Error-Driven Generalist+Experts (EDGE): A Multi-stage Ensemble Framework for Text Categorization

Jian Huang*
College of Information Sciences and Technology
Pennsylvania State University
University Park, PA 16802
jhuang@ist.psu.edu

Omid Madani
SRI International
AI Center
Menlo Park, CA 94025
madani@ai.sri.com

C. Lee Giles
College of Information Sciences and Technology
Pennsylvania State University
University Park, PA 16802
giles@ist.psu.edu

ABSTRACT

We introduce a multi-stage ensemble framework, *Error-Driven Generalist+Expert* or EDGE, for improved classification on large-scale text categorization problems. EDGE first trains a *generalist*, capable of classifying under all classes, to deliver a reasonably accurate initial category ranking given an instance. EDGE then computes a confusion graph for the generalist and allocates the learning resources to train experts on relatively small groups of classes that tend to be systematically confused with one another by the generalist. The experts' votes, when invoked on a given instance, yield a reranking of the classes, thereby correcting the errors of the generalist. Our evaluations showcase the improved classification and ranking performance on several large-scale text categorization datasets. EDGE is in particular efficient when the underlying learners are efficient. Our study of confusion graphs is also of independent interest.

Categories and Subject Descriptors

H.3.3 [**Information Search and Retrieval**]: Retrieval models

General Terms

Algorithms, Experimentation, Theory

Keywords

Ensemble learning, text categorization, many class classification

1. INTRODUCTION

Automated text categorization has found a variety of applications, in personalization, document routing, recommendation, and ad placement [24]. Often, the number of classes needed for informative and useful categorization of text is large, easily exceeding 1000s. A number of data sets, such as Wikipedia, Yahoo! Directory, and OHSUMED already contain many thousands of classes and 100s of thousands of instances [15]. Well-designed linear classifiers have been shown to be among the most accurate [7, 24, 5, 4], and recently, highly scalable discriminative linear methods have been developed [17, 16, 18]. These methods treat classes as "flat", and hence improving their accuracy is a good possibility: identifying the one correct class from thousands of candidates, under linearity constraints, can be error-prone. In this work, we take a data driven approach to improve accuracy, while exploiting the efficiency benefits provided by recent advances in linear methods. We note at the outset that these ideas are in principle applicable to other learning algorithms and data types as well.

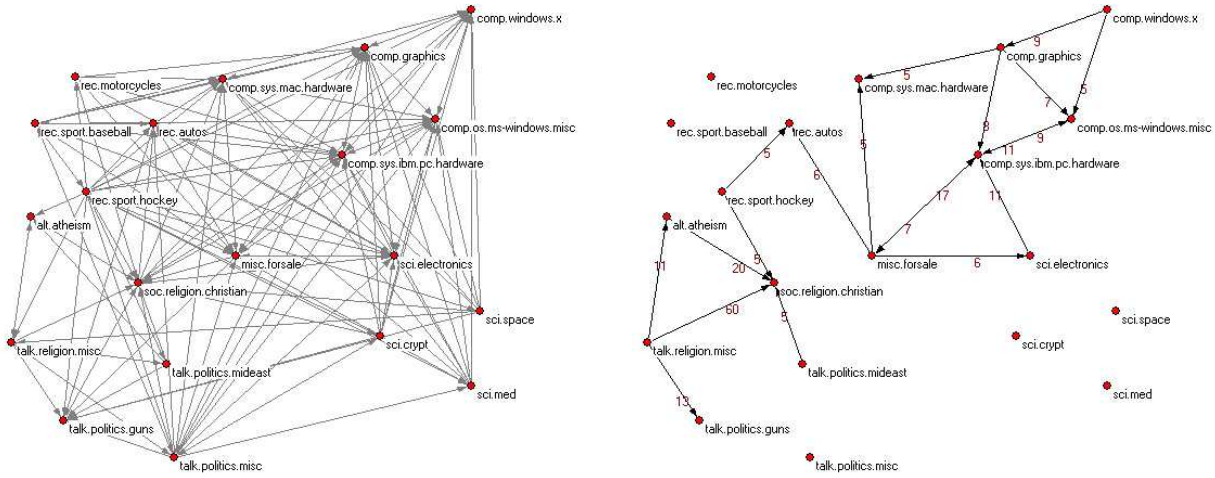
The overall approach, which we refer to as *Error-Driven Generalist+Expert* or EDGE, has roughly two parts. First a reasonably accurate (and efficient) *generalist* classifier is trained on the entire data. The confusion pattern of the generalist is characterized by a confusion graph, wherein nodes are concepts and the presence of an edge indicates significant confusion between the concept pair connected (as we describe in more detail). Highly connected concepts are then identified as "compound concepts". In the second part, an *expert* classifier is trained for each such group (only on the instances belonging to some concept in the group). Classification then consists of first utilizing the generalist, then possibly invoking some of the experts and aggregating their votes, for improved classification.

The first part of the paper reports on our studies of confusion graphs obtained from training supervised learning algorithms on several data sets, and is of independent interest. We explore the question of whether a significant portion of errors tend to be systematic, and if so in what patterns. We find that there are significant amount of systematic errors and groupings, and in particular the confusion patterns are often in star shapes. We also report a number of different statistics, and provide examples of confused concepts.

The second part of the paper explores whether this systematic confusion can be substantially corrected, without incurring too many extra offsetting errors. We train experts on each of the compound concepts, possibly using a learning algorithm different from the generalist. Experts tend to classify more accurately within their own classes than the

*This work was primarily done when the author interned at Yahoo! Research.

CIKM'08, October 26–30, 2008, Napa Valley, California, USA.



(a) Confusion graph of FF ($\theta = 1$)

(b) Confusion graph of FF ($\theta = 5$)

Figure 1: Confusion graphs generated from the Newsgroup 20 dataset [20] using the FF algorithm [16].

generalist (as we show), the intuition being that the number of classes is substantially smaller, thus the discrimination task can be easier. Another question is how to combine the votes of the different experts (when invoked) and the generalist together. We explore aggregation schemes and study different combinations for the generalist and the experts. Our experiments on a number of data sets show that simple aggregation schemes can significantly enhance overall accuracy. The overall classification and training time degrades but only moderately due to the second stage. Thus we show that the data driven EDGE approach is promising.

We assume basic familiarity with machine learning for text classification. A good reference is [24].

2. PATTERNS OF ERRORS

We first define the concept of confusion and then analyze its patterns on real world data, the findings of which motivate the design of algorithms in the following section.

2.1 Confusion as a Source of Ranking Error

The *confusion matrix* is a well known technique for depicting the error made by the classifier, especially in the multiclass single label learning setting [11]. Specifically, a confusion matrix $\mathbf{CM}$ is a $K \times K$ matrix (K is the number of classes), where an element $n_{i,j}$ represents the number of times the classifier classifies an instance of class i as class j. Consequently, the off-diagonal elements correspond to the errors made by the classifier on the data. In our setting, classifiers may *recall* multiple true categories for each instance. We extend the idea to characterize ranking error with the notion of *confusion* as follows. Seeing an instance $\mathbf{x}^t$, the learned model $\mathcal{M}$ retrieves i_n concepts and assigns a score s_{c_i} for each of the concept c_i. Hence the model produces a multi-label $\tilde{\mathbf{y}}^t = (c^t_{i_1}, ..., c^t_{i_n})$ ranked in descending order according to the scores.

DEFINITION 2.1 (CONFUSION). *For a labeled instance $(\mathbf{x}^t, \mathbf{y}^t)$, the confusion function CONF is defined as*

$$CONF[c_i \rightarrow c_j; (\mathbf{x}^t, \mathbf{y}^t)] = I[s_{c_j} > s_{c_i} \wedge c_i \in \mathbf{y}^t \wedge c_j \notin \mathbf{y}^t] \quad (1)$$

where I is the Iverson bracket and $\mathbf{y}^t$ are the true categories. In other words, concept c_i is confused *with concept c_j* iff *c_j is ranked higher than c_i, where c_i is one of the true labels but c_j is not.*

Confusion corresponds to an arc pointing from a true class to a false one that has been ranked higher by the system. Confusion captures error in categorization and is referred to as a *reversed pair* in machine learning literature [5].

Similar to the confusion matrix, we are mostly interested in systematic error made by the model in the dataset and thus this leads to the definition of *cumulative confusion* between concept c_i and c_j as follows:

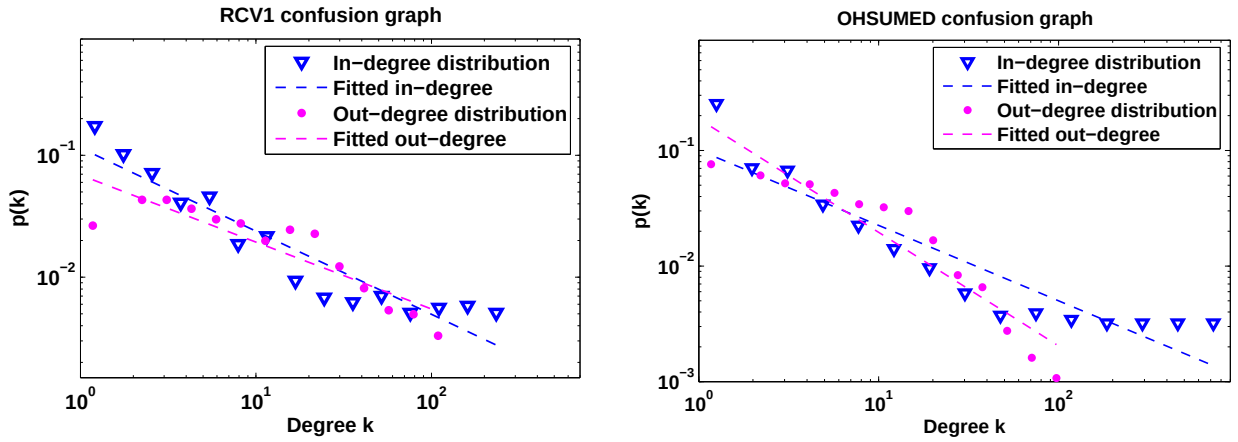
$$N_{\text{CONF}}[c_i \rightarrow c_j] = \sum_t \text{CONF}[c_i \rightarrow c_j; (\mathbf{x}^t, \mathbf{y}^t)] \quad (2)$$

Note that in the single label case, $N_{\text{CONF}}[c_i \rightarrow c_j]$ exactly corresponds to the element $nc_{i,j}$ in the confusion matrix.

As an analogy to confusion matrix, we formally define confusion graph as a means to analyze classification errors.

DEFINITION 2.2 (CONFUSION GRAPH). *Confusion graph $G(V, A; \theta)$ is a directed graph. In this graph, an arc (v_i, v_j) belongs to the arc set A if and only if $N_{CONF}[v_i \rightarrow v_j] > \theta$, where θ is a positive threshold. Each arc (v_i, v_j) is associated with weight $w_{i,j} = N_{CONF}[v_i \rightarrow v_j]$. The vertex set V is constituted with all the categories minus the singleton categories (i.e. having zero degree).*

Figure 1 demonstrates the confusion graph of the Feature Focus algorithm (FF) [16] on the Newsgroup 20 dataset [20]. Setting $\theta = 5$ (Figure 1 right), we observe that confusion is usually incurred on highly related categories. For instance, the categories *sci.electronics*, *comp.os.ms-windows.misc*, *comp.graphics* and *misc.forsale* are confused with *comp.sys.ibm.pc.hardware*. In particular, we note that confusion may occur between categories that belong to different branches of the taxonomy, e.g. *sci.electronics* and *comp.sys.ibm.pc.hardware*. On the other hand, categories that are close, in terms of 'semantic' distance in the taxonomy, may not necessarily be confused with each other,



(a) RCV1 degree distribution

(b) OHSUMED degree distribution

Figure 2: In-degree and out-degree distribution and the fitted line of confusion graphs, shown in log scale using logarithmic binning.

e.g. *comp.sys.mac.hardware* and *comp.sys.ibm.pc.hardware*. In this real example, traditional hierarchical classification approaches may redundantly train multiclass classifiers, dictated by the underlying taxonomy. Also, they may risk prematurely splitting the classification tasks, which makes the discrimination more challenging due to the large number of categories in high levels of the hierarchy. The remainder of this section seeks for natural groupings of classification tasks transcending the boundaries of a taxonomy.

2.2 Confusion Graph Analysis

We explore the confusion graph $G_{\text{RCV1}}(V, A; \theta)$[1] of the Reuters dataset (see Section 4) by using statistical graph analysis techniques. The findings motivate the choice of sub-learning problems for experts. Intuitively, we expect the confusion graph to possess some structural characteristics, e.g., clustering based on topics as relevant topics are more difficult to differentiate.

2.2.1 *Power Law Degree Distributions*

Figure 2(a) illustrates the degree distribution of the confusion graph of RCV1. We note that both in-degree and out-degree follow the power-law distribution, as one can visually fit a line in the log-log plot (the fitted in- and out-degree distributions are $P_k^{(in)} \sim k^{-0.683}$ and $P_k^{(out)} \sim k^{-0.549}$ respectively). Similar scale-free degree distribution is also observed in the OHSUMED (Figure 2(b)) and Y! Web data. We note that the in-degree distribution has a wider spread in both diagrams. In Figure 2(a), there are 30 nodes with in-degree greater than 100, compared to only 4 with out-degree greater than 100. Such a high in-degree node (called sink), combined with its satellite categories, constitutes a *star pattern* in the confusion graph. An example of the star pattern is found in Figure 1, comprised by *ibm.pc.hardware* and its satellites *ms-windows*, *electronics*, *graphics* and *forsale*.

2.2.2 *Disassortative Mixing by Node Degrees*

We consider the mixing patterns in the confusion graph in terms of node degrees. For a directed graph, the assortativity coefficient r [19] is defined as

$$r = \frac{\sum_i j_i k_i - \sum_i j_i \sum_i k_i / M}{\sqrt{[\sum_i j_i^2 - (\sum_i j_i^2)/M][\sum_i k_i^2 - (\sum_i k_i^2)/M]}} \quad (3)$$

where j_i and k_i are the excess in- and out- degree of a node i respectively, and M is the total number of arcs. Essentially, this quantity is the Pearson correlation between the normalized out- and in-degree. We found the confusion graph of RCV1 to be disassortative ($r = -0.274$). This type of degree anti-correlation implies that high in-degree nodes (sinks) are preferentially attached to by low out-degree nodes, and vice versa. Similar disassortative pattern is also found in technological (e.g. the assortativity of Internet is -0.189) and biological networks [19].

2.2.3 *Directional Connection*

We are interested in the two categories incident on a confusion arc. In G_{RCV1}, nearly 90% arcs originate from a less frequent category to a equally or more frequent category in the training data. Also, 58% arcs link from a category to another one that is at least five times more frequent. Hence, rarer concepts are more prone to be confused with more frequent concepts, but not vice versa. This is not surprising as many text categorization algorithms favor common categories to attain high (micro) accuracy.

2.2.4 *Clustering Coefficient*

Clustering coefficient of node i is defined as the proportion of the number of (closed) triangles connected to i to the number of triples centered on i. Formally, the clustering coefficient of a node i is [27]:

$$C_i = \frac{|\{(v_j, v_k) | v_j, v_k \in \text{Neighborhood}(v_i)\}|}{d_i(d_i - 1)} \quad (4)$$

where d_i is the total degree of node i and $\text{Neighborhood}(v_i) = \{v_j | (v_i, v_j) \bigvee (v_j, v_i) \in A\}$.

In G_{RCV1}, the mean clustering coefficient is 0.203, manifesting significant local clustering effect. This finding is in concord with scale-free networks, as a similar size random network has much lower clustering coefficient value

[1] In this confusion graph, $|V| = 306$, $|A| = 7268$ and $\theta = 5$.

0.0777. For confusion graphs, this indicates that two concepts confused with a common concept are also likely to be confused with each other.

2.3 Motivations for Algorithm Design

To sum up, the confusion graph is a scale-free network dominated by star-patterned subgraphs. More precisely, each subgraph has many satellite concepts with confusion edges pointing to the sink. Due to disassortative degree mixing, overlaps between subgraphs are most likely low degree nodes. According to directional connection, these satellite categories typically correspond to rarer categories. Moreover, the high local clustering effect of confusion graphs suggest that concepts are likely to connect to each other in these subgraphs as well.

These findings suggest a natural way to divide the complex text categorization problem into sub-problems and we may obtain a better solution if we can better solve these smaller learning problems. The systematic patterns of error, rather than random occurrences, permit the development of a two-stage generalist and expert learning framework, which is described in detail in the following section.

3. METHODS

In this section, we first formally define a few important concepts, before discussing the details of the EDGE framework. We then describe the methods of aggregation and finally make remarks on the proposed algorithm.

3.1 Preliminaries

The learning problem is naturally divided into sub-problems in an error-driven manner.

DEFINITION 3.1 (COMPOUND CONCEPT). *A compound concept CC_s is a set of concepts containing a sink v_s and its immediate neighbors:*

$$CC_s = \{v_s\} \cup \{v_t | w_{t,s} = N_{CONF}[v_t \rightarrow v_s] > \theta\} \quad (5)$$

where the sink v_s belongs to the set of top $p\%$ nodes[2] *according to the in-degree distribution of the confusion graph $G(V, A; \theta)$.*

DEFINITION 3.2 (EXPERT). *An expert E_s is a model learned from the truncated problem $Tr(CC_s)$, derived from the compound concept CC_s and its corresponding instances,*

$$Tr(CC_s) = \{(\mathbf{x}^t, \mathbf{z}^t) | \mathbf{z}^t = \mathbf{y}^t \cap CC_s \text{ and } \mathbf{z}^t \neq \phi\} \quad (6)$$

By contrast, a generalist is a model learned from the entire learning problem $\{(\mathbf{x}^t, \mathbf{y}^t)\}$. The concepts of generalist and expert play a central role in the EDGE framework. Briefly, the generalist provides an initial ranking of concepts efficiently, while experts fine tune the ranking among the retrieved concepts. It is also worthwhile to note the connection between the two concepts. An expert can be regarded as a special type of 'generalist', which simply acknowledges the predictions of the generalist for concepts outside its realm of expertise. For concepts within the compound concept, the expert reweighs them and thus the expert can also implicitly rank all the categories given the generalist's advice.

[2]The parameter $p\%$ in the definition is motivated by the previous findings, as nodes in the tail of the in-degree distribution correspond to the sinks in the star patterns.

3.2 The Error-Driven Generalist+Experts (EDGE) Framework

The EDGE framework is a two stage ensemble learning method consists of generalist and expert learning, as shown in Figure 3. An aggregator is responsible for combining the generalist and experts' decisions.

Algorithm 1 describes the training phase of EDGE. The framework utilizes multiclass learning algorithm(s) (denoted by Comp_Learn_Algo) for training a generalist and experts. After learning a generalist, we compute cumulative confusion and construct the confusion graph G. A set of compound concepts are then found for the top $p\%$ sinks according to the in-degree distribution of G. In line 6, the original training data are truncated so that instances not containing any relevant concepts (in the compound concept) are removed. Also, features not pertinent to the compound concept are not presented to the learner, and thus the experts may consume much less model space. For each of these compound concepts, an expert is learned in line 7. Here the learner may differ from that in line 2, as a more accurate model may better serve as an expert. If the expert is better suited for classification than the generalist (line 8), it is added to the set of experts EXP (line 9).

Algorithm 1 EDGE training phase.

Input: Training dataset $\{(\mathbf{x}^t, \mathbf{y}^t)\}_{t=1}^N$; θ
Output: GEN and EXP
1: Train generalist GEN=Comp_Learn_Algo($\{(\mathbf{x}^t, \mathbf{y}^t)\}_{t=1}^N$)
2: Compute the confusion graph $G(V, A; \theta)$ of GEN in the training data
3: Obtain compound concepts $\{CC_1, ..., CC_M\}$ by using the top $p\%$ sinks according to the in-degree distribution of G (where $M = |V| \cdot p\%$).
4: EXP $\leftarrow \phi$
5: **for** $m = 1$ to M **do**
6: Obtain a truncated learning problem $Tr(CC_m)$ for compound concept CC_m
7: Train an expert E_m = Comp_Learn_Algo$'(Tr(CC_m))$
8: **if** E_m's training error is lower than that of GEN in $Tr(CC_m)$ **then**
9: EXP$\leftarrow$ EXP$\cup\{E_m\}$
10: **end if**
11: **end for**
12: **return** Generalist model GEN and experts EXP

The classification phase of EDGE is shown in Algorithm 2, where the subscripts 1 and 2 differentiate the probabilities predicted by first level classifier (the generalist) and second level classifiers (experts) respectively. First, the generalist produces an initial ranking of categories and computes the prior probability of compound concepts. For each compound concept having prior probability higher than γ[3], the corresponding expert is then activated to re-score the categories within the compound concept. Line 6 and 7 realize the view of a specialist learning framework [10]. An expert reweighs the concepts in its compound concept proportionate to its own probabilistic prediction (line 7). On the other hand, it leaves the sum of weights for the concepts outside its expertise unchanged by simply passing on the prediction of the generalist. In line 10, the scores of

[3]γ is an insensitive parameter and is empirically set to 0.1 in our experiments.

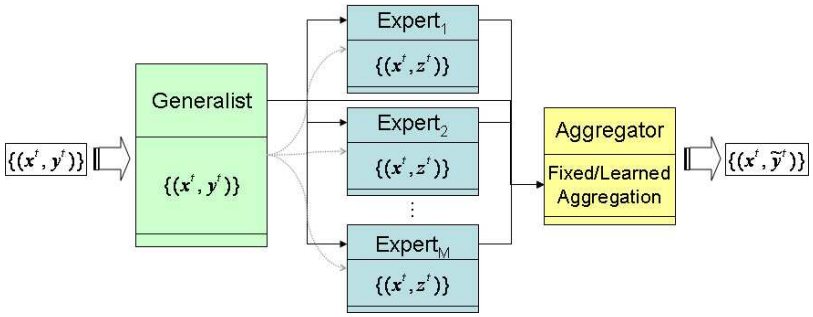


Figure 3: Architecture of the EDGE framework.

Algorithm 2 EDGE classification phase.

Input: Testing instance $\mathbf{x}$; activation threshold γ.
Output: Predicted categories $\tilde{\mathbf{c}}$.

```
GEN predicts probabilistic score s_{c_i}(GEN) = P_1(c_i|x) for each category c_i
for all E_m ∈ EXP do
    Compute the probability of compound concept CC_m: P_1(CC_m|x) = Σ_{c_i ∈ CC_m} P_1(c_i|x)
    if P_1(CC_m|x) > γ then
        Activate expert E_m to predict probabilistic score P_2(c_j|CC_m, x), where c_j ∈ CC_m
        Normalize the probabilistic scores of concepts predicted by the awake expert E_m
        s_{c_j}(E_m) = P_2(c_j|CC_m, x) · P_1(CC_m|x)
    end if
end for
Aggregate scores assigned by GEN and the subset awake experts in EXP
return Sorted categories c̃ using the aggregate scores
```

the generalist and experts are aggregated to make the final prediction, the details of which are discussed next.

3.3 Aggregation Methods

We now turn our attention to exploring several aggregation methods for combining the ranking decisions given by the generalist and experts. The first approach applies fixed mathematical rules for aggregation, and the latter learns such aggregation rules from data.

3.3.1 Reranking with Fixed Combination

Before proceeding to the aggregation methods, we first recall the setting of our work. First, the framework is intended to handle large scale datasets and thus it prohibits the use of computationally expensive combination methods. Second, experts in the EDGE framework specialize on different part of the feature space and only have access to the features and categories in their respective compound concepts. This is further complicated by insufficient training data in rare categories. As such, experts' performance may not be directly comparable with each other. Various simple combination methods have been proposed in the literature, including averaging [25], voting [29] belief integration in Bayesian formalism and Dempster-Schafer formalism [29], etc. However, many of these methods implicitly assume the comparability of classifiers for pooling or require expensive computation, which is not applicable in our setting.

The averaging method, which simply computes the arithmetic average of the classifiers' output, is an exception to these drawbacks. Simple averaging has been observed to provide comparable, sometimes superior, accuracy, with minimum computation cost [25]. Following the notations in Algorithm 2, consider the aggregation of the generalist output $s_{c_i}(\text{GEN})$ and the scores $s_{c_i}(E_m)$, where $E_m \in \text{A\_Exp}(i)$ (the set of activated experts for class i). If $\text{A\_Exp}(i)$ is empty, the aggregated score is trivially $s_{c_i}(\text{GEN})$. Otherwise, the simple averaging score is[4],

$$s_{c_i}^{ave} = \frac{1}{|\text{A\_Exp}(i)|+1}\left[s_{c_i}(\text{GEN}) + \sum_{E_m \in \text{A\_Exp}(i)} s_{c_i}(E_m)\right] \quad (7)$$

Since the performance of experts learned in EDGE may be uneven, simple averaging can be susceptible to poorly performed expert. Robust order statistics combiners have been suggested as alternative simple combination methods with theoretical underpinnings [26]. Assume that we have N scores from the generalist and experts for class i (by replacement of labels), ordered ascendingly,

$$s_{c_i}^{(1)} \le s_{c_i}^{(2)} \le ... \le s_{c_i}^{(N)} \quad (8)$$

Accordingly, the kth order statistics is $s_{c_i}^{(k)}$ and the max and min combiners are $s_{c_i}^{max} = s_{c_i}^{(N)}$ and $s_{c_i}^{min} = s_{c_i}^{(1)}$.

Depending on the underlying component learning algorithm, we assume the probabilistic scores assigned by the generalist and experts to be an estimation of the posterior class probability. Hence the interpretation of the max combiner is to choose the classifier that is most confident about its prediction as the combiner's output. The downside is that it can be dominated by the classifier that generally outputs high values, and deteriorates if such classifier performs poorly. The min combiner performs a minimax operation, and may suffer less from a single error [26]. The spread combiner was proposed in [26] to avoid the impact of a single classifier's output on the final output,

$$s_{c_i}^{spr} = \frac{1}{2}(s_{c_i}^{min} + s_{c_i}^{max}) \quad (9)$$

3.3.2 Learning The Aggregation

From an alternative point of view, the aggregation of decisions in general can be regarded as obtaining the optimal

[4]Recall that an expert can be seen as a special 'generalist' that reranks categories in its compound concept and passes on the generalist' decisions for other categories. The output of the generalist and experts are thus treated similarly.

configuration of the prediction of experts and generalist. Therefore, we can create a new learning problem where the features are the outputs of the generalist (for all categories) and the experts (for their respective compound concepts), and the output space corresponds to all the categories. A fully fledged classifier can be used for learning the aggregation. From the perspective of this additional learner, the generalist and experts act as agents for feature reduction and selection, as generally there are many more (orders of magnitude) features than categories. Also, the learning task may be much less difficult as feature values are strongly indicative of the categories. Therefore, a simple (perhaps linear) learner would suffice for this task to minimize computation burden and avoid overfitting.

As a conclusion to this part, we establish the links of aggregation to other ensemble learning framework. By using different types of classifiers in the generalist and experts, or by training classifiers using different features, the biases and error of the classifiers will be less correlated. In this manner, the ensemble framework promotes diversity among classifiers which closely follows the same rationale as many other ensemble methods. Also, the fixed combination method has also been shown to reduce error due to variance reduction.

3.4 Remarks on the Algorithm

We first comment on several aspects of efficiency of the proposed EDGE framework. The computation of the compound concepts is done by testing the generalist on the training data. In online learning setting, this can usually be done along with the update step.

Second, compared to boosting methods [9] where weak hypotheses are of the same size, each expert is usually much smaller than the generalist and is learned much faster. Also, unlike weak hypotheses learned in a sequential manner, experts learn and predict independently and thus the computation can be performed in parallel.

Third, EDGE is flexible in combining different types of multi-class learners (to strike the balance of space and time efficiency) to achieve superior results. Boosting strong classifiers, on the other hand, may lead to overfitting.

Also, although we use linear methods for efficient categorization, the two-stage classification and the combination methods render the EDGE method nonlinear. Nonlinearity (and the capability of experts to better discriminate between *difficult* categories) yields performance gains over a monolithic linear learning method, as we show in the next section.

4. EXPERIMENTS AND RESULTS

In this section we first introduce two sets of evaluation metrics, followed by the description of the experiment datasets and component learners. A series of experiments are conducted to investigate the performance of the aggregation methods, the experts and the EDGE algorithm.

4.1 Evaluation Metrics

We use two sets of metrics to evaluate the efficacy of the proposed approach. The first set of metrics measure the position of the highest ranked true category, denoted by $k_{\mathbf{x}}$ ($k_{\mathbf{x}}$ is an integer and $k_{\mathbf{x}} = \infty$ if true categories are not retrieved). We define recall at k, denoted by R_k, to be the proportion of instances for which at least one of the retrieved true categories are among the top k categories:

$$R_k = E[k_{\mathbf{x}} \leq k] \tag{10}$$

In the single label (multiclass) setting, R_1 corresponds to the traditional accuracy. Since the learners in this paper are capable of providing a list of ranked categories, we also report R_5 to measure the portion of test instances having at least one true category among the top five categories.

In the multi-label or ranked retrieval setting, we are interested in the algorithm's performance in ranking categories. $\max F1$ is derived from the $F1$ measure which has been commonly used in evaluating Information Retrieval systems [5]. For an instance $\mathbf{x}$, the $F1(r)$ value (at position r) is the harmonic average of precision $P(r)$ and recall $R(r)$, i.e.

$$P(r) = \frac{TP(r)}{r} \qquad R(r) = \frac{TP(r)}{NTP} \qquad F1(r) = \frac{2P(r) \cdot R(r)}{P(r) + R(r)}$$

where $TP(r)$ is the number of true positives in the top r positions and NTP is the total number of true categories of instance $\mathbf{x}$. $maxF1$ is thus defined as $maxF1 = \max_r F1(r)$.

Precision Recall Break Even Point (PRBEP) [2, 30] is commonly used for measuring the performance of text categorization. A text categorization algorithm typically exhibits some trade-offs between precision and recall. For instance, an algorithm that retrieves all categories can achieve perfect recall (1) while scoring poor precision, and vice versa. For a document (or an instance) $\mathbf{x}$, PRBEP is the precision or recall at position r where they are equal. Equivalently, PRBEP corresponds to the point where the classifier is tuned to have the same false positive and false negative. Note that at this point, $F1(r)$ is equal to $P(r)$ or $R(r)$, and thus PRBEP is no greater than $maxF1$.

Another popular metric for the text categorization community is 11-point Average Precision (11-pts AvgP), which computes the average of interpolated precision values of the retrieved categories in a document in 11 recall levels (0%, 10% ... , 100%) [30].

The global average values of $maxF1$, PRBEP and 11-pts AvgP, giving equal weight to each document (micro-averaging), are reported in the sequel. We remark that in tasks with thousands of classes, such as categorizing web pages or news articles, we seek to label a given text document with one or a few classes that the system is confident about. In these tasks, performance measures based on ranking classes per instance, as we have presented above, are more appropriate than category based metrics (i.e. those measures that evaluate the quality of the ranking of instances for each category). The ranking metrics $maxF1$ and PRBEP can be regarded as summary statistics of the precision-recall curve. Moreover, they are independent of the cutoff threshold usually required by text categorization algorithms. In addition, 11-pts AvgP accounts for the precision of the algorithm across all recall levels. To sum up, R_1 and PRBEP are strict measures of relevance, while R_5 and $maxF1$ are relatively more optimistic metrics.

4.2 Evaluation Datasets

Table 1 summarizes the datasets used in the evaluation. Two benchmark datasets, commonly used for text categorization [31], are used for evaluating the performance of the proposed algorithm. RCV1 [14] is the training split of the Reuters Corpus Volume 1 data. OHSUMED [12] is a sample of collection from the US National Library of

Medicine's bibliographical database PubMed. OHSUMED covers abstracts from 270 bio-medical journals in a five year span from 1987 to 1991. The labels are the human-assigned Medical Subject Heading (MeSH) terms. Y! Web is a subset of the Yahoo! web directory for classifying web pages.

These three datasets are large-scale collections (hundreds of thousands of instances and features as well as tens of thousands of categories) used in operational settings. In our experiments, features are standard unigrams and instance vectors are l_2 normalized. Also, these datasets have been randomly split into two subsets in each trial, 90% for training and the remaining 10% for testing. More details of these datasets can be found in [16].

Table 1: Dataset overview. N, D and K stand for the total number of instances, features and categories. The last two columns represent the average number of features and categories per instance.

Dataset	N	D	K	Feats /Inst.	Cats. /Inst.
RCV1	23,149	47k	414	76	2.1
Y! Web	69,591	685k	14k	210	1.0
OHSUMED	233,445	233k	14.3k	87	12.3

4.3 Component Learners

We briefly describe the two types of component learners used in our experiments. These algorithms learn linear classifiers, and are capable of handling many training instances and very high input (feature) dimensionality. Since the goal of the paper is to develop a meta-learning framework, we refer the readers to the original papers for more details of the respective algorithms.

The first type of multi-class learners are *feature-based*. The Feature-Focus (FF) [17, 16] algorithm efficiently learns an index from features to categories. The index is kept sparse by controlling out-degrees of features and using weights adding and dropping policy during online learning [16]. FF drastically reduces training and prediction time compared to other one-against-rest and top-down hierarchical classification approaches. Meanwhile, FF enjoys similar or higher levels of accuracy when compared to the state of the art (e.g. one-versus-rest SVMs), and it is substantially better than simpler methods such as Naive Bayes [16].

The second type of multi-class classifiers are *prototype-based*, where a prototype vector $\mathbf{w}$ is learned for each class. In prediction, the similarity between a class prototype and a test instance is computed by their inner product and categories are retrieved according to the similarity scores. Passive Aggressive (PA) [4] formalizes the learning problem as an optimization problem of minimizing the sum of the prototype changes and the hinge loss (or its variants). Compared to feature-based methods, prototype-based methods require the storage of a K by D prototype matrix (where K is the number of classes and D the feature dimensions), which may be infeasible when K and D are large. Also, the time for evaluating the similarity of class prototypes is proportional to the number of classes and thus they are inefficient when there are thousands of classes.

4.4 Experiments with Aggregation Methods

The first set of experiments evaluate the performance of different aggregation rules as discussed in Section 3.3. We picked RCV1 as the test set which has relatively fewer categories. Hence the overlap between compound concepts is more significant and allows us to better distinguish the relative performance of different combiners. We arbitrarily set the number of experts to 10 and each compound concept covers 30% to 60% of the categories.

The upper half of Table 2 shows the performance of baseline classifiers FF and PA. PA outperforms FF in this dataset, at the cost of more space consumption and much longer (over 10 times) training time. Hence we used FF as the generalist and PA as experts in EDGE. The lower half of the table demonstrates the performance of EDGE using different combiners. We observe that in general, EDGE outperforms the baseline component learners. The performance of the order statistics combiners *min* and *max* is similar, outperformed by the learning aggregation method (using PA). The simple averaging combiner performs slightly better than the learning aggregation method in the ranking metrics, and similar in the binary recall metrics. Considering the efficiency in learning and testing, we choose to use the averaging combiner in the following.

On a side note, one is interested in the performance of combining the stronger baseline in EDGE. As is shown, EDGE using PA as both the generalist and experts performs better than the PA baseline but inferior to that with the FF and PA combination. One intuitive explanation is that using different classifiers promotes diversity in the ensemble and reduces the correlation of errors (biases) in the generalist and experts. Experts may thus be more capable of rectifying the confusion of the generalist.

4.5 Experts Learn Better

The premise for EDGE to outperform the generalist is that experts should be more capable of ranking categories in their compound concepts, either because the discrimination task is easier with fewer classes or because a more sophisticated learning method is used. We verify this by using FF as the learner for both the generalist and experts in Y! Web data. Figure 4 compares the R1 value of the generalist and experts (before deactivation in line 8 in Algorithm 1) in the truncated learning problems. Since all the points locate above the $y = x$ line, we conclude that experts perform better than the generalist within their domains of expertise. Similar scatter plots were also observed in other datasets (not shown due to space constraints).

Table 2: Performance comparison of component learners and EDGE using different combiners in the RCV1 dataset. The generalist and experts (and learning aggregation) are separated by a plus sign. Highest values in a column are shown in bold font.

Learner	$R1$	$R5$	$maxF1$	PRBEP
FF baseline	0.781	0.943	0.767	0.651
PA baseline	0.856	0.974	0.833	0.750
EDGE[FF+PA] *max*	0.849	0.981	0.847	0.758
EDGE[FF+PA] *min*	0.866	0.969	0.834	0.755
EDGE[FF+PA] *spr*	0.865	0.981	0.855	**0.774**
EDGE[FF+PA] *ave*	**0.874**	**0.983**	**0.860**	**0.778**
EDGE[PA+PA] *ave*	0.865	0.981	0.851	0.767
EDGE[FF+PA+PA]	**0.874**	0.978	0.851	0.772

Table 3: Comparison of accuracy in three datasets. Relative percentage improvement of EDGE over the corresponding generalist method is shown in brackets. Standard deviations of the values in five splits are not greater than 0.008. The highest values of each metric in the datasets are shown in bold.

Dataset	Learner	$R1$	$R5$	$maxF1$	PRBEP	11-pts AvgP
RCV1	FF	0.781	0.943	0.767	0.651	0.733
	EDGE [FF+PA]	**0.874** (+11.9%)	**0.983** (+4.2%)	**0.860** (+12.1%)	**0.778** (+19.5%)	**0.840** (+14.6%)
	Perceptron	0.621	0.815	-	-	-
	Committee	0.769	0.918	-	-	-
	SVM	0.783	0.939	-	-	-
Y! Web	FF	0.375	0.576	0.514	0.375	0.473
	EDGE [FF+FF]	**0.395** (+5.3%)	**0.593** (+3.0%)	**0.528** (+2.7%)	**0.395** (+5.3%))	**0.485** (+2.5%)
	Perceptron	0.098	0.224	-	-	-
	Committee	0.207	0.335	-	-	-
OHSUMED	FF	0.785	0.989	0.553	0.474	0.497
	EDGE [FF+FF]	0.877 (+12.0%)	**0.995** (+0.6%)	0.582 (+5.3%)	**0.508** (+7.2%)	**0.532** (7.0%)
	EDGE [FF+PA]	**0.928** (+18.2%)	**0.995** (+0.6%)	**0.584** (+5.6%)	0.507 (+7.0%)	**0.516** (+3.8%)

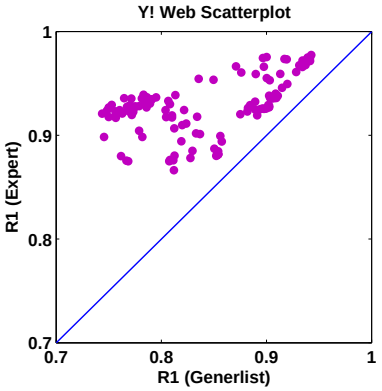


Figure 4: Scatter plot of R1 values of the experts and the generalist in Y! Web dataset using FF.

4.6 Performance Comparison

We include the comparison results with several popular linear text categorization methods in the RCV1 and Y! Web datasets [16]. Support Vector Machines (SVMs) was originally designed for two class classification problems. One-versus-rest is the most popular mechanism for reducing multi-class problems to two-class problems, where a set of K classifiers are trained to discriminate between one class and the other $K-1$ classes. Though the comparison of these independently trained classifiers (on different problems) may not be meaningful, properly regularized binary classifiers are capable of achieving competitively empirical results [21]. We use a state-of-the-art efficient variant of the SVMs algorithm [13] in our comparison. As the number of classes exceeds the range of 10s or 100s of categories (many class setting), one-against-rest methods are incapable of handling problems of such scale. In the realm of K-ary classification, We empirically compare our methods with the Perceptron algorithm [22], which uses additive updates for linear online learning. Additionally, we compare with a committee of 10 Perceptrons [3], which has shown performance comparable to linear SVM while being more efficient. Note that these learning algorithms are different from our component learners as well as the EDGE ensemble mechanism.

Table 3 summarizes the average results of EDGE over five trials of each dataset[5]. The generalists used in these datasets are competent in the text categorization task (e.g., FF achieves similar R1 and R5 as SVMs, significantly outperforming Perceptron and Committee). On the other hand, EDGE overall yields substantial performance gains in both classification and ranking metrics in all three datasets.

As we've shown earlier, EDGE achieves over 10% gain in the $R1$ and $maxF1$ metrics compared to the generalist in the RCV1 dataset. EDGE also outperforms linear one-versus-rest SVMs in the R1 and R5 metrics. In the single-label Y! Web dataset, the improvement of EDGE in the $R1$ and $R5$ metrics are 5% and 3% respectively. On the other hand, the OHSUMED dataset has as many as 12 categories per instance and we are more interested in the ranking results. Using FF and PA as the experts, EDGE outperforms the generalist by 5%, 7% and 7% in the $maxF1$, PRBEP and 11-pts AvgP metrics. We also provide the results of committee of 10 perceptrons [16] for comparison. EDGE outperforms committee by 13% in the RCV1 dataset and as much as 91% in the Y! Web dataset. In all, EDGE improves the accuracies of the component learners without showing signs of overfitting in these large-scale datasets.

It is also worthwhile to note the scalability of the methods using the largest dataset OHSUMED. FF finished learning in 2 minutes, whereas PA was not able to run as a generalist algorithm due to the huge number of categories (and consequently the prohibitive space and time consumption). EDGE learned 60 FF experts (with tens to hundreds of concepts each) in less than 25 minutes. By decomposing the categories into compound concepts, EDGE also allowed PA to learn as experts, yielding slightly better results. We note that the performance is better than that of KNN (micro-F1 = 0.51, substantially slower and with significant feature reduction) [30], the only text categorization method used on the full domains of MeSH terms in OHSUMED [31].

4.7 Improvements on a Relative Scale

The ultimate performance metric for classification algorithms is classification error, which consists of model error and Bayes error. Bayes error is the irreducible error of the

[5]Empirically, 20% was used for p% and θ=5, 5, and 200 in RCV1, Y! Web and OHSUMED respectively. The choice of these parameters is however not sensitive for EDGE.

Table 4: Percentage of instances fixed. $R1$ Gen and $R1$ max denote the $R1$ value of the generalist and the optimal classifier respectively.

Dataset	#Fixed	#Reversed	#Net fixed	#Fixable	Fixed Percentage	$R1$		
						GEN	EDGE	max_fixed
RCV1 (training)	1,120	320	818	1,318	62%	0.930	0.971	0.993
RCV1 (testing)	276	64	212	425	50%	0.781	0.874	0.964

problem, with respect to the true data distribution. The underlying distribution, however, is in general unknown and thus one can only measure such error indirectly. This section reveals the approximate reduction of model error to help us better interpret the results in Table 3.

We first define two events corresponding to the fine-tuning of the experts. We call an instance *fixed*, if the generalist fails to rank its true category in the first rank but EDGE successfully ranks it first (via experts' reranking and decision aggregation). Similarly, an instance is *reversed*, if EDGE fails to rank the true category first while the generalist ranks it correctly. Clearly, model error is reduced if the number of net fixed (#fixed - #reversed) is greater than 0. We also call an instance *fixable* if there is an edge from the true category to the top-ranked false category in the confusion graph. In other words, we ignore those insignificant confusions below the θ threshold which are prone to be noise. A perfect error-driven framework (denoted by max_fixed) may optimally remove all edges from the confusion graph, though this is generally not achievable because the Bayes error is non-zero with overlapping class distributions.

Table 4 illustrates the gain of EDGE in the $R1$ metric in the RCV1 dataset. EDGE is able to fix 62% of the fixable instances in the training data, and approximately half in the test data. This is also reflected in its R1 value, which is approximately the mid-point of that of GEN and max_fixed.

5. RELATED WORK AND DISCUSSIONS

Automatically categorizing text documents is a challenging task with the availability of massive datasets [7, 31, 15]. The EDGE method proposed in this paper leverages recent advances in linear classification methods [5, 4, 16] and provides a scalable solution for this problem.

EDGE belongs to the family of ensemble learning, which is an important subject of machine learning research. We only discuss a few popular ensemble methods, highlighting similarity and distinction with the proposed method. Boosting is a particularly popular error-driven ensemble method. AdaBoost [9] regards each learner as a *weak hypothesis* and combines multiple learners into a committee for classification. Unfortunately, boosting methods are not practical given the scale of the text categorization problems: it is infeasible to run a many-class learner on the reweighed data for hundreds or thousands of iterations. Also, they risk overfitting when the underlying learner is strong. TreeBoost.MH [8] is an extension to the popular multiclass AdaBoost.MH algorithm [23] by accounting for the hierarchical structure. It reduces the time complexity by an exponential factor compared to AdaBoost.MH. However, to achieve similar accuracy, TreeBoost.MH is still two orders of magnitude slower than the linear learning methods used in this paper, and thus is not scalable to massive data. Stacking [28] is another meta-learning approach for pooling classifiers. Stacking works by first learning multiple classifiers and then a *generalizer* which combines their outputs. This is not unlike learning aggregation in Section 3.3.2. However, the first level classifiers still learn from the entire dataset. In contrast to these ensemble methods, samples from the entire feature space during testing, we propose to EDGE only learns experts that specialize only on part of the feature space (features and categories).

Taxonomies offer potential efficiency and efficacy advantages in text categorization [6, 1, 15]. However, premature mistakes in high-level classifiers can hamper accuracy in lower level categories. Also, a taxonomy may be unavailable or may not be a tree (e.g. MeSH terms in OHSUMED form a DAG), besides its complex implementation. Instead of learning classifiers for sibling categories, EDGE discovers compound concepts in a data-driven manner. These compound concepts are not confined to the hierarchical structure, as sibling categories are not necessarily confused and easily confused categories may be non-siblings (e.g. *alt.atheism* vs. *soc.religion.christian* in Figure 1). Hence error-driven decomposition of the text categorization problem may overcome the aforementioned drawbacks in hierarchical text categorization and yield improved performance.

Our work shares similarities with the work of [11], who used the confusion matrix of a Naive Bayes classifier to scale up one-versus-rest SVM learning. In their work, the first level classifier discriminates among all categories. For the highest predicted class, the two-class SVMs (trained on that class and its confused categories in one-versus-rest setting) are invoked to confirm or overturn that prediction. However, the first level classifier may prematurely categorize an instance into a class that the SVMs experts are unable to recover. Also, since the set of SVMs need to be learned for *each* category, this method is not appropriate for the many-class and multi-label learning setting, and the experiments were run on relatively small data sets (news groups and Reuters, with 20 and 60 classes). In our work, EDGE learns an expert per compound concept with respect to the confusion graph (a generalized notion of the confusion matrix in the multi-label setting). Since the number of compound concepts tends to be significantly lower than the total number of concepts, and each generated problem is smaller, training time is significantly reduced in the second stage. Furthermore, rather than relying on the decision of a single expert, EDGE aggregates the decision of the generalist and all activated experts, which possesses the merits of ensemble learning mentioned earlier.

6. CONCLUSION AND FUTURE WORK

In the setting of text categorization with tens of thousands of categories, building a highly accurate and efficient many-class classifier is essentially a very difficult task. Rather than attempting to build a monolithic classifier, EDGE decomposes the learning problem and builds a number of smaller and more accurate classifiers, which we call experts in this

paper. Experts utilize fewer classes and lower dimensionality and may be complementary to each other. The discovery of these experts is data driven: learning resources are placed on distinguishing concepts that tend to be confused by the generalist. By combining the decisions of the experts and the generalist, EDGE demonstrates substantial improvement in classification and ranking accuracies in large-scale text categorization datasets. There are several opportunities for improvement. First, experts can be combined with consideration of their ratings (i.e. their relative prediction performance on data). Also, learning a generalist in conjunction with the experts, possibly in an online manner, may prove useful. We also want to explore the use of statistical tests of significance in constructing the confusion graphs and in computing compound concepts.

7. ACKNOWLEDGMENTS

The authors would like to thank Rosie Jones, Preston McAfee and Ron Brachman from Yahoo! Research for valuable comments and reviewers for helpful feedback.

8. REFERENCES

[1] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *Proc. of 13th ACM international conference on Information and knowledge management (CIKM)*, 2004.

[2] R. Caruana and A. Niculescu-Mizil. Data mining in metric space: an empirical analysis of supervised learning performance criteria. In *Proc. of 10th ACM SIGKDD Conference*, 2004.

[3] V. R. Carvalho and W. W. Cohen. Single-pass online learning: performance, voting schemes and online feature selection. In *Proc. of 12th ACM SIGKDD Conference*, 2006.

[4] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7, 2006.

[5] K. Crammer and Y. Singer. A family of additive online algorithms for category ranking. *Journal of Machine Learning Research*, 3:1025 – 1058, 2003.

[6] S. Dumais and H. Chen. Hierarchical classification of web content. In *Proc of 23th ACM SIGIR conf.*, 2000.

[7] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, 1998.

[8] A. Esuli, T. Fagni, and F. Sebastiani. TreeBoost.MH: A boosting algorithm for multi-label hierarchical text categorization. In *Proc of 13th Int'l Conf on String Processing and Information Retrieval (SPIRE)*, 2006.

[9] Y. Freund and R. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Computer & System Sciences*, 55(1), 1997.

[10] Y. Freund, R. Schapire, Y. Singer, and M. Warmuth. Using and combining predictors that specialize. In *ACM Symp. on Theory of Computing (STOC)*, 1997.

[11] S. Godbole, S. Sarawagi, and S. Chakrabarti. Scaling multi-class support vector machines using inter-class confusion. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 513–518, 2002.

[12] W. Hersh, C. Buckley, T. Leone, and D. Hickam. OHSUMED: An interactive retrieval evaluation and new large test collection for research. In *Proc. of the 17th ACM SIGIR Conference*, pages 192 – 201, 1994.

[13] S. Keerthi and D. DeCoste. A modified finite newton method for fast solution of large scale linear SVMs. *Journal of Machine Learning Research*, 6, 2005.

[14] D. Lewis, Y. Yang, T. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5, 2004.

[15] T. Liu, Y. Yang, H. Wan, H. Zeng, Z. Chen, and W. Ma. Support vector machines classification with a very large-scale taxonomy. *KDD Explorations*, 2005.

[16] O. Madani and M. Connor. Large-scale many-class learning. In *SIAM Conf on Data Mining (SDM)*, 2008.

[17] O. Madani, W. Greiner, D. Kempe, and M. R. Salavatipour. Recall systems: Effcient learning and use of category indices. In *Proceedings of the 11th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2007.

[18] O. Madani and J. Huang. On updates that constrain the features' connections during learning. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD)*, 2008.

[19] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67:026126, 2003.

[20] J. Rennie, L. Shih, J. Teevan, and D. Karger. Tackling the poor assumptions of naive bayes text classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pages 616 – 623, 2003.

[21] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *J. Machine Learning Research*, 5, 2004.

[22] F. Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 56(6):386–408, 1958.

[23] R. Schapire and Y. Singer. Improved boosting algorithms using confidence-rated prediction. *Machine learning*, 37(1):297 – 336, 1999.

[24] F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 2002.

[25] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, 1996.

[26] K. Tumer and J. Ghosh. Robust combining of disparate classifiers through order statistics. *Pattern Analysis & Applications*, 5(2):189 – 200, 2002.

[27] D. J. Watts and S. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[28] D. H. Wolpert. Stacked generalization. *Neural networks*, pages 241 – 259, 2002.

[29] L. Xu, A. Krzyzak, and C. Y. Suen. Methods of combining multiple classifiers and their applications to handwriting recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 22(3):418 – 435, 1992.

[30] Y. Yang. An evaluation of statistical approaches to text categorization. *J. of Information Retrieval*, 1999.

[31] Y. Yang, J. Zhang, and B. Kisiel. A scalability analysis of classifiers in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR)*, 2003.