rankings the on entire sequence are all perfect with a ranking margin*

$$\gamma = \min_t \left\{ \min_{r\in y^t} \{\bar{w}^*_r \cdot \bar{x}^t\} - \max_{s\in\mathcal{Y}-y^t} \{\bar{w}^*_s \cdot \bar{x}^t\} \right\} > 0 \ .$$

	Reuters-21578 (average)	Reuters-2000 Subset 1	Subset 2	Subset 3	Full Set
Train Set Size	8,631.2	639	5,139	50,139	521,439
Test Set Size (used)	2,157.8	10,000	10,000	10,000	287,944
No. of classes	89.8	102	102	102	102
Average topics per document	1.24	3.11	3.16	3.12	3.20
No. of features					
Before feature selection	25,061	225,201	225,324	225,324	225,329
After feature selection	3,468.2	4,529	4,174	5,015	9,325
Average no. of unique Terms					
Before feature selection	49.2	151.9	131.7	137.0	137.0
After feature selection	36.7	111.9	95.0	108.6	121.2

Table 2: Summary of properties of the datasets used in the experiments. For Reuters-21578 the numbers are averaged over five folds of the whole dataset while for Reuters release 2000 we used a fixed split into a training set and a test set (see text).

If MMP is used with ErrSetSize and the Uniform update, then its cumulative loss is bounded by,

$$\sum_{t=1}^{T} \ell^t \leq 2(k-1)\frac{B^2}{\gamma^2} \ .$$

9. Experiments

In this section we describe the experiments we performed that compare the above updates of MMP with various ranking-losses. For comparison purposes we also implemented and evaluated an adaptation of the Perceptron algorithms for multi-labeled documents and a multi-label version of Rocchio's algorithm (Rocchio, 1971). We start with a description of the datasets used in our experiments.

9.1 Datasets

We evaluated the algorithms on two text corpora. Both corpora were provided by Reuters.

Reuters-21578: The documents in this corpus were collected from the Reuters newswire during 1987, and are available from *http://www.daviddlewis.com/resources/testcollections* . We used the ModApte pre-processing of the corpus and further processed the documents as follows. All words were converted to lower-case, digits were mapped to a single token designating it is a digit, and non alpha-numeric characters were discarded. We also used a stop-list to remove very frequent words. The number of different words left after pre-processing is 27,747. After the pre-processing, the corpus contains 10,789 documents each of which is associated with *one* or more topics. The number of different topics in the ModApte version of Reuters-21578 is 91. Since this corpus is of relatively small size, we used 5-fold cross validation in our experiments and did not use the original

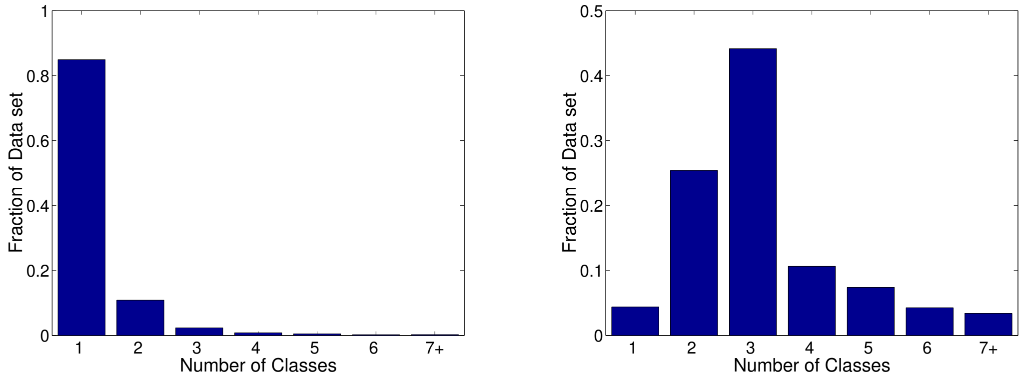

Figure 3: The distribution of the number of relevant topics per document in the Reuters-21578 corpus (left) and in Reuters-2000 corpus (right).

partition into a training set and a test set. While each document in the Reuters-21578 corpus can be multi-labeled, in practice the number of such documents is relatively small. Over 90% of the documents are associated with a single topic. On the left hand side of Fig. 3, we show the empirical distribution of the number of relevant topics. The average number of relevant topics per document is 1.24. A summary of the properties of the Reuters-21578 dataset appears in Table 2.

Reuters-2000: This corpus contains $809,383$ documents collected from the Reuters newswire in a year period (1996-08-20 through 1997-08-19). Since this corpus is large, we used the first two-thirds of the corpus for training and the remaining third for evaluation. The training set consisted of all documents that were posted from 1996-08-20 through 1997-04-10, resulting in 521,439 training documents. The size of the corpus which was used for evaluation is $287,944$. We pre-processed the corpus as follows. We converted all upper-case characters to lower-case, replaced all non alpha-numeric characters with white-spaces, and discarded all the words appearing only once in the training set. The number of different words that remained after this pre-processing is 225,329. Each document in the collection is associated with *zero* or more topic codes. There are 103 different topics in the entire corpus, however, only 102 of them appear in the training set. The remaining topic marked GMIL (for millennium issues) appears as a relevant topic in only 5 documents in the test set. We therefore discarded this topic. Each document in the corpus is also tagged by multiple topics with a much larger overlap between the topics than the Reuters-21578 corpus. About 70% of the documents are associated with at least three different topics. The average number of topics associated with each document is 3.2. The distribution of the number of relevant topics per document appears on the right hand-side of Fig. 3. Since the training set is large, we also evaluated the algorithms on subsets of the training set. We picked subsets of size 500, 5000 and 50000 from the entire training set. We then added to each subset a pre-selected set of 139 documents from the tail of the full training set. This construction ensures that each topic appears at least twice in each of the reduced training sets. It also ensures that all the training sets share a common subset (prefix) of documents. We are thus able to compare the online results for the different training sets on the common subset. We refer to these training sets as Subset 1,2 and 3, respectively. A summary of the properties of the dataset is given in Table 2.

9.2 Documents Representation

All the algorithms we evaluated use the same document representation. We implemented the pivoted length normalization of Singhal et al. (1996) as our term-weighting algorithm. This algorithm is considered as one of the most effective algorithms for document ranking and retrieval.

We now outline the pivoted length normalization. Let d_i^l denote the number of times a word (or term) indexed l appears in the document indexed i. Let m_i denote the number of unique words that appear in the document indexed i, $m_i = |\{l \ : \ d_i^l > 0\}|$. Let r_l be the number of documents in which the term indexed l appears. As before, the total number of documents in the corpus is denoted by m. Using these definitions, the inverse document frequency (idf) of the word indexed l is $\mathrm{idf}_l = \log(m/r_l)$. The average frequency of the terms appearing in document indexed i is,

$$\hat{d}_i = \frac{\sum_l d_i^l}{m_i} \ ,$$

and the empirical average number of unique terms per-document, denoted $\hat{u}$, is calculated from the entire corpus as follows,

$$\hat{u} = \frac{1}{m} \sum_{i=1}^{m} m_i \ .$$

Using these definitions, the *tf* weight of a word indexed l appearing in the document indexed i is,

$$\mathrm{tf}_l^i = \frac{\left(1 + \log(d_i^l)\right) / \left(1 + \log(\hat{d}_i)\right)}{1.0 - slope + slope \times (m_i/\hat{u})} \ .$$

Here $slope$ is a parameter between 0 and 1. We set $slope = 0.3$, which leads to the best performance in the experiments reported in (Singhal et al., 1996). Finally, the features constituting each document are the products of the tf and idf weights of the words appearing in the document, $x_l^i = \mathrm{idf}_l \times \mathrm{tf}_l^i$.

9.3 Algorithms for Comparison

In addition to the different update schemes of MMP, we also implemented two more algorithms: Rocchio's algorithm (Rocchio, 1971) and the Perceptron algorithm. As with MMP, these algorithms use the same pivoted length normalization as their vector space representation and employ the same form of topic-ranking by using a set of prototypes $\bar{w}_1, \ldots, \bar{w}_k$. Note that, with the exception of Rocchio, all the algorithms we implemented and evaluated are online algorithms.

Rocchio. We implemented an adaptation of Rocchio's method as adapted by Ittner et al. (1995) to text categorization. In this variant of Rocchio, the set of prototype vectors $\bar{w}_1, \ldots, \bar{w}_k$ are calculated as follows,

$$w_r^l \stackrel{\mathrm{def}}{=} \max\left\{0, \frac{\beta}{|R_r|} \sum_{i \in R_r} x_l^i - \frac{\gamma}{|R_r^c|} \sum_{i \in R_r^c} x_l^i \right\} \ ,$$

where R_r is the set of documents which contain the topic r as one of their relevant topics and R_r^c is its complement, i.e., all the documents for which r is not one of their relevant topics. Following the parameterization used by Ittner et al. (1995), we set $\beta = 16$ and $\gamma = 4$. Last, as suggested by Amit Singhal in a private communication, we normalize all of the prototypes to have a unit norm.

Perceptron. Since the Perceptron algorithm is designed for binary classification problems, we decomposed the multi-label problem into multiple binary classification problems. For each topic r, we constructed a separate training set as follows. We labeled all the documents whose indices appear in R_r as positive and the rest of the documents are labeled as negative. We then ran the Perceptron algorithm on each of the binary problems separately and independently. We therefore obtained again a set of prototypes $\bar{w}_1, \ldots, \bar{w}_k$ each of which is an output of the corresponding Perceptron algorithm.

9.4 Feature Selection

For all datasets, the number of unique terms after the pre-processing stage described above was still large: 27,747 words in the complete Reuters-21578 and 225,329 words in the complete training set of Reuters-2000. Since we used cross-validation for Reuters-21578, the actual number of unique terms was slightly lower, 25,061 on the average. Since this number of unique terms was still relatively large, we employed feature selection for both corpora to further reduce this number. We used the weights of the prototypes generated by the adaptation of Rocchio's algorithm described above as our method for feature selection. For each topic, we sorted the terms according to their weights as assigned by Rocchio. We then took for each topic the maximum between a hundred terms and the top portion of 2.5% terms from the list sorted according the scores of Rocchio. This ensures that for each topic we have at least 100 terms. The combined set of selected terms is used as the feature set for the various algorithms and is of size 3,468.2 for Reuters-21578 and 9,325 for the complete training set of Reuters-2000. The average number of unique words per document in the cross-validated training sets was reduced from 49 to 37 for Reuters-21578 and from 137 to 121 for the complete training set of Reuters-2000. After this feature selection stage, we applied all the algorithms to the same representation of documents. A summary of the properties of the Reuters-21578 and the four training sets of Reuters-2000 is given in Table 2.

9.5 Experimental Setup

As we have just discussed, we evaluated the algorithms on five training sets in our experiments: the Reuters-21578 corpus with 5-fold cross-validation and four training sets forming subsets of different sizes of the Reuters-2000 corpus. For each training set, we first generated prototypes by running Rocchio's algorithm with all the features available. We then selected a subset of features as described in Section 9.4. We represented the training set and the test set using the new set of features as described in Section 9.2, and built twenty two different sets of prototypes for each training set we have experimented with: the first set of prototypes was generated by Rocchio, the second set was generated using the modified Perceptron algorithms, and the rest of prototypes were created by MMP with five different ranking-losses (IsErr, ErrSetSize, OneErr, 1-AvgP, 1-maxF_1) where each loss was trained with four different update schemes (Uniform, Max, Prop, and Rand), yielding twenty different sets of prototypes. We then evaluated each of the algorithms using the corresponding test sets (see again Table 2). The evaluation of each of the learned sets of prototypes was performed with respect to *all* of the ranking-losses. As we discuss shortly, by evaluating each variant of MMP not only with respect to the loss it was trained with but also with respect to all the other losses we are able to check whether there exists a ranking-loss that is universally good for training topic-rankers regardless of the actual ranking-loss employed for evaluation. Each of the online algorithms was run five times on the training set. After each epoch we evaluated the resulting set of prototypes on the test set. Since in most cases training the algorithms using more than one

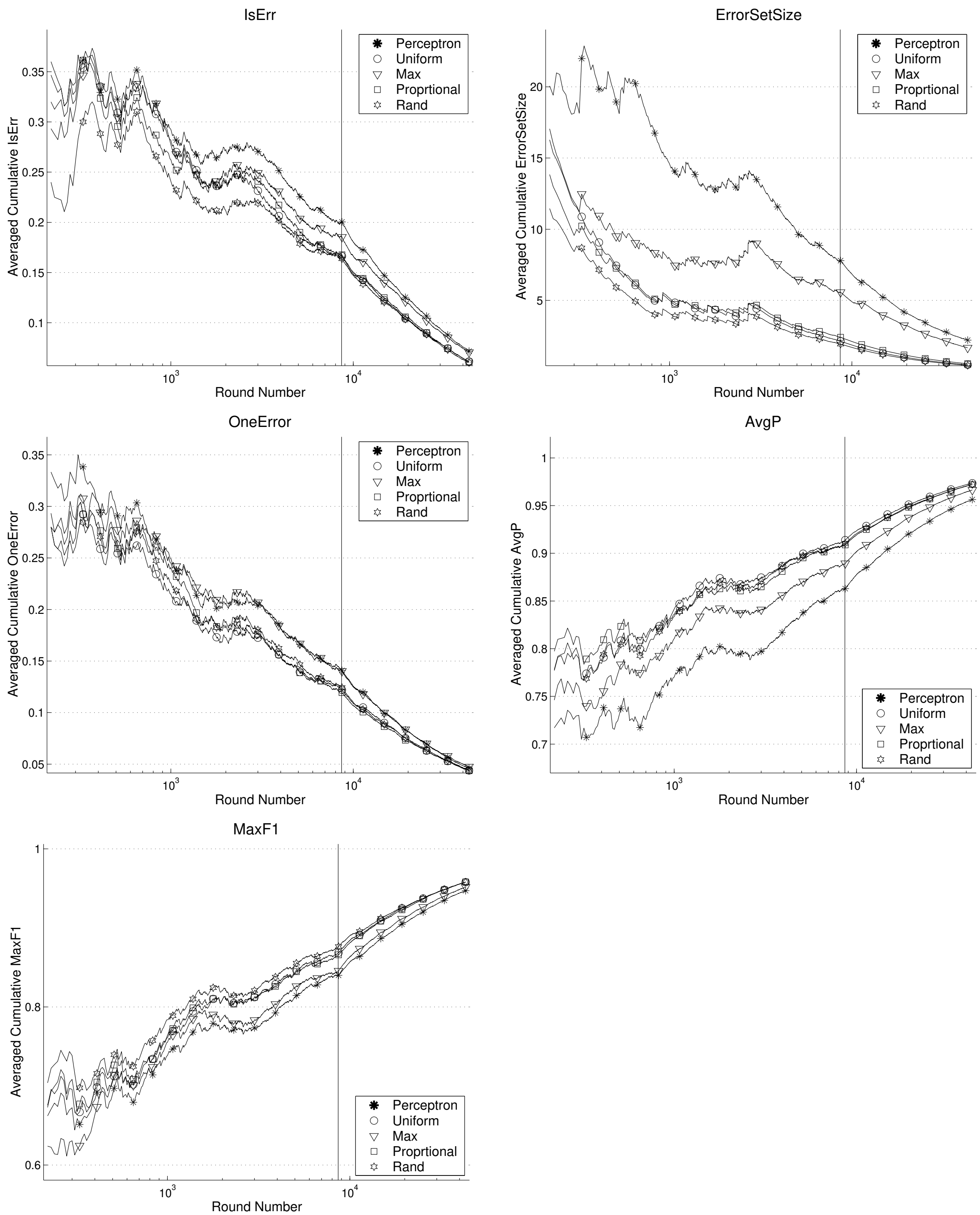

Figure 4: The round-averaged ranking-loss as a function of the number of training documents that were processed for Reuters-21578. The vertical line in each figure indicates the end of the first epoch.

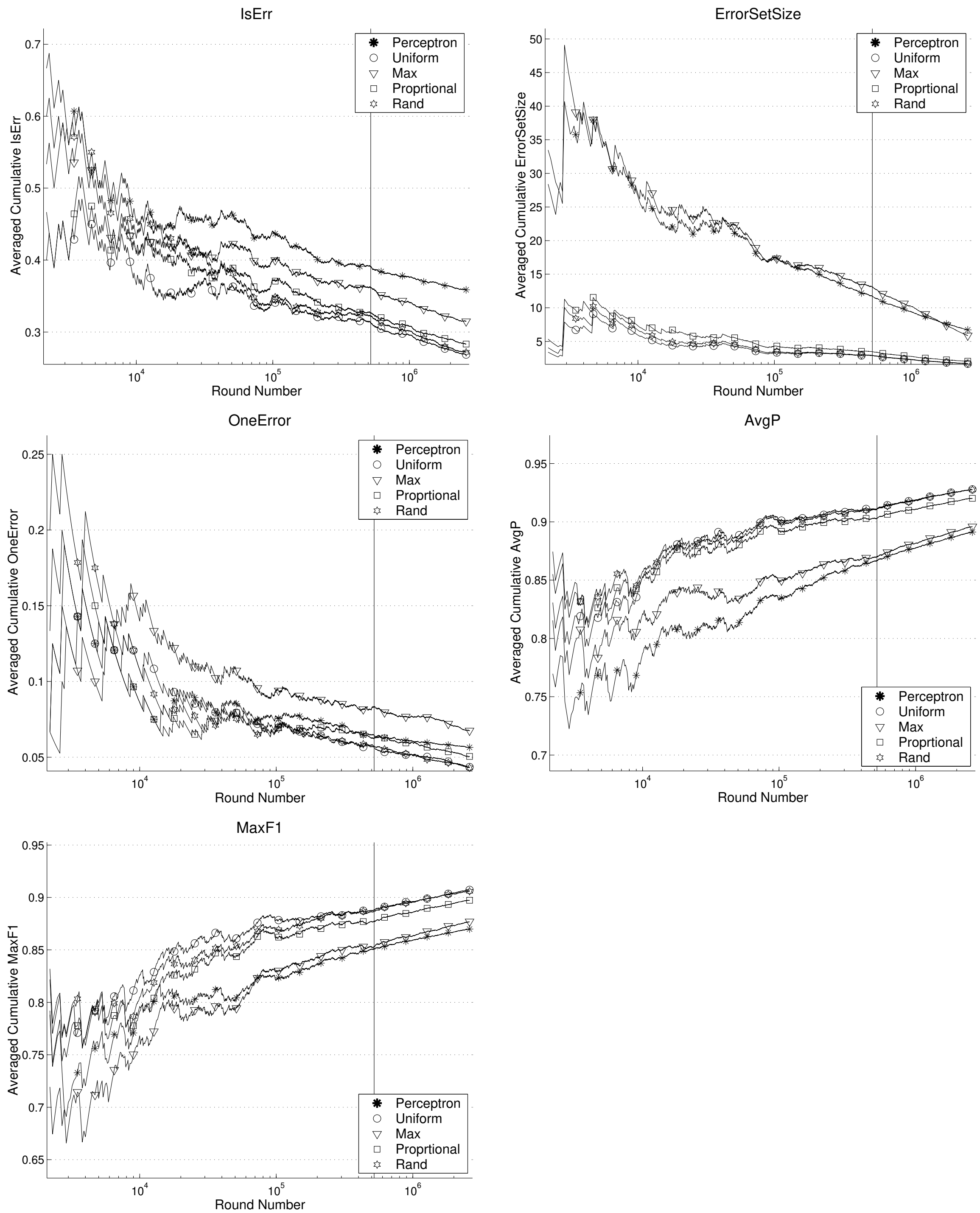

Figure 5: The round-averaged ranking-losses as a function of the number of training documents that were processed for Reuters-2000. The vertical line indicates the end of the first epoch.

Algorithm		Ranking-Loss used for Evaluation				
Training-Loss	Update	IsErr x100	ErrSetSize	OneErr x100	AvgP x100	maxF_1 x100
Rocchio		18.82	1.17	14.48	89.73	84.53
Perceptron		15.71	4.87	9.59	90.50	88.56
Uniform	IsErr	13.07	1.33	8.65	93.63	90.32
	ErrSetSize	17.99	**0.78**	13.42	91.07	85.53
	OneErr	15.14	2.03	8.63	92.79	89.64
	AvgP	12.72	1.12	**8.20**	93.93	90.75
	maxF_1	**12.36**	0.94	8.23	**94.14**	**90.80**
Max	IsErr	14.53	2.60	9.54	92.43	89.17
	ErrSetSize	23.50	3.58	17.65	86.81	80.74
	OneErr	17.16	4.35	9.75	90.90	88.05
	AvgP	**14.14**	**2.50**	**9.03**	**92.58**	**89.59**
	maxF_1	15.02	2.77	10.10	92.04	88.61
Proportional	IsErr	12.88	1.50	8.48	93.60	90.45
	ErrSetSize	20.31	**1.05**	15.56	89.74	83.33
	OneErr	15.63	2.49	8.95	92.30	89.23
	AvgP	**12.83**	1.28	**8.28**	**93.83**	**90.63**
	maxF_1	13.57	1.27	9.10	93.45	89.87
Rand	IsErr	**12.52**	1.31	8.18	93.89	90.81
	ErrSetSize	18.02	**0.80**	14.00	90.99	85.08
	OneErr	14.48	1.99	8.18	93.08	90.12
	AvgP	12.60	0.99	**8.11**	**94.11**	**90.84**
	maxF_1	13.09	1.06	8.56	93.75	90.38

Table 3: A comparison of the performance of the various algorithms on the test-set for different ranking-losses on Reuters-21578.

epoch over the training set yielded only minor improvement, we mostly report results obtained at the end of the first epoch and defer the comparison of the performance with more than a single epoch to the end of the section.

9.6 Results

We start with a comparison of the performance of the online algorithms (all the variants of MMP and the Perceptron) on the *training* data. In Fig. 4 and Fig. 5 we show the performance of MMP and Perceptron with respect to the five ranking-losses on Reuters-21578 and Reuters-2000. One of the goals in comparing the online performances of the algorithms is to assess whether the empirical results agree with the formal analysis. Therefore it is natural to use the loss employed by the update also for evaluation. We thus evaluated each of the variants of MMP only with respected to the loss it employed for updating the prototypes. On each round (new document), we computed the cumulative loss of the algorithms divided by the number of documents processed so far, $\sum_{t=1}^{s} loss(y^t, \mathcal{R}^t)/s$. Each plot in Fig. 4 and Fig. 5 shows the average cumulative ranking-loss as a function the number of examples along five epochs through the data. The vertical line in each plot designates the end of the first epoch. We can see from the figures that the MMP algorithm used with the Uniform update performs well with respect to all of the ranking-losses. The second best update after Uniform is Rand, then Prop, and Max is the worst performer, often lagging significantly behind the other

Algorithm		Ranking-Loss used for Evaluation				
Training-Loss	Update	IsErr x100	ErrSetSize	OneErr x100	AvgP x100	maxF_1 x100
Rocchio		70.71	12.42	24.42	72.62	63.25
Perceptron		38.86	10.43	6.04	87.40	85.65
Uniform	IsErr	$\boxed{\mathbf{30.68}}$	2.87	$\boxed{\mathbf{4.73}}$	$\boxed{\mathbf{92.12}}$	$\boxed{\mathbf{89.70}}$
	ErrSetSize	34.59	$\boxed{\mathbf{2.84}}$	5.81	90.98	87.95
	OneErr	42.14	5.02	5.60	88.95	86.11
	AvgP	32.68	3.02	5.01	91.65	89.02
	maxF_1	31.98	2.94	5.15	91.78	89.10
Max	IsErr	**34.83**	**6.04**	**5.76**	**90.03**	**87.53**
	ErrSetSize	51.36	11.50	11.50	83.34	78.76
	OneErr	53.04	20.57	8.66	81.44	79.29
	AvgP	41.00	9.06	7.29	87.48	84.57
	maxF_1	38.83	8.00	6.73	88.32	85.58
Proportional	IsErr	**31.87**	**3.47**	**5.36**	**91.45**	**88.83**
	ErrSetSize	37.72	3.51	7.29	89.62	85.97
	OneErr	45.36	6.57	6.66	87.50	84.36
	AvgP	35.27	3.80	5.82	90.55	87.65
	maxF_1	34.16	3.71	5.86	90.79	87.89
Rand	IsErr	**30.71**	2.91	**4.88**	**92.08**	**89.58**
	ErrSetSize	34.91	**2.87**	6.03	90.87	87.71
	OneErr	43.33	5.25	5.94	88.58	85.55
	AvgP	32.94	3.05	5.03	91.58	88.94
	maxF_1	32.07	2.98	5.26	91.72	88.99

Table 4: A comparison of the performance of the various algorithms on the test-set for different ranking-losses on Reuters-2000.

three updates. Nonetheless, the Perceptron algorithm often seems to be doing even worse than Max. Two notable exceptions are the results of the Perceptron algorithm with respect to IsErr and OneErr on Reuters-21578. This relatively good performance of the Perceptron might be attributed to the fact that the Reuters-21578 corpus is essentially uni-labeled and thus IsErr, OneErr and the classification error used by the Perceptron are practically synonymous. As we discuss in the sequel, the performance after most of the documents have been processed is also highly correlated with the performance of the algorithms on the unseen test data. This type of behavior indeed agrees with the formal analysis of online algorithms (Helmbold and Warmuth, 1995).

The performance of the algorithms on the test sets is summarized in five tables, one for each training set. A summary of the performances on Reuters-21578 is given in Table 3. A summary of the performances on the full training set of Reuters-2000 is given in Table 4. The results for the three smaller subsets of Reuters-2000 are given in Tables 6, 7, 8 in App. A. For Rocchio, Perceptron and each update of MMP, we give the results with respect to five different ranking-losses. In addition, for each update we provide results obtained by using each of the five ranking-losses for training. We thus use each loss both for training and for evaluation and therefore have $5 \times 5 = 25$ results for each update. Each ranking-loss used for evaluation constitutes a column in each table. For each such loss, we designate the best loss achieved by any of the algorithms with a rectangle. For each update, we use bold-face to highlight the best results with respect to each of the losses. In addition we also provide in Fig. 6, 7 and 8 precision-recall graphs for Reuters-21578, the full training set of Reuters-2000, and the smallest subset of Reuters-2000, respectively.

The relative performance of the variants of MMP and the Perceptron algorithm with respect to most of the ranking-losses on the test set is consistent with their behavior on the training sets. Again, the Uniform update attains the lowest ranking-loss in most settings, and then Rand, Prop, Max and finally the Perceptron algorithm. In all the experiments, we see that with respect to the ErrSetSize ranking-loss, the best performing algorithm is MMP trained with ErrSetSize itself. However, the relative performance of the variants of MMP with respect to the other four ranking-losses is not systematic. There is no clear winner in general, though for Reuters-21578 it seems that for the rest of ranking-losses the best ranking-loss to use for training are AvgP and maxF_1. The relative performance of all the variants is clearer on the full training set of the Reuters-2000 corpus. Here the best ranking-loss to be used for training seems to be IsErr no matter what ranking-loss is used for evaluation. Note that IsErr is either 0 (when the predicted ranking is perfect) or 1. Thus, using this loss for training implies that all examples with imperfect predicted ranking receive the same overall weight. We defer a discussion of why IsErr seems the best loss for training to the closing section.

Though the Perceptron algorithm performs worse in most of the cases with respect to all of the different ranking-losses, its performance is still rather impressive. The main deficiency of the Perceptron is its poor performance in terms of ErrSetSize. It achieves the worst ErrSetSize values in most cases. This behavior can also be observed in the precision-recall graphs. The precision the Perceptron algorithm for low recall values is competitive with all the variants of MMP and even better than the variants that employ ErrSetSize on Reuters-21578 for recall values below 0.9. However, as the recall value increases the precision of the Perceptron algorithm drops sharply, and for high recall values, it exhibits the worst precision. One possible explanation for this behavior is that the Perceptron is tailored for classification and is thus insensitive to particularly bad topic-rankings that spread the relevant topics all over the ranked list. It therefore moves each of the prototypes in the same proportion regardless of how bad or good the ranking is.

Despite our attempt to implement a state-of-the-art version of Rocchio which takes into account phenomena like the length of the documents, Rocchio's performance was the worst in all experiments with the exception of the smallest subset of Reuters-2000. This is especially surprising since in a head-to-head comparison of Rocchio with recent variants of AdaBoost (Schapire et al., 1998) the performance on various corpora of the two algorithms was practically indistinguishable. (Despite the fact that it took two orders of magnitude longer to train the latter.) Amit Singhal in a private communication offered one possible explanation for this relatively poor performance. Rocchio was originally designed for *document* retrieval. Furthermore, the recent improvements that that employ length normalization were tuned on TREC's document retrieval tasks. Despite its similarity in nature to document ranking, the topic ranking task seems to exhibit different statistical characteristics and these are likely to require new adaptations and tuning for topic ranking problems. Nonetheless, on the smallest subset of Reuters-2000, Rocchio was the best performer with respect to all ranking-losses except for OneErr. This behavior is also reflected in the precision-recall graphs (Fig. 8). It is clear from the figure that, on the smallest subset, Rocchio achieves the highest precision for almost all of the recall values. Rocchio's performance is in particular good on the small subset for high recall values. For very low recall values, the performance of the different algorithms on the smallest subset of Reuters-2000 is very similar with the exception of the max update. This again might be

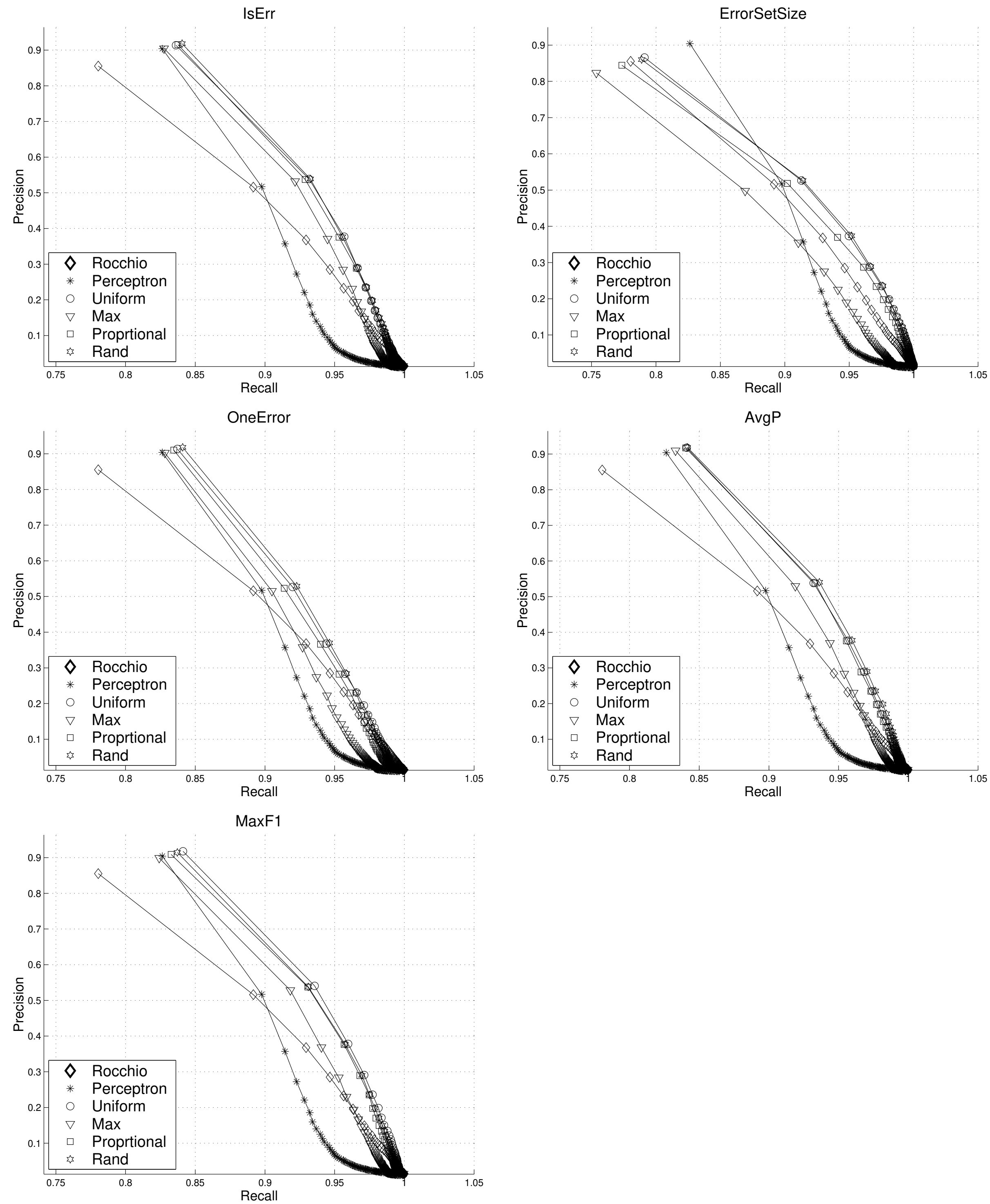

Figure 6: Precision (y-axis) versus recall (x-axis) graphs for the various algorithms on Reuters-21578.

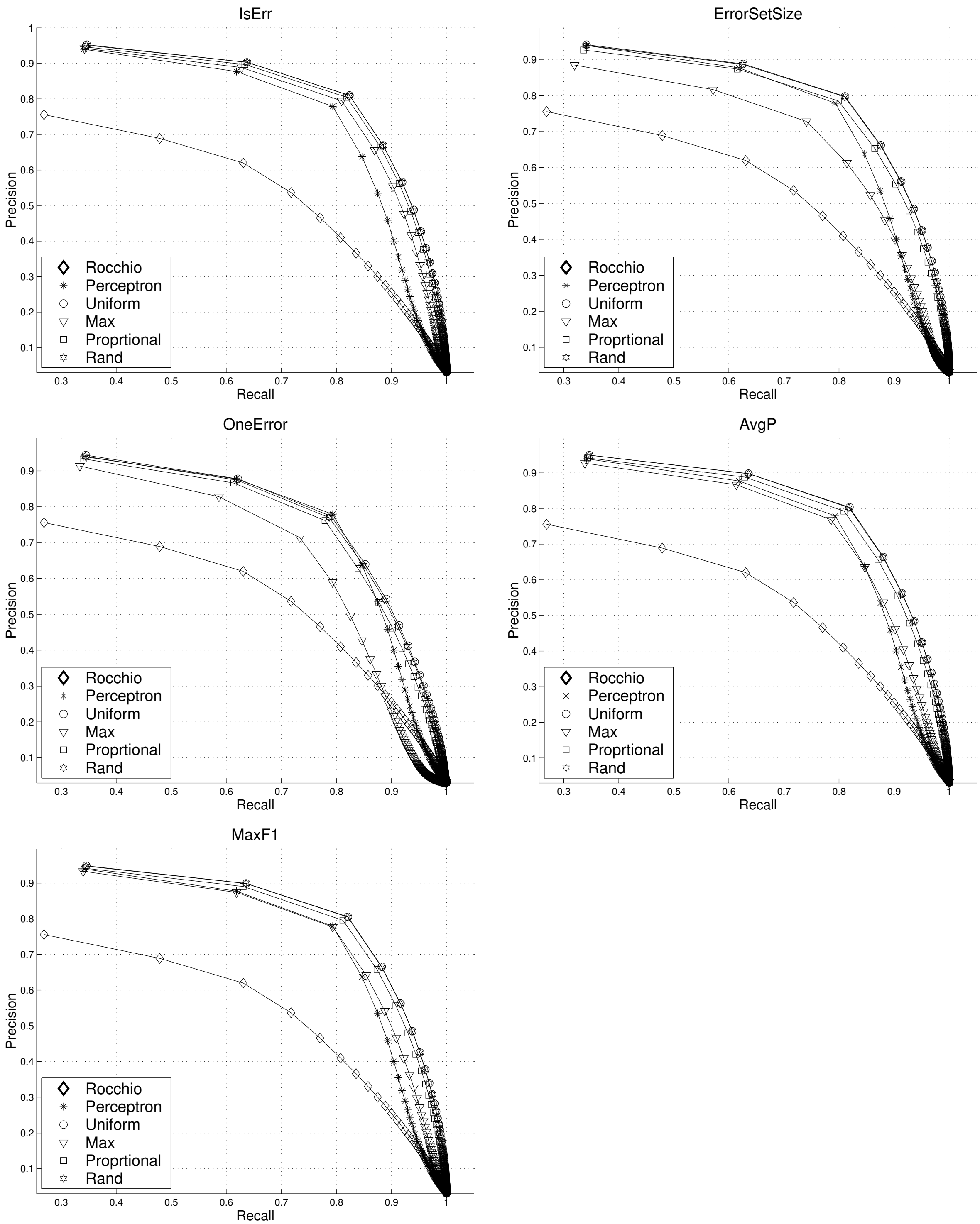

Figure 7: Precision (y-axis) versus recall (x-axis) graphs for the various algorithms on Reuters-2000.

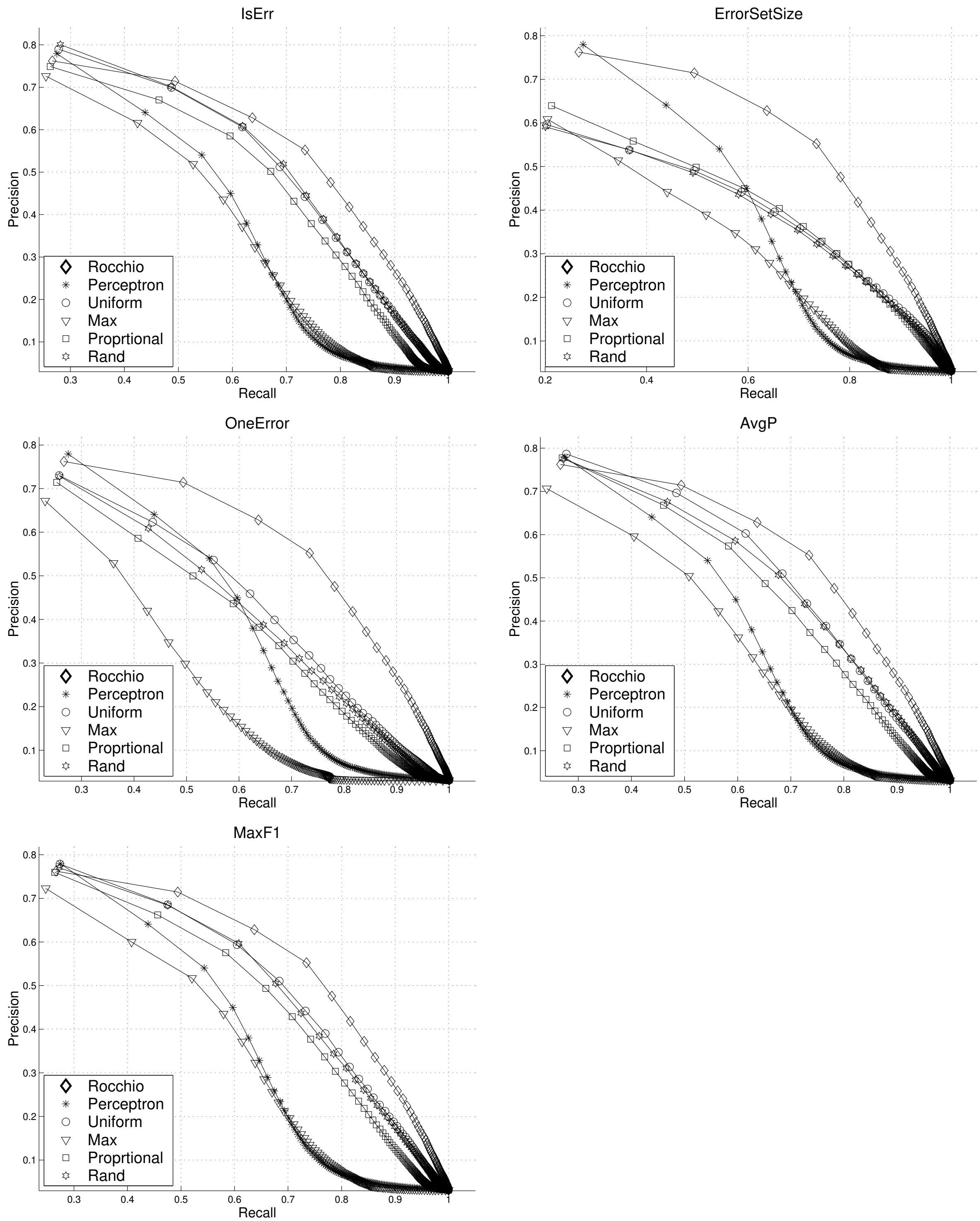

Figure 8: Precision (y-axis) versus recall (x-axis) graphs for the various algorithms on smallest subset of Reuters-2000.

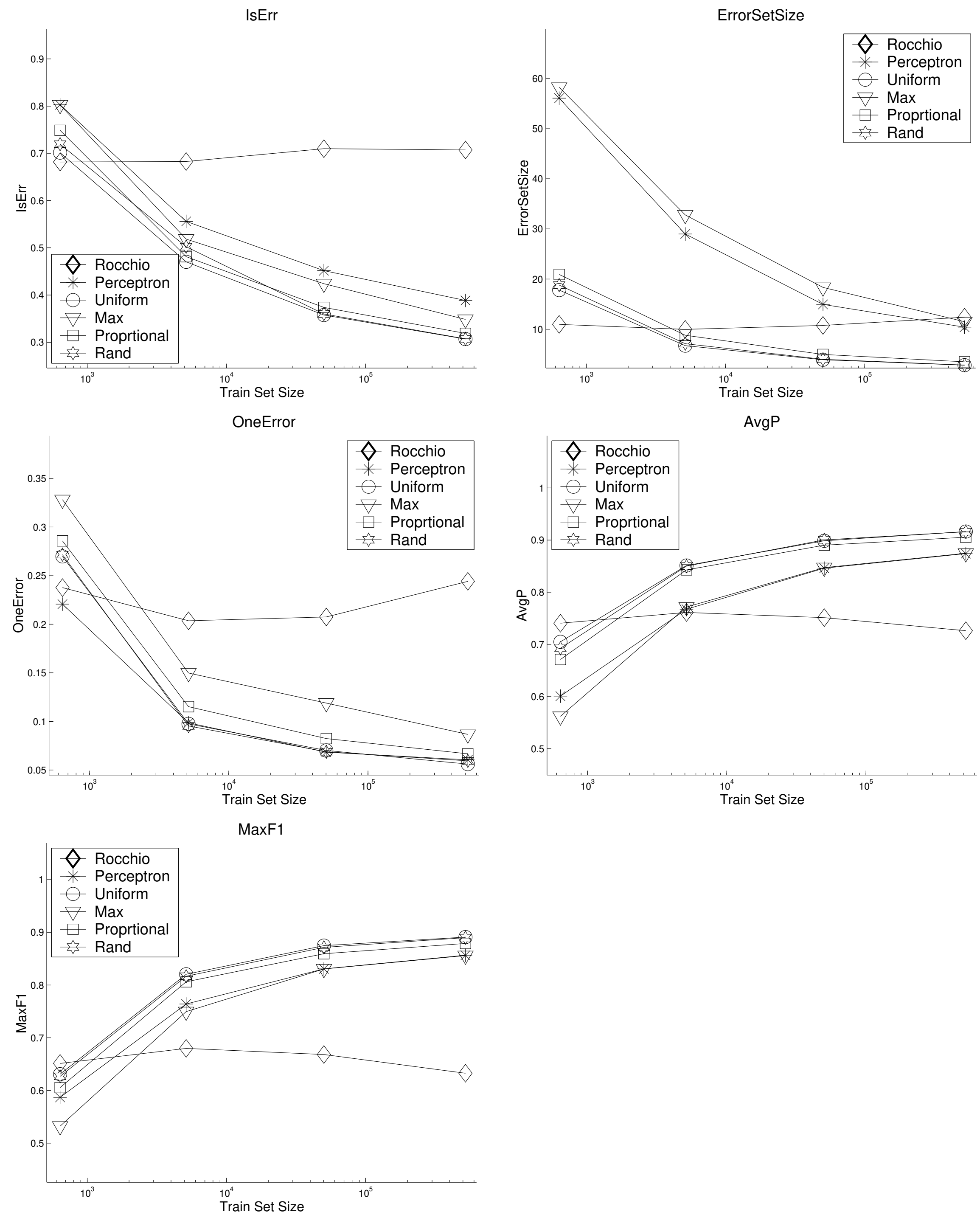

Figure 9: A comparison of the performance of the various algorithms on the test-set as a function of the number of cycles through the training set using five ranking-losses (IsErr, ErrSetSize, OneErr, AvgP, maxF1) on Reuters-2000. Each ranking-loss was used both for training and for evaluations.

		No. of Cycles				
Measure	Algorithm	1	2	3	4	5
IsErr x100	Perceptron	15.71	14.67	14.37	14.05	14.20
	Uniform	13.07	11.92	11.93	11.85	11.82
	Max	14.53	12.83	12.23	12.18	12.08
ErrSetSize	Perceptron	4.87	4.44	4.46	4.58	4.75
	Uniform	0.78	0.71	0.70	0.70	0.69
	Max	3.58	2.71	2.41	2.37	2.36
OneErr x100	Perceptron	9.59	9.20	8.81	8.72	8.77
	Uniform	8.63	8.06	8.07	7.84	8.04
	Max	9.75	8.94	8.66	8.73	8.67
AvgP x100	Perceptron	90.50	91.09	91.20	91.35	91.20
	Uniform	93.93	94.34	94.25	94.23	94.22
	Max	92.58	93.63	93.76	93.78	93.81
maxF_1 x100	Perceptron	88.56	89.21	89.56	89.68	89.60
	Uniform	90.80	91.76	91.79	91.69	91.60
	Max	88.61	90.55	90.70	90.63	91.03

Table 5: A comparison of the performance of the various online algorithms on the test sets (cross-validated) of Reuters-21578 as a function of the number of cycles through the training set.

attributed to the fact that, for very low recall values, the precision value is highly correlate with $1 - \text{OneErr}$.

The performance of the various algorithms on test data from Reuters-2000 as a function of the training set sizes is given in Fig. 9. As before, we evaluated each variant of MMP only with respect to the ranking-loss that was used during its training. (The behavior of the algorithms when training and testing with different ranking-losses was found to be similar.) We can see from the figure that the performance of the online algorithms (Perceptron and MMP) improves as the size of the training set increases. The most notable improvement is when the number of training documents increases from 639 to 5,139. On the other hand, Rocchio's performance does not seem to improve at all as the number of training documents increases. Furthermore, in some cases Rocchio exhibits a slight degradation in performance as the training set size increases. One possible explanation for this behavior is the feature selection scheme we employed which was fixed regardless of the training set size. Since the weights of the prototypes built by Rocchio are not modified based on the actual performance, a careful tuning and selection that takes the training set size into account seems to be crucial.

Finally, we focus our attention on the performance of the online algorithms as a function of the number of training epochs. The results of these experiments setting are summarized in Table 5 for Reuters-21578 and in Tables 9 and 10 (in App. B) for Reuters-2000. Each row in the tables corresponds to a different ranking-loss that was used in training and in the evaluation stage on test data. For each such loss, we compare the Perceptron and MMP with two updates: Uniform and Max. We confined ourselves to these two updates, since the results after a single epoch indicate that

Uniform is the best performing update while Max is the worst. Although for brevity we do not report here results for the other updates, these results indeed fall between the results for Uniform and Max. We report in the tables the performance at the end of each of the five epochs. We see from the tables that cycling through the training data does improve the performance of the algorithms with respect to IsErr, ErrSetSize and OneErr. Using AvgP and maxF_1 as the ranking-losses (both for training and evaluation) does not yield a significant improvement. The Max update seems to benefit the most from multiple cycles through the training data. Since the Max update takes care of only a single wrongly ordered pair of topics, multiple runs can be used to modify the weights of most if not all of the wrongly ordered topic pairs. It seems that there is a natural trade-off between the complexity of the updates (in terms of the number of pairs they modify) and the number of epochs that are needed to reach good performance. The Perceptron algorithm also seems to benefit from multiple runs through the data. Concretely, for IsErr, the loss of the Perceptron on Reuters-21578 decreases from a value of 15.7% after the first epoch to about 14% after the last epoch – an improvement of 10%. On the same corpus, MMP with the Max update reduces IsErr from a value of 14.5% after the first epoch to about 12% after the last epoch – an improvement of 20%. In contrast, examining the results obtained Reuters-2000 we see that MMP with the Uniform update using IsErr both for training and evaluation does not improve with the number of training epochs. Furthermore, it seems that often there is a slight degradation in performance, which indicates that MMP with Uniform starts to overfit the large training set. Voting and averaging techniques (Freund and Schapire, 1998) might help in preventing overfitting in such cases.

10. Summary and Discussion

In this paper we described a simple yet effective family of online algorithms for topic-ranking. Each algorithm in the family is defined through an instantiation of a set of constraints and a loss function that it attempts to minimize along its run. The approach that we took reduces the topic-ranking problem into multiple pairs of relevant and non-relevant topics that are immediately summed into weights. Thus, the run time of each algorithm per-document is linear in the number of topics. We discussed a simple analysis in the mistake bound model that provides a uniform bound for all the algorithms in the family. We performed extensive experiments with two text corpora. The experiments reveal that the online performance on the training data is pretty much reflected when testing the algorithms on new unseen data without adaption. To our surprise, the experiments indicate that the loss used for training does not necessarily achieve the best empirical loss. Furthermore, there does not seem to be a single ranking loss that consistently performs well in all settings. Nonetheless, we found that the simple notion of ranking-loss, IsErr, often results in very good performance. In particular, on the full training set of Reuters release 2000, IsErr outperforms all of the other ranking losses regardless of the loss used for evaluation. In terms of the type of the update, we found that Uniform update, which allocates the same weight for each wrongly ordered pair of topics, achieves the overall best results, but cycling through the data can improve the performance of variants which update only a subset of the wrongly ordered pairs on each round. It is interesting to note that the Max update reduces to the update devised by Collins and Duffy (2002) in re-ranking applications of uni-labeled data. Therefore, our results suggest that it might be possible to further improve the performance of the Perceptron re-ranking algorithm of Collins and Duffy if we employ update schemes that are similar to the Uniform update used in this paper. Last, we would like to note that all of the updates discussed in the paper ignore the norms of the resulting prototypes. However, there is strong theoretical and empirical evidence (see for instance (Vapnik, 1998, Cristianini and Shawe-Taylor, 2000) and the many references therein) that motivates updates which take into account the norm of the resulting prototypes. We leave this research direction for future work.

Acknowledgements:

Thanks to Amit Singhal for clarifications and suggestions on pivoted-length normalization. Thanks also to Noam Slonim for discussions and to Ofer Dekel and Benjy Weinberger for their help in pre-processing the corpora. Special thanks to Leonid Kontorovich for carefully reading the manuscript and to the anonymous reviewers for their constructive comments. Last, we would like to acknowledge the financial support of EU project KerMIT No. IST-2000-25341.

Appendix A. Detailed summary of results for Reuters-2000

Algorithm		Ranking-Loss				
Loss	Update	IsErr x100	ErrSetSize	OneErr x100	AvgP x100	maxF_1 x100
Rocchio		$\boxed{68.16}$	$\boxed{10.97}$	23.77	$\boxed{74.07}$	$\boxed{65.13}$
Perceptron		80.30	56.07	22.07	60.09	58.68
Uniform	IsErr	**70.16**	20.43	**21.03**	**70.59**	**64.41**
	ErrSetSize	86.95	**17.81**	40.37	58.94	45.20
	OneErr	80.16	27.42	26.98	63.24	56.11
	AvgP	71.70	18.40	21.36	70.49	63.93
	maxF_1	74.29	18.22	22.15	69.98	63.09
Max	IsErr	**80.23**	58.09	**27.37**	**58.49**	**54.84**
	ErrSetSize	93.84	58.24	39.15	50.54	41.99
	OneErr	93.40	86.10	32.82	46.81	44.80
	AvgP	83.78	**57.88**	29.35	56.19	51.75
	maxF_1	82.57	59.33	27.71	57.07	53.21
Proportional	IsErr	**74.89**	28.76	25.10	**67.42**	60.20
	ErrSetSize	89.95	**20.92**	36.06	59.36	47.28
	OneErr	84.98	36.82	28.57	59.68	52.58
	AvgP	75.94	25.64	**22.25**	67.11	**61.11**
	maxF_1	75.64	26.23	24.04	67.02	60.53
Rand	IsErr	71.95	20.09	$\boxed{\mathbf{19.94}}$	**70.81**	**64.75**
	ErrSetSize	87.76	18.82	40.93	58.33	44.40
	OneErr	82.24	31.00	27.16	61.46	54.45
	AvgP	76.58	**18.41**	22.44	69.27	62.25
	maxF_1	**71.78**	18.97	22.97	69.68	62.66

Table 6: A comparison of the performance of the various algorithms on the test-set for different ranking-losses using Subset 1 (639 training documents) of Reuters-2000.

Algorithm		Ranking-Loss				
Loss	Update	IsErr x100	ErrSetSize	OneErr x100	AvgP x100	maxF_1 x100
Rocchio		68.26	9.98	20.35	76.13	67.97
Perceptron		55.60	29.01	9.85	76.74	76.39
Uniform	IsErr	47.03	8.33	$\boxed{\textbf{8.03}}$	85.41	$\boxed{\textbf{82.26}}$
	ErrSetSize	57.29	$\boxed{\textbf{6.73}}$	13.20	81.55	75.62
	OneErr	65.17	14.03	9.74	78.02	74.76
	AvgP	49.04	6.90	8.99	85.12	81.21
	maxF_1	$\boxed{\textbf{46.75}}$	7.01	8.47	$\boxed{\textbf{85.60}}$	82.04
Max	IsErr	**51.85**	**23.73**	**11.50**	**79.57**	**77.06**
	ErrSetSize	76.85	32.79	20.24	69.16	63.68
	OneErr	73.65	44.49	14.98	67.76	66.23
	AvgP	60.80	26.75	11.82	77.16	74.40
	maxF_1	58.88	26.63	11.96	77.47	74.95
Proportional	IsErr	**48.13**	11.28	**8.53**	**84.35**	**81.36**
	ErrSetSize	63.69	**8.80**	16.47	78.41	71.41
	OneErr	57.94	16.05	11.51	78.89	75.40
	AvgP	49.32	9.46	9.51	84.27	80.57
	maxF_1	50.04	9.93	9.11	84.13	80.60
Rand	IsErr	50.23	8.51	8.65	84.81	81.26
	ErrSetSize	58.87	**7.12**	13.19	81.16	75.22
	OneErr	60.05	13.14	9.53	79.52	76.17
	AvgP	49.58	7.38	**8.31**	85.02	81.50
	maxF_1	**48.10**	7.14	8.75	**85.38**	**81.66**

Table 7: A comparison of the performance of the various algorithms for different ranking-losses using Subset 2 (5,139 training documents) of Reuters-2000 for training.

Algorithm		Ranking-Loss				
Loss	Update	IsErr x100	ErrSetSize	OneErr x100	AvgP x100	maxF_1 x100
Rocchio		70.99	10.76	20.74	75.13	66.81
Perceptron		45.15	14.96	6.84	84.60	83.07
Uniform	IsErr	$\boxed{\mathbf{35.76}}$	4.00	$\boxed{\mathbf{5.16}}$	$\boxed{\mathbf{90.64}}$	$\boxed{\mathbf{87.99}}$
	ErrSetSize	43.19	3.93	8.82	87.66	83.36
	OneErr	48.99	7.44	7.01	86.04	82.84
	AvgP	39.01	4.04	5.75	89.85	86.84
	maxF_1	36.67	$\boxed{\mathbf{3.92}}$	5.46	90.28	87.48
Max	IsErr	**42.39**	**10.57**	**7.07**	**86.98**	**84.35**
	ErrSetSize	65.35	18.34	15.04	77.89	72.30
	OneErr	68.78	31.93	11.88	73.92	71.66
	AvgP	47.55	14.35	8.02	84.73	81.90
	maxF_1	45.93	13.75	7.34	85.39	83.04
Proportional	IsErr	**37.38**	**4.92**	**5.71**	**89.72**	**86.97**
	ErrSetSize	48.00	4.99	11.65	85.57	79.97
	OneErr	49.33	8.92	8.24	85.33	81.71
	AvgP	40.36	5.01	6.21	89.01	85.93
	maxF_1	40.99	4.94	6.24	89.05	85.93
Rand	IsErr	**35.98**	4.02	**5.55**	**90.51**	**87.70**
	ErrSetSize	43.36	4.01	8.87	87.63	83.23
	OneErr	51.30	7.42	6.88	85.50	82.25
	AvgP	38.68	4.06	5.58	90.03	87.07
	maxF_1	36.93	**3.97**	5.88	90.21	87.14

Table 8: A comparison of the performance of the various algorithms for different ranking-losses using Subset 3 (50,139 training documents) of Reuters-2000 for training.

Appendix B. Results of Cycling though Training Set

			No. of Cycles				
Training Set Size	Measure	Algorithm	1	2	3	4	5
639	IsErr x100	Perceptron	80.30	77.72	76.38	74.98	73.00
		Uniform	70.16	71.26	69.66	70.08	68.88
		Max	80.23	80.48	72.93	70.53	70.66
	ErrSetSize	Perceptron	56.07	56.61	56.55	57.72	57.36
		Uniform	17.81	16.05	15.90	15.91	15.94
		Max	58.24	51.89	46.82	45.25	44.92
	OneErr x100	Perceptron	22.07	20.19	20.15	19.75	19.78
		Uniform	26.98	26.21	24.80	24.46	24.90
		Max	32.82	29.61	30.02	27.20	28.23
	AvgP x100	Perceptron	60.09	61.11	61.20	61.31	62.18
		Uniform	70.49	72.52	71.91	72.17	72.23
		Max	56.19	63.15	65.07	65.97	66.45
	maxF_1 x100	Perceptron	58.68	60.04	60.49	61.00	61.84
		Uniform	63.09	67.09	66.86	66.81	67.62
		Max	53.21	58.59	61.05	61.01	61.79
5139	IsErr x100	Perceptron	55.60	52.75	50.44	50.48	50.21
		Uniform	47.03	46.40	47.09	45.97	45.73
		Max	51.85	48.80	46.83	46.88	46.30
	ErrSetSize	Perceptron	29.01	24.00	23.21	23.61	24.07
		Uniform	6.73	5.68	5.69	5.69	5.73
		Max	32.79	24.28	21.83	21.15	21.07
	OneErr x100	Perceptron	9.85	8.92	8.83	8.79	8.74
		Uniform	9.74	10.22	9.10	8.70	10.15
		Max	14.98	13.02	11.81	12.83	10.45
	AvgP x100	Perceptron	76.74	79.38	80.33	80.23	80.25
		Uniform	85.12	86.63	86.59	86.75	86.67
		Max	77.16	81.75	82.86	83.42	83.41
	maxF_1 x100	Perceptron	76.39	78.33	79.21	79.22	79.26
		Uniform	82.04	83.33	83.41	83.38	83.55
		Max	74.95	80.20	80.95	81.48	81.40

Table 9: The performance of the various algorithms as a function of the number of epochs using Subset 1 and Subset 2 of Reuters-2000 for training.

			No. of Cycles				
Training Set Size	Measure	Algorithm	1	2	3	4	5
50139	IsErr x100	Perceptron	45.15	44.04	44.31	44.27	44.15
		Uniform	35.76	36.72	37.83	37.12	38.65
		Max	42.39	39.81	40.27	39.84	39.86
	ErrSetSize	Perceptron	14.96	13.75	13.29	13.81	13.43
		Uniform	3.93	3.43	3.53	3.61	3.66
		Max	18.34	13.02	11.82	10.99	10.85
	OneErr x100	Perceptron	6.84	6.59	7.19	7.53	7.46
		Uniform	7.01	7.36	6.97	6.97	7.35
		Max	11.88	9.93	9.48	9.01	8.40
	AvgP x100	Perceptron	84.60	85.31	85.32	84.82	84.91
		Uniform	89.85	89.65	89.26	89.31	89.11
		Max	84.73	86.67	87.18	87.34	87.70
	maxF_1 x100	Perceptron	83.07	83.60	83.31	82.81	82.92
		Uniform	87.48	86.71	86.41	86.26	86.50
		Max	83.04	84.97	85.31	85.04	85.12
521439	IsErr x100	Perceptron	38.86	38.79	38.71	38.65	38.62
		Uniform	30.68	30.64	31.10	31.39	31.65
		Max	34.83	34.37	34.63	34.34	34.25
	ErrSetSize	Perceptron	10.43	9.08	8.45	7.80	7.63
		Uniform	2.84	2.90	3.08	3.18	3.27
		Max	11.50	8.18	6.55	5.57	5.18
	OneErr x100	Perceptron	6.04	6.40	6.31	6.66	6.36
		Uniform	5.60	6.30	5.92	6.37	6.26
		Max	8.66	8.32	7.95	8.49	7.74
	AvgP x100	Perceptron	87.40	87.64	87.78	87.86	87.94
		Uniform	91.65	91.23	91.02	90.96	90.87
		Max	87.48	88.36	88.34	88.56	88.63
	maxF_1 x100	Perceptron	85.65	85.54	85.64	85.48	85.66
		Uniform	89.10	88.80	88.60	88.36	88.38
		Max	85.58	85.91	86.27	86.16	86.16

Table 10: The performance of the various algorithms as a function of the number of epochs using Subset 3 of Reuters-2000 for training.

References

M. Collins and N. Duffy. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *30th Annual Meeting of the Association for Computational Linguistics*, 2002. to appear.

K. Crammer and Y. Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems 14*, 2001a.

K. Crammer and Y. Singer. Ultraconservative online algorithms for multiclass problems. In *Proceedings of the Fourteenth Annual Conference on Computational Learning Theory*, 2001b.

N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, 2000.

R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

A. Elisseeff and J. Weston. A kernel method for multi-labeled classification. In *Advances in Neural Information Processing Systems 14*, 2001.

Y. Freund, R. Iyer, R. E.Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. In *Machine Learning: Proceedings of the Fifteenth International Conference*, 1998.

Y. Freund and R. E. Schapire. Large margin classification using the perceptron algorithm. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, 1998. To appear, *Machine Learning*.

D. P. Helmbold and M. K. Warmuth. On weak learning. *Journal of Computer and System Sciences*, 50:551–573, 1995.

D. J. Ittner, D. D. Lewis, and D. D. Ahn. Text categorization of low quality images. In *Symposium on Document Analysis and Information Retrieval*, pages 301–315, Las Vegas, NV, 1995. ISRI; Univ. of Nevada, Las Vegas.

H. T. Ng, W. B. Goh, and K. L. Low. Feature selection, perceptron learning, and a usability case study for text categorization. In *Proceedings of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 67–73, 1997.

J. Rocchio. Relevance feedback information retrieval. In Gerard Salton, editor, *The Smart retrieval system—experiments in automatic document processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–407, 1958. (Reprinted in *Neurocomputing* (MIT Press, 1988).).

G. Salton. Developments in automatic text retrieval. *Science*, 253:974–980, 1991.

R. E. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, 37(3):1–40, 1999.

R. E. Schapire, Y. Singer, and A. Singhal. Boosting and Rocchio applied to text filtering. In *SIGIR '98: Proceedings of the 21st Annual International Conference on Research and Development in Information Retrieval*, 1998.

A. Singhal, C. Buckley, and M. Mitra. Pivoted document length normalization. In *Research and Development in Information Retrieval*, pages 21–29, 1996.

C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, 1979.

V. N. Vapnik. *Statistical Learning Theory*. Wiley, 1998.