Large Scale Multi-Label Classification via MetaLabeler

Lei Tang
Arizona State University
Tempe, AZ 85287
L.Tang@asu.edu

Suju Rajan
Yahoo! Inc.
Sunnyvale, CA 94089
suju@yahoo-inc.com

Vijay K. Narayanan
Yahoo! Inc.
Sunnyvale, CA 94089
vnarayan@yahoo-inc.com

ABSTRACT

The explosion of online content has made the management of such content non-trivial. Web-related tasks such as web page categorization, news filtering, query categorization, tag recommendation, etc. often involve the construction of multi-label categorization systems on a large scale. Existing multi-label classification methods either do not scale or have unsatisfactory performance. In this work, we propose MetaLabeler to automatically determine the relevant set of labels for each instance without intensive human involvement or expensive cross-validation. Extensive experiments conducted on benchmark data show that the MetaLabeler tends to outperform existing methods. Moreover, MetaLabeler scales to millions of multi-labeled instances and can be deployed easily. This enables us to apply the MetaLabeler to a large scale query categorization problem in Yahoo!, yielding a significant improvement in performance.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database applications—*Data Mining*; H.4.m [**Information Systems**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

MetaLabeler, Multi-label Classification, Large Scale, Hierarchical Classification, Query Categorization, Meta Model

1. INTRODUCTION

The rapid growth of online content has made it critical to develop methods that facilitate easy storage, searching, and retrieval of such content. One such methodology involves placing web data into hierarchies or catalogs. While human editors can manually categorize data into taxonomies, such editorial effort does not scale well to the size of online content. Furthermore, different types of online content such as HTML documents, images, video, social network graphs, etc. require domain-specific handling. Hence, the development of a scalable and automated categorization system is important.

The problem of developing accurate text-classifiers has been well studied in the machine learning communities. One such approach is Support Vector Machine (SVM) classification. SVMs have been shown to have exceptional performance in the categorization of large quantities of data in very high-dimensional spaces [11]. Two aspects of online data, however, make it difficult to directly apply SVM categorizers in that domain.

- The data is multi-class. Textual data such as web-pages can be classified into many categories. The Yahoo! Directory, for instance, has a few million nodes into which web-pages are categorized.
- The data is multi-label. Data such as queries, web-pages, products, and even users can have more than one label. For instance, a query such as "Jaguar" can have a label set containing *Animals*, *Software*, and *Automotive* classes.

A commonly used approach to address multi-class classification is to decompose the multi-class problem into a number of binary problems. For instance, in the One-vs-Rest (also known as One-vs-All) approach a decision boundary is learned independently for each individual class by treating all the instances belonging to that class as positive and all the other instances as negative. During scoring, a data point is assigned to the class for which it has a maximum positive score. Despite its simplicity, the One-vs-Rest approach yields good performance in practice. In fact, when tuned carefully, this method has been shown to outperform other complex output-space decomposition methods and even multi-class SVMs [20].

One-vs-Rest approach could be extended to multi-label prediction as well by choosing as the set of classes for which the data point has a positive score. A drawback with this approach is that some of the binary classification problems can be highly imbalanced [9, 22]. The default threshold of SVM decision function is typically optimized for accuracy and could lead to a biased prediction toward the negative class. One possible solution is to tune a threshold based on cross-validation according to some performance evaluation metric such as F1-measure. This thresholding strategy is demonstrated to improve the performance of the vanilla One-vs-Rest SVM [15, 7]. However, learning thresholds for each class in large scale applications is computationally expensive as there are typically a very large number of classes.

Recently, there has been a lot of efforts in addressing the multi-label problem [26]. However, most of the proposed approaches [8, 25, 28, 33, 34] are either classifier-specific

WWW 2009, April 20–24, 2009, Madrid, Spain.
ACM 978-1-60558-487-4/09/04.

or do not scale to the size of web data. For instance, a real-world query categorization problem we have[1] consists of roughly 1.5 million queries in more than 6000 categories, which requires a scalable and efficient multi-label classifier. In this work, we propose a simple and effective approach called *MetaLabeler* which in conjunction with an One-vs-Rest SVM handles multi-label prediction easily. The proposed approach automatically identifies the right number of relevant labels for each instance without the need for expensive cross-validation. Moreover, our model does not rely on the underlying classifier and scales well to the size of the data as well to the number of underlying classes. We conduct extensive experiments to compare our model with other baseline methods both on benchmark data and a real-world query categorization application. It can be seen that the proposed solution tends to outperform other methods, especially when the underlying classifier makes use of meta-information about the classes, such as class hierarchies.

2. RELATED WORK

Multi-label classification studies the problem in which a data instance can have multiple labels [26]. This phenomenon prevails in a variety of tasks including document filtering, text categorization [15], web mining [17], tag recommendation in social bookmarking system [12], etc. Many approaches have been proposed to address multi-label classification, including margin-based methods, structural SVMs [25], parametric mixture models [28], k-nearest neighbors [33], maximum entropy models [8, 34], and ensemble methods [29, 27]. Most of these methods utilize the relationship between multiple labels for collective inference. There have also been attempts to find a low-dimensional subspace shared among multiple labels [10, 31]. Unfortunately, most of the proposed methods do not scale to our needs with millions of data instances in thousands of categories.

For large scale multi-label categorization system, One-vs-Rest approach, which builds binary classifiers independently for each category, is still widely used [15, 16]. A post-processing step that can set thresholds according to some evaluation measure like macro-F1 or micro-F1 avoids the expensive training procedure of the above mentioned approaches. Yang [30] studied three different thresholding strategies: rank-based cut, proportion-based cut, and score-based cut and compared the pros and cons for each strategy. An exploration of various cross-validation schemes and heuristics for score-based approach is detailed in [7].

In real-world systems, data is typically organized into hierarchies [3, 17, 24, 23]. Utilizing the hierarchy has been shown to yield performance improvements over the flat classifiers [4, 14]. Margin-based methods that take into consideration the similarities of different categories in the taxonomy have been proposed [1, 25, 21]. Alternatively, smoothing methods that utilize the dependency information in the hierarchy have also been shown to yield performance improvements [18, 19]. However, the computational complexity involved in training and deploying these models is a deterrent to their adoption in large scale categorization systems.

3. MULTI-LABEL PREDICTION

The main focus of this work, is to classify the textual data found on the World Wide Web into a set of pre-defined categories or classes. The set of classes can be large and may or may not have dependencies, such as a class hierarchy, defined on it. Let K denote the set of classes associated with a given categorization task. Each class has a set of data points associated with it, say $\mathbf{X} \in R^{N \times M}$ where M represents the dimensionality of feature space. Each data point can have one or more class labels associated with it. Let $\mathbf{y} \in \{0,1\}^K$ be a class-vector corresponding to a data point $\mathbf{x}$. Given a new data point $\mathbf{x}$ that was unseen at the time of training, the goal is to be able to identify (with good precision and recall) the set of classes $\hat{\mathbf{y}}$ associated with $\mathbf{x}$, such that $\hat{\mathbf{y}}$ is *close* to the true class-vector $\mathbf{y}$.

We break the above problem into two steps.

- Step 1: To obtain a ranking of class membership for each instance $\mathbf{x}$ into the set of K classes.
- Step 2: To predict the number of top classes to be returned from the ranking.

3.1 Obtaining Class Membership Ranking

Given a data point $\mathbf{x}$, we first want to obtain a vector $\mathbf{f}(\mathbf{x}) \in R^K$ where the score $f_i(x)$ denotes the class membership of $\mathbf{x}$ in class i where $i = 1, \cdots, N$. While there exists several ranking classifiers, we choose the One-vs-Rest SVM because of its superior performance in the multi-class text categorization task [11, 20]. The scores output by the binary SVMs are then used to get a ranking of the class membership for $\mathbf{x}$. Given data instances $\mathbf{X} \in R^{N \times M}$ with class-label vector $\mathbf{Y} \in \{0,1\}^{N \times K}$, a decision function $\mathbf{f}$ is built for each of the K classes by considering the instances belonging to a class as positive and all the other instances as negative. Restricting ourselves to linear SVMs, each of the decision functions predicts a score for an instance $\mathbf{x}$ as below:

$$f_k(x) = \mathbf{w_k}^T \mathbf{x} + b_k \tag{1}$$

where $\mathbf{w_k}$ is the weight vector and b_k is the bias term associated with class C_k. Given a test instance $\mathbf{x}$, we classify it as category C_k if $f_k(\mathbf{x}) > 0$. The score vector $\mathbf{f}(\mathbf{x})$ can also be used to rank the class membership of $\mathbf{x}$ in the K categories.

3.2 Metalabeler: Predicting the Top-n Classes

To determine the top-n classes from the score vector, we learn a function from the data to the number of labels. This involves two steps.

- Constructing the meta dataset.
- Learning a meta-model.

We use the toy example in Table 1 to illustrate the construction of meta data. Consider the data in Table 1a consisting of 4 instances in 4 categories. The label of the meta data (shown in Table 1b) is the number of labels for each instance in the raw data. For example, $\mathbf{x}_2$ belongs to C_1, C_3 and C_4 in the raw data. The corresponding meta label is then 3. Assume that the raw data $\mathbf{x}$ has been transformed into the meta data $\phi(\mathbf{x})$, a mapping from the meta data to the meta label then could be learned. There are several approaches for constructing the meta dataset:

Content-based MetaLabeler The simplest approach is to use the raw data as it is. That is,

$$\phi(\mathbf{x}) = \mathbf{x} \tag{2}$$

[1] More details discussed in Section 6.1.

Table 1: Construction of Meta Data

Data	Labels
$\mathbf{x_1}$	C_1, C_3
$\mathbf{x_2}$	C_1, C_2, C_4
$\mathbf{x_3}$	C_2
$\mathbf{x_4}$	C_2, C_3

(a) Raw Data

Meta Feature	Meta Label
$\phi(\mathbf{x_1})$	2
$\phi(\mathbf{x_2})$	3
$\phi(\mathbf{x_3})$	1
$\phi(\mathbf{x_4})$	2

(b) Meta Data

Score-based MetaLabeler The score vector of each label $\mathbf{f}(\mathbf{x})$ can be considered as a summary of the instance associated with the set of labels and could be used as the meta data. Hence,

$$\phi(\mathbf{x}) = [(f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_K(\mathbf{x})] \tag{3}$$

where $f_i(\mathbf{x})$ is the prediction score for each category.

Rank-based MetaLabeler Intuitively, if the gap among the prediction scores of top-ranking categories is small, it is likely that the data instance belongs to all the top-ranking classes. This suggests another method for constructing meta data based on ranked prediction scores:

$$\phi(\mathbf{x}) = \psi(f_1(\mathbf{x}), f_2(\mathbf{x}), \cdots, f_K(\mathbf{x})) \tag{4}$$

where ψ is a function that sorts the scores and $\psi(\mathbf{x})$ is a vector of sorted scores.

Based on the constructed meta data, we learn a mapping function from the meta features to the meta label ($\mathbf{y_{ML}}$), that is, the number of labels. A natural choice is regression. However, the prediction of the meta-model in such case is not necessarily an integer (say, 2.5 labels), and we need to determine whether 2 or 3 labels are actually associated with the data instance. To avoid this dilemma, we consider the meta learning as a multi-class classification problem, and once again resort to the One-vs-Rest strategy as it is efficient, scalable yet effective. The detailed algorithm is summarized in Figure 1, in which we have an optional parameter λ to set the maximum number of labels for prediction. The training data might include some outliers with a large class label set, the utility of which is suspect, and the cut-off can help remove such outliers.

3.3 Prediction

As for prediction, the One-vs-Rest SVM works in conjunction with the MetaLabeler as follows:

1. Given an instance $\mathbf{x}$, obtain its class membership ranking based on the score vector $\mathbf{f}(\mathbf{x})$.

2. Construct the input to the meta model $\phi(\mathbf{x})$ for each $\mathbf{x}$ using either the content-based, score-based or rank-based approach.

3. Predict the number of labels y_{ML} for instance $\mathbf{x}$ based on the meta-model.

4. Pick the top y_{ML} highest scoring categories as the labels for prediction.

In this section, we have proposed a simple, yet efficient solution to the problem of multi-label classification. Note that the proposed algorithm is *classifier agnostic* as both the ranking classifier as well as the MetaLabeler can be designed with any powerful classifier or ranking algorithm.

```
Input:  Multi-label data ($\mathbf{X}$, $\mathbf{Y}$);
        Maximum number of labels: $\lambda$;
Output: Meta-model $M(\mathbf{x})$.
1. Construct meta data $\phi(\mathbf{X})$ from raw data $\mathbf{X}$;
2. Construct meta label $\mathbf{y_{ml}} = |\mathbf{Y}|$;
3. Set the maximum number of labels:
     $\mathbf{y_{ML}}(\mathbf{y_{ML}} > \lambda) = \lambda$;
4. Learn $M(\mathbf{x})$ via One-vs-Rest SVM on $(\phi(\mathbf{X}), \mathbf{y_{ML}})$;
```

Figure 1: Algorithm for MetaLabeler

Our proposed MetaLabeler, especially the score-based MetaLabeler might appear similar to stacking. Stacking [5] is an ensemble method in which meta-model is learned based on the probability distributions of multiple heterogeneous classifiers. However, the motivation of MetaLabeler is to predict the number of labels for each instance whereas stacking is proposed to combine different classifiers.

3.4 Other Approaches

In this section, we sketch other approaches that have been proposed for the problem of multi-label classification. All of these approaches assume that there exists a classifier that can provide scores indicative of class memberships. The scores that are produced are then used to determine the set of labels that are to be predicted for the data instance $\mathbf{x}$. These thresholding strategies typically fall into three categories [30]:

Rank-based cut (RCut) RCut assigns labels based on the ranking of class labels for each data point $\mathbf{x}$. The ranking could be obtained by sorting the prediction scores. Based on the ranking, $\mathbf{x}$ is assigned to the top n categories, where n is a user specified parameter. Typically, n is set to the average length of class labels in the training data. Suppose instances in a dataset have 3.5 labels on average, n could be set to either 3 or 4.

Proportion-based cut (PCut) In this method, for each category C_k, the test instances are sorted by the scores for C_k and the top t instances are predicted as belonging to class C_k. Here, t is computed based on the prior probability of C_k estimated on the training data. PCut requires the prediction scores for all the test data and hence is seldom used in real-world applications where test data arrives in an online fashion.

Score-based local optimization (SCut) SCut tunes the threshold for each category based on improving a user defined performance measure. The SCut strategy was optimized for different evaluation criteria in [7].The basic idea is to split the training data into different folds and cycle through each category to tune the threshold based on the performance on the validation data.

Based on extensive experiments with the k-nearest neighbor classifier on 5 different text corpora, Yang [30] concluded that the SCut approach tends to overfit and is unstable across different datasets while PCut is more stable. However, PCut requires access to all of the test data to make a prediction, ruling out its utility in most real-world applications. A comparison of SCut versus RCut for hierarchical

SVMs with the different thresholding strategies showed that SCut outperforms RCut in terms of macro and micro F1 [17].

Comparing MetaLabeler with the above three approaches, MetaLabeler can be considered as local RCut with n optimized for each individual data instance. RCut mentioned above presents a scheme to select the top-ranking labels based on the parameter n, whose value is fixed globally. On the contrary, the number of labels in MetaLabeler is determined dynamically by the supplied data instance.

4. EXPERIMENTAL METHODOLOGY

To show the efficacy of MetaLabeler, extensive experiments were conducted on two benchmark datasets. In this section we explain the evaluation measure used, the experimental setup, baseline methods, and details of the data used in our experiments.

4.1 Evaluation Measures

The commonly used performance evaluation criteria for multi-label classification are exact match ratio, micro-F1, and macro-F1 [7]. Given test data $\mathbf{X} \in R^{N \times M}$, let $\mathbf{y_i}, \hat{\mathbf{y}}_\mathbf{i} \in \{0,1\}^K$ be the true label set and the predicted label set for instance $\mathbf{x_i}$.

Exact Match Ratio is defined as

$$\textit{Exact Match Ratio} = \frac{1}{N}\sum_{i=1}^{N} I[\mathbf{y_i} = \hat{\mathbf{y}}_\mathbf{i}] \qquad (5)$$

where I is the indicator function. Exact match ratio is essentially the accuracy used for binary classification extended to multi-label case. As seen in the above equation, exact match ratio does not consider partial match between the true labels and predictions. The alternatives which count the partial match are macro-F1 and micro-F1.

Macro-F1 is the F1 averaged over categories.

$$\textit{Macro-F1} = \frac{1}{K}\sum_{k=1}^{K} F_1^k \qquad (6)$$

For a category C_k, the precision (P^k) and the recall (R^k) are calculated as,

$$P^k = \frac{\sum_{i=1}^N y_i^k \hat{y}_i^k}{\sum_{i=1}^N \hat{y}_i^k}; \quad R^k = \frac{\sum_{i=1}^N y_i^k \hat{y}_i^k}{\sum_{i=1}^N y_i^k}.$$

Then F1 measure, defined as the harmonic mean of precision and recall is computed as follows:

$$F_1^k = \frac{2P^kR^k}{P^k + R^k} = \frac{2\sum_{i=1}^N y_i^k \hat{y}_i^k}{\sum_{i=1}^N y_i^k + \sum_{i=1}^N \hat{y}_i^k}$$

Micro-F1 is computed using the equation of F_1^k and considering the predictions as a whole. More specifically, it is defined as

$$\textit{Micro-F1} = \frac{2\sum_{k=1}^K \sum_{i=1}^N y_i^k \hat{y}_i^k}{\sum_{k=1}^K \sum_{i=1}^N y_i^k + \sum_{k=1}^K \sum_{i=1}^N \hat{y}_i^k}. \qquad (7)$$

According to the definition, macro-F1 is more sensitive to the performance of rare categories while micro-F1 is affected more by the major categories. In our experiments, all the three measures are examined carefully.

Table 2: Characteristics of Yahoo! Data

Dataset	N	M	K	AveL	MaxL
Arts	7441	17973	19	1.62	13
Business	9968	16621	17	1.55	11
Computers	12371	25259	23	1.48	17
Education	11817	20782	14	1.44	10
Entertainment	12691	27435	14	1.40	14
Health	9109	18430	14	1.60	12
Recreation	12797	25095	18	1.41	17
Reference	7929	26397	15	1.15	10
Science	6345	24002	22	1.37	11
Social	11914	32492	21	1.24	10
Society	14507	29189	21	1.64	17

4.2 Baseline Methods

Three versions of MetaLabeler are tested: content-based ($Meta_c$), score-based ($Meta_s$), and rank-based ($Meta_r$). We do not set any cut-off for maximum number of labels for MetaLabeler. We also compare our method with other thresholding strategies:

- Vanilla SVM (SVM_v). The One-vs-Rest SVM without any post-processing procedure. During prediction, all the labels with a positive score are selected.

- RCut with n equal to the average number of labels per instance. Normally, the average number of labels is not an integer. Hence, n can be set as either

  $$n = \lfloor AveLabel \rfloor \quad \text{or} \quad n = \lceil AveLabel \rceil.$$

  The two versions are denoted as $RCut_c$ and $RCut_a$, respectively to indicate that the methods are conservative and aggressive in predicting the number of labels.

- SCut tuned based on micro-F1 ($SCut_i$) or macro-F1 ($SCut_a$). We follow the threshold tuning presented as Algorithm 1 in [7]. Essentially, the threshold tuning process cycles over each category to optimize the desired performance measure. The final threshold is the average of 5-fold cross validation. Two other algorithms (SCutFBR.1 and SVM.1) presented in [7] require a heuristic *fbr* value to be provided by users or selected from a set of values based on another layer of cross-validation which could be computationally expensive.

We use linear SVMs [6] as the base classifier. The parameter C in SVM is selected from a proper set of values. Note that the vanilla SVM's performance is quite sensitive to this parameter. We originally used the default value ($C = 1$) for the SVMs and it yielded extremely low accuracy on some of the datasets. Typically, $C = 200$ to 1000 gives much better result for SVM_v.

4.3 Datasets

The first benchmark dataset was extracted from the Yahoo! directory. This multi-topic web categorization data was used in [28, 10].The dataset consists of 11 independently compiled subsets, each of which corresponds to a top-level category in the Yahoo! Directory. In effect, this gives us 11 independent multi-label datasets. The data was preprocessed following the methodology in [10]. We removed

Table 3: Performance on Yahoo! multi-topic web page categorization data. The three horizontal blocks denote the exact match ratio, micro-F1 and macro-F1 respectively. Each column denotes one data set. The entries in bold denote the best one in each column. All the results are averaged over 30 runs. The last column *Ave* shows the results averaged over 11 data sets.

	Arts	Busi.	Comp.	Edu.	Enter.	Health	Rec.	Ref.	Sci.	Social	Society	Ave
SVM_v	27.78	**64.41**	41.07	30.67	43.97	47.70	37.16	45.40	35.06	55.14	34.92	42.12
$RCut_c$	34.50	54.41	47.13	38.95	52.63	43.19	48.09	**60.95**	48.74	64.01	**40.79**	48.49
$RCut_a$	9.82	19.05	7.21	8.45	6.38	19.73	7.23	1.94	5.36	4.67	5.69	8.68
$SCut_a$	24.58	60.39	37.68	28.40	44.78	46.37	38.42	46.20	37.40	52.51	10.26	38.82
$SCut_i$	27.40	63.61	41.14	30.91	45.83	46.66	39.92	48.53	39.20	56.71	33.67	43.05
$Meta_c$	35.85	64.15	**48.94**	**41.02**	**53.55**	**52.79**	**50.28**	59.28	49.48	**64.05**	38.60	**50.73**
$Meta_s$	**36.71**	59.40	45.96	39.70	51.70	50.50	49.38	58.78	**50.05**	63.41	36.07	49.24
$Meta_r$	20.50	59.03	36.76	22.94	42.76	39.49	32.28	46.91	35.73	48.16	17.62	36.56
SVM_v	45.35	79.95	56.02	49.15	59.02	67.88	52.39	60.17	51.36	67.43	50.35	58.10
$RCut_c$	47.02	71.13	55.94	50.05	59.36	62.30	56.63	**65.78**	56.59	69.40	52.03	58.75
$RCut_a$	48.35	69.68	53.31	50.39	53.67	64.78	51.90	53.50	51.34	57.35	50.33	54.96
$SCut_a$	47.72	79.16	56.53	51.74	61.89	69.36	56.06	61.87	56.00	66.46	41.92	58.97
$SCut_i$	49.04	**80.19**	58.63	52.54	62.25	69.49	56.53	63.26	57.06	69.10	**53.96**	61.10
$Meta_c$	50.34	79.59	**59.09**	**53.38**	**63.21**	**69.82**	**59.05**	65.25	**58.80**	**70.11**	53.12	**61.98**
$Meta_s$	**50.79**	78.23	58.26	52.93	61.88	69.36	58.67	65.09	59.17	69.80	51.43	61.42
$Meta_r$	39.83	71.89	44.57	43.95	52.12	56.53	43.63	54.69	47.82	56.45	34.95	49.68
SVM_v	34.00	49.50	31.79	40.69	49.32	59.77	44.96	39.48	41.69	36.82	31.07	41.74
$RCut_c$	34.92	30.30	26.13	41.47	46.13	43.16	46.20	50.29	46.42	38.14	31.31	39.50
$RCut_a$	39.58	38.15	32.87	44.06	46.39	56.58	47.35	40.43	46.00	36.16	36.83	42.22
$SCut_a$	40.37	**53.28**	**40.00**	45.56	**55.10**	**62.98**	51.15	47.63	50.26	**45.35**	**38.86**	**48.23**
$SCut_i$	37.98	51.60	36.42	44.54	54.01	62.23	50.12	44.92	48.88	42.33	36.23	46.30
$Meta_c$	40.20	51.46	37.58	**46.04**	55.07	62.06	**52.90**	**49.86**	**51.33**	44.79	35.69	47.91
$Meta_s$	**40.40**	49.26	36.95	45.14	52.74	61.45	52.55	49.78	52.03	43.46	34.14	47.08
$Meta_r$	31.84	39.87	27.09	35.95	43.77	44.75	37.95	40.92	40.97	34.58	24.71	36.58

those categories with less than 100 web pages, words occurring less than 5 times and web pages with no topics. Each web page was represented as a bag of words using the TF-IDF encoding and was normalized to unit length. Table 2 summarizes some characteristics of the data. N, M, K represents the number of instances, dimensions and categories, respectively. AveL denotes the average number of labels per instance, and MaxL denotes the maximum number of labels for an instance. While the data has, less than 2 labels on average per data point, one instance could have up to 17 labels. We randomly sampled 2000 instances for training and the remaining are used for testing. This process was repeated 30 times, and the averaged result is reported.

The second dataset used was the subsets of RCV1 [15]. The data is publicly available at LibSVM site[2]. As in the case of the Yahoo! dataset there are 5 subsets, each with 3000 data instances for training and 3000 for testing, with in total 101 categories. On average, each instance has around 3.22 labels and the maximum number of labels per instance is 14. This dataset, different from previous data, is highly imbalanced with the largest category having around 1400 instances and the smallest with only 1 instance in training data. Note that 5-fold cross validation is not applicable for the rare classes. Thresholds for $SCut_i$ and $SCut_a$, in such cases, were selected based on the tuning on the training data.

5. EXPERIMENTAL RESULTS

The performance of the different multi-label strategies is shown in Tables 3 and 4. Each horizontal block in the table corresponds to an evaluation measure that is exact match ratio, micro-F1, and macro-F1 respectively.

5.1 Performance on Yahoo! Data

As observed in Table 3, $RCut_c$ and $RCut_a$ do not necessarily improve over the performance of vanilla SVM. Specifically $RCut_a$ which assigns labels aggressively is very likely to make mistakes as indicated by the extremely low exact match ratio score. $SCut_a$, $SCut_i$, $Meta_c$ and $Meta_s$, on the other hand, achieve better performance than the vanilla SVM. However, $Meta_r$, performs worse than SVM_v.

A strong pattern observed in Table 3 is that $Meta_c$ tends to outperform other methods in terms of exact match ratio and micro-F1. The performances of $SCut_a$ and $Meta_c$ are comparable with respect to macro-F1, as $SCut_a$ is the winner in 6 out of the 11 datasets while $Meta_c$ wins on 4. Note that the *SCut* strategies require cross-validation for each class to choose thresholds whereas the MetaLabeler needs to be tuned only once while training and is much more efficient. The efficiency difference between these two methods will be discussed in detail later.

The MetaLabeler could be considered as a local *RCut* with the number of labels determined automatically. To examine the difference, we plot the content-based MetaLabeler ($Meta_c$) versus the conservative/aggressive *RCut* in Figure 2 in terms of both micro-F1 and macro-F1. The x-axis denotes the 11 datasets. It is evident that our method by determining the number of labels dynamically for each instance, achieves superior performance in most cases. A similar trend is observed for exact match ratio as well.

Comparing the different meta labeling strategies, we observe that the content-based and score-based MetaLabeler

[2] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/multilabel.html

Table 4: Performance on RCV1-subsets. The three blocks are the exact match ratio, Micro-F1, and Macro-F1 respectively. Ave denotes the averaged result over 5 subsets.

	Sub_1	Sub_2	Sub_3	Sub_4	Sub_5	Ave
SVM_v	39.93	40.07	41.67	39.37	38.10	39.83
$RCut_c$	29.23	29.67	30.43	29.43	28.93	29.54
$RCut_a$	3.13	3.90	3.63	3.47	2.97	3.42
$SCut_i$	14.03	26.17	21.07	20.47	33.60	23.07
$SCut_a$	14.97	24.13	15.70	20.93	33.90	21.93
$Meta_c$	**43.23**	**43.60**	**44.30**	**42.93**	**42.20**	**43.25**
$Meta_s$	26.13	33.20	33.00	36.23	18.40	29.39
$Meta_r$	36.07	32.40	39.83	31.67	29.43	33.88
SVM_v	72.88	73.69	73.91	72.98	72.84	73.26
$RCut_c$	72.58	72.40	73.05	72.81	71.83	72.53
$RCut_a$	69.15	69.01	69.30	69.52	68.27	69.05
$SCut_i$	61.03	69.58	64.22	66.19	67.61	65.73
$SCut_a$	63.53	70.56	63.54	68.81	70.57	67.40
$Meta_c$	**74.52**	**74.67**	**75.13**	**74.92**	**74.04**	**74.66**
$Meta_s$	66.53	73.18	72.97	72.23	67.50	70.48
$Meta_r$	73.07	70.33	73.17	66.54	65.95	69.81
SVM_v	35.01	35.75	34.12	33.04	34.47	34.48
$RCut_c$	39.31	38.43	37.50	40.16	37.46	38.57
$RCut_a$	41.79	40.36	39.81	43.19	39.62	40.95
$SCut_i$	33.06	35.03	32.59	32.86	33.70	33.45
$SCut_a$	39.07	40.84	37.16	40.18	41.50	39.75
$Meta_c$	43.47	**43.57**	**42.13**	**42.98**	**41.31**	**42.69**
$Meta_s$	32.28	40.84	41.97	38.25	41.20	38.91
$Meta_r$	**44.90**	39.55	38.23	28.34	27.87	35.78

are comparable with the former having a slight edge. $Meta_r$ performs even worse than the vanilla SVM. This is possibly due to the fact that it is important to have the prediction scores associated with the class labels. This information is lost after sorting thereby leading to a bad meta-model.

5.2 Performance on RCV1

The performance on each RCV1 subset and the averaged result are summarized in Table 4. For this dataset in almost all the cases the content-based MetaLabeler yields the best performance. The vanilla SVM after cross-validation gives a reasonably good performance after careful parameter tuning of the SVMs. The threshold based strategies, $RCut$ and $SCut$ deteriorate the performance of SVM_v mostly, except that $RCut_a$ and $SCut_a$ yield a higher macro-F1 by giving more predictions than necessary to boost the recall of each category. Only our meta-model consistently improves the baseline SVM_v in terms of all the three measures. In contrast to the Yahoo! directory data, SCut strategies do not work well here. This is due to the presence of rare classes with only one positive instance in the training data. Threshold tuning which was done on the training data possibly overfit the data and becomes quite unreliable for those classes with few instances. As before, the content-based MetaLabeler is more robust and demonstrates superior performance over other approaches.

5.3 Robustness of the MetaLabeler

The content-based MetaLabeler works well in both benchmark datasets. However, the performance of the score-based and rank-based MetaLabeler is quite unstable. This could be due to the effect of unbounded scores in SVM prediction. Hence, we also experimented with building a meta-model based on calibrated prediction scores via isotonic regression as detailed in [32]. Simply stated, isotonic regression maps the unbounded SVM scores into the $[0, 1]$ range. Figure 3 shows the micro-F1 of various meta-models on 11 Yahoo! web datasets. Cal_s and Cal_r represent the performance of $Meta_s$ and $Meta_r$ after calibration. It can be seen that calibration resulted in a worse performance for $Meta_s$ while improving $Meta_r$. This makes sense as the gap between the ranked scores is more accurate after calibration. Calibration in general has been reported to improve the classification performance, especially the rank-related measure, over uncalibrated scores [2]. This could also be observed for our rank-based MetaLabeler. Unfortunately, the calibration does not help for score-based MetaLabeler. Since the scores within certain interval are calibrated to the same value, this might become problematic for those prediction scores close to the interval boundary. So it is best to use raw content as meta features because this would cause no loss of information. Such a content-based MetaLabeler outperforms all the other variants of MetaLabeler across almost all the multi-label datasets.

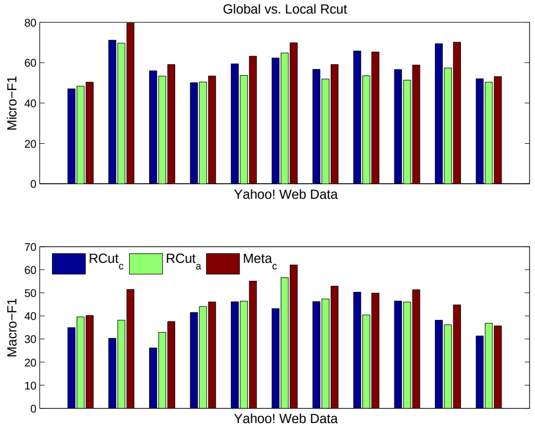

Figure 2: $Meta_c$ vs. $RCut$

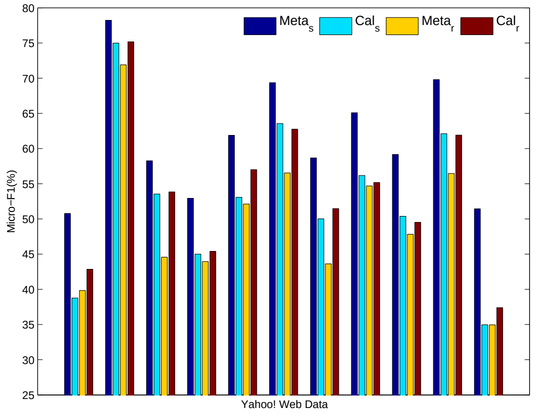

Figure 3: Meta-Models with Calibration

5.4 Bias with MetaLabeler

On both datasets the average number of labels tends to be smaller, with relatively a smaller subset of the data having a class label set of much larger size. This imbalanced distribution in the number of class labels leads to a conservative prediction bias with the MetaLabeler.

For instance, consider the category Society in Yahoo! web page data as it contains the largest number of samples. The distribution of the number of labels for one trial of the test data is presented in Table 5, with l denotes the number of labels, $Truth$ and $Pred$ denote the number of instances that have l labels in the test data and predictions respectively. As can be seen, the majority of the data have only one label. Though there are some instances which contain more than 5 labels, this is a "rare class" in the meta-model learning. The imbalanced distribution in the training data leads to a meta-model that favors predicting lesser number of labels. This could be easily verified by the $Pred$ row in Table 5. The number of instances with only 1 predicted label is much larger than the true number and for all other cases the predicted number is less than the truth. This indicates that our MetaLabeler picks more labels only if the instance has a strong signal of having more than 1 label. This bias helps maintain a relatively high precision in the resultant performance.

Table 5: Distribution of Number of Labels in *Society*

l	1	2	3	4	5	6	7	8+
$Truth$	7638	3180	1098	328	119	27	13	104
$Pred$	9819	2195	389	68	19	3	3	11

5.5 Scalability Analysis

It was observed on Yahoo! web page categorization data, the $SCut_i$ and $SCut_a$ methods demonstrate performance comparable to that of $Meta_c$ in terms of micro-F1 and macro-F1 (shown in Table 3). However, SCut strategy requires cross-validation to select the optimal thresholds. If cross-validation is not applicable, SCut could over-fit the training data as shown in Table 4. MetaLabeler, in contrast, tends to be more reliable.

Here we use the Yahoo! Society data again as an example to examine the scalability of SCut and MetaLabeler. In particular, we increase the number of training samples gradually from 1000 to 10000 and record the computation time on an AMD 4400+ desktop. The computation time for each method is plotted in Figure 4, where SVM_v denotes the computation time to construct One-vs-Rest SVM and $SCut_a$ and $Meta_c$ denote the *additional* computation time after the vanilla SVM is built. As $SCut_i$ and $SCut_a$ share the same time complexity approximately, we only present $SCut_a$ to make the figure legible.

The computation times of the different methods are linear with respect to number of samples. However, $SCut_a$ increases much faster than $Meta_c$. In particular, when the training samples scale to 10000, the additional time needed to construct MetaLabeler is 148 seconds whereas $SCut_a$ is ten times as much at 1487 seconds. Therefore, our content-based MetaLabeler outperforms SCut with respect to classification performance and efficiency, and is more favorable in large scale multi-label categorization system.

Note that the analysis so far examined the scalability of SCut and MetaLabeler only in terms of training samples. In a large scale categorization system, the number of classes as well as samples could be huge. Recall that MetaLabeler builds a meta-model from the content to the number of labels. So as long as the maximum number of labels per instance is relatively small, which is typically true in practice, the training of MetaLabeler should be efficient. SCut, however, has to sort the prediction scores of all the training data for each class which operation scales linearly with respect to the total number of classes. Also, SCut further requires cross-validation which only increases the computational complexity. As we will show in a real-world application in the next section, SCut requires days to train while MetaLabeler finishes in two hours.

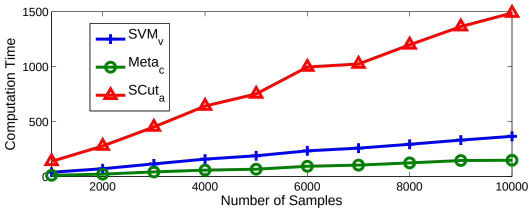

Figure 4: Computation Time on Yahoo! Society data

6. METALABELER IN LARGE SCALE CATEGORIZATION SYSTEM

The content-based MetaLabeler was shown to perform well on reasonably sized multi-label classification tasks. We claim the MetaLabeler is an efficient and reliable solution that should work well even when presented with very large datasets. In the following subsections we validate this claim with experiments on a real-world large scale query categorization system.

6.1 Query Categorization

Internet portals and search engines aim to deliver content and ads to users that are relevant to their interests. In such applications, it is very useful to categorize the user search queries into relevant nodes in a taxonomy of user interests. Such a taxonomy has been constructed by human experts in Yahoo!, with nodes in the taxonomy spanning a large variety of topics including for example, finance to home garden decoration. These nodes are organized in a hierarchy based on the semantic similarity of the user interests. There are in total 6,433 interest categories in this taxonomy, distributed over 8 levels. The nodes at deeper levels correspond to more specific user interests.

A user query can typically be categorized into multiple nodes in the taxonomy. For instance, a query "0% interest credit card no transfer fee" is labeled as belonging to the following three categories:

Financial Services/Credit, Loans and Debt/Credit/Credit Cards/Credit Card Balance Transfers

Financial Services/Credit, Loans and Debt/Credit/Credit Cards/Low-Interest Credit Cards

Financial Services/Credit, Loans and Debt/Credit/Credit Cards/Low-No-Fee Credit Cards

In this example, labeling the query as *Financial Services/Credit, Loans and Debt/Credit/Credit Card* is acceptable; however, this ignores the extra information in the query viz., that the user is really interested in a balance transfer card with no interest. In this taxonomy, there are 14 different daughter nodes under the parent node *Financial Services/Credit, Loans and Debt/Credit/Credit Cards* and hence the specific interest of the user can indeed be captured in this taxonomy.

Our goal is to build machine learned classifiers into this hierarchical taxonomy such that the queries can be classified

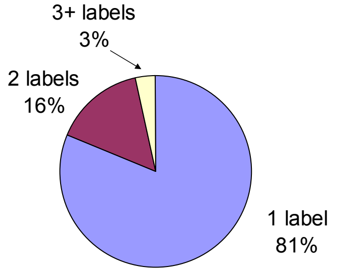

Figure 5: Label Distribution on Query Data

Table 6: Distributions with respect to depths of the taxonomy. #nodes represents the number of categories of that depth in the taxonomy; #instances denotes the number of data instances with at least one label at the depth; AveLabel denotes the average number of labels per query.

Depth	#nodes	#instances	AveLabel
1	21	70,506	1.0018
2	231	274,298	1.0396
3	1027	498,753	1.0810
4	2021	403,622	1.0926
5	1812	274,767	1.1086
6	1001	114,983	1.0617
7	272	28,776	1.0677
8	48	3,024	1.0754

accurately into relevant classes. For training purposes, we use a data set of 1.5 million unique queries that have been manually labeled into nodes in this taxonomy by human editors. Figure 1 shows the distribution of the number of labels per query in this dataset. About 20% of the queries in this dataset have more than 1 label as shown in Figure 5. The maximum number of labels per query is 26, and the average number of labels per query is 1.23. Table 6 shows the number of categories, queries and labels with respect to depths in the taxonomy. It is observed that most of the instances reside at levels 2 to 6.

6.2 MetaLabeler with Flat Model

In this subsection, we build a one-vs-rest SVM for the problem directly without leveraging any hierarchical information during training. For evaluation, we split the data into two parts: roughly 1 million for training and 0.5 million for testing. Each query is represented using bag of words, resulting in total 120,000 features. Here, we compare content-based MetaLabeler with SCut only as it is the winning thresholding strategy on the benchmark data. We used a linear SVM based One-vs-Rest approach for the construction of the meta-model. We emphasize the fact that both operations of obtaining ranking scores and meta-label prediction are classifier agnostic. We use linear SVMs as our underlying model because of its ability to handle large quantities of training data and ease of deployment in online, real-world systems. Evaluation of recently developed multi-class SVMs [13] in our problem is an ongoing process.

The vanilla SVM training alone takes 12 hours to finish. Note that one query can have at most 26 labels and so the MetaLabeler requires no more than 26 extra binary SVMs, which adds an additional 2 hours. Due to the large size of the data, SCut training involves many I/O operations. For instance, the size of the prediction score file for the training data in all the categories is 47GB large and takes 3 hours to output. After spinning the machine for approximately 27 hours, we finally obtain the threshold tuned on the training data. Keep in mind that in cross validation, we have to train vanilla SVM and then conduct the SCut tuning in each trial, which is way too expensive.

Table 7: Performance of MetaLabeler on Flat Model

	Micro-F1			Macro-F1		
Depth	Flat	Meta	SCut	Flat	Meta	SCut
1	90.03	**91.31**	64.44	87.38	**89.09**	58.69
2	87.38	**89.15**	62.01	75.79	**79.69**	38.06
3	83.28	**86.23**	58.17	64.72	**69.13**	22.59
4	80.03	**83.53**	48.48	60.35	**64.79**	15.26
5	77.57	**81.71**	42.22	58.78	**64.17**	13.84
6	77.00	**81.35**	40.06	56.72	**61.31**	9.81
7	76.31	**82.63**	34.09	56.67	**65.59**	13.18
8	73.03	**78.41**	25.69	49.10	**54.87**	7.71

Table 7 shows the performance at each depth in the taxonomy. To calculate the performance, a query belonging to a category node is considered associated with all the ancestor nodes as well. In the table, "Flat" denotes the vanilla One-vs-Rest SVM, "Meta" denotes the content-based MetaLabeler. "SCut" is tuned according to the corresponding measure. The entries in bold denotes the best model.

Evidently, with MetaLabeler, we are able to improve the flat model in terms of both micro-F1 and macro-F1 across all levels in the taxonomy. SCut, despite the expensive computational cost, has very low recall. The threshold have been over-estimated as many queries are predicted with no labels. This could be due to the inter-dependency of the classes and extremely small number of samples in some nodes for training. In contrast, our proposed MetaLabeler is quite reliable and efficient.

6.3 MetaLabeler with Hierarchical Model

The query dataset has meta-information about the classes, in the form of a class hierarchy. Leveraging such hierarchical information has been shown to yield better performance than classifiers that ignore such information [4, 14, 16]. In the following sub-sections, we investigate the effectiveness of the MetaLabeler in a hierarchical model. SCut, due to its expensive training time without guaranteeing better performance, is skipped in this setting.

6.3.1 Hierarchical Classification Model

In order to better leverage the class hierarchy information, we build a hierarchical classifier as follows:

- An One-vs-Rest SVM is built at each node based on the training data at the subtree of that node.
- For prediction, the set of classifiers is traversed in a top-down fashion. Starting from the root node, a data instance is first classified into the set of the level 1 nodes. The winning node in level 1 then decides the set of nodes that are to be explored at level 2. The data is pushed down the tree until a leaf node is reached.
- To account for the data instances that belong to an internal node, we construct an artificial "parent" leaf-node under the internal node. If a query is classified into such "parent" leaf-node, the prediction output is then the corresponding parent node.

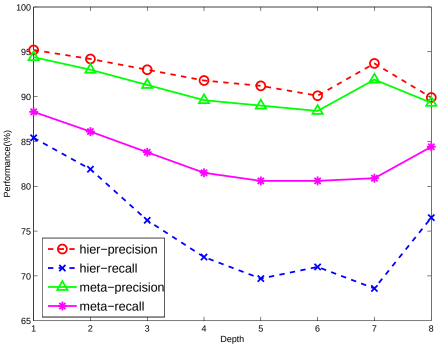

Figure 6: MetaLabeler vs. Hierarchical Model

The proposed MetaLabeler could be easily embedded into the training and prediction phase of a hierarchical classifier. A content-based MetaLabeler was built at each node in the taxonomy. During prediction, we explore multiple paths, depending on the prediction of the MetaLabeler, using either depth-first or breadth-first search. The final output of the hierarchical model is a set of labels for each query.

6.3.2 Hierarchical Multi-Label Prediction

The micro-averaged precision and recall of MetaLabeler versus the original hierarchical model is presented in Figure 6. The precision, as shown in the figure, actually decreases by $1-2\%$. In contrast, the improvement of recall is drastic. The recall at deeper levels increases from 70% to over 80%. We also show the improvement in terms of micro-F1 and macro-F1 in Table 8. With MetaLabeler, both measures improve by $1-9\%$ at different levels. Even though we take a top-down multi-path exploration method for prediction, our model typically predicts a limited number of labels due to the predictive bias of MetaLabeler. This effectively avoids the explosion of number of labels for hierarchical prediction as indicated by the high precision ($> 90\%$).

A closer look at the performance of the hierarchical model in Table 8 versus the flat model performance in Table 7 reveals that hierarchical model consistently yields a higher micro-F1 and macro-F1 at deeper levels. The best performance is achieved when MetaLabeler is utilized together with the hierarchical information of classes.

The content-based MetaLabeler works pretty well in our query categorization problem as well as the benchmark data. In order to get some intuition into this behavior, we checked the features with the highest weight in the meta-model at the root node of the hierarchical model. Words such as *overstock.com*, *blizzard*, *threading*, etc., rank among the top features. The word *blizzard*, for instance, is associated with labels like *Video Game*, *Computer Game Software*, *Fast Food Restaurant*, and *Weather Information*. The content-based MetaLabeler is able to extract such generic features that are typically associated with multiple labels and weight them appropriately.

7. CONCLUSIONS & FUTURE WORK

Multi-label classification appears in various web-related tasks such as web page categorization, query-categorization, and user profiling. Typically, the categories are organized into a hierarchy. Unfortunately, classifying data into the set of relevant nodes in the taxonomy is not an easy task.

Table 8: MetaLabeler on Hierarchical Model.

	Micro-F1		Macro-F1	
Depth	Hier	Meta	Hier	Meta
1	90.03	**91.25**	87.29	**88.84**
2	87.62	**89.42**	74.91	**78.86**
3	83.77	**87.39**	65.40	**70.57**
4	80.77	**85.36**	63.96	**70.62**
5	79.01	**84.59**	65.06	**72.91**
6	79.42	**84.32**	67.83	**73.68**
7	79.21	**86.05**	67.22	**76.65**
8	82.66	**86.78**	63.75	**70.40**

In this work, we propose *MetaLabeler* to determine the number of relevant labels automatically. Three versions of MetaLabeler are presented: content-based, score-based and rank-based. Based on extensive experiments on benchmark data and our large scale query categorization problem, we conclude that the content-based MetaLabeler in conjunction with a classifier that ranks class memberships tends to outperform other multi-label prediction systems. The proposed solution scales to millions of samples and can easily be incorporated into hierarchical classification systems. The MetaLabeler requires neither expensive cross-validation nor human interaction. The simplicity of training and scoring coupled with its performance make the MetaLabeler a viable solution for the construction of large scale multi-label classification system.

One limitation of MetaLabeler is that it lacks the flexibility to be tuned for user-specified precision/recall levels. A possible solution that could address this problem is to construct a meta-model that takes into account the label confidence in the training data. We are also planning to apply MetaLabeler to social bookmarking system with large number of labels.

8. ACKNOWLEDGMENTS

Most of the work was done when Lei Tang was an intern in Yahoo! SDS Data Mining and Research Group during summer 2008. We thank Varun Chandola for the implementation of hierarchical categorization system.

9. REFERENCES

[1] L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM '04: Proceedings of the thirteenth ACM international conference on Information and knowledge management*, pages 78–87, New York, NY, USA, 2004.

[2] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pages 161–168, New York, NY, USA, 2006. ACM.

[3] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan. Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *The VLDB Journal*, 7(3):163–178, 1998.

[4] S. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR '00: Proceedings of the 23rd annual international ACM SIGIR conference on*

Research and development in information retrieval, pages 256–263, New York, NY, USA, 2000.

[5] S. Džeroski and B. Ženko. Is combining classifiers with stacking better than selecting the best one? *Mach. Learn.*, 54(3):255–273, 2004.

[6] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.

[7] R.-E. Fan and C.-J. Lin. A study on threshold selection for multi-label classication. 2007.

[8] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 195–200, New York, NY, USA, 2005. ACM Press.

[9] N. Japkowicz and S. Stephen. The class imbalance problem: A systematic study. *Intell. Data Anal.*, 6(5):429–449, 2002.

[10] S. Ji, L. Tang, S. Yu, and J. Ye. Extracting shared subspace for multi-label classification. In *KDD '08: Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 381–389, New York, NY, USA, 2008. ACM.

[11] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Heidelberg et al., 1998.

[12] I. Katakis, G. Tsoumakas, and I. Vlahavas. Multilabel text classification for automated tag suggestion. In *Proceedings of the ECML/PKDD 2008 Discovery Challenge*, 2008.

[13] S. S. Keerthi, S. Sundararajan, K.-W. Chang, C.-J. Hsieh, and C.-J. Lin. A sequential dual method for large scale multi-class linear svms. In *KDD*, pages 408–416, 2008.

[14] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, pages 170–178, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[15] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *J. Mach. Learn. Res.*, 5:361–397, 2004.

[16] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, 7(1):36–43, 2005.

[17] T.-Y. Liu, Y. Yang, H. Wan, Q. Zhou, B. Gao, H.-J. Zeng, Z. Chen, and W.-Y. Ma. An experimental study on large-scale web categorization. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, pages 1106–1107, New York, NY, USA, 2005. ACM.

[18] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML '98: Proceedings of the Fifteenth International Conference on Machine Learning*, pages 359–367, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc.

[19] K. Punera and J. Ghosh. Enhanced hierarchical classification via isotonic smoothing. In *WWW '08: Proceeding of the 17th international conference on World Wide Web*, pages 151–160, 2008.

[20] R. Rifkin and A. Klautau. In defense of one-vs-all classification. *JMLR*, 5:101–141, 2004.

[21] J. Rousu, C. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *J. Mach. Learn. Res.*, 7:1601–1626, 2006.

[22] L. Tang and H. Liu. Bias analysis in text classification for highly skewed data. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 781–784, 2005. IEEE Computer Society.

[23] L. Tang, H. Liu, J. Zhang, N. Agarwal, and J. J. Salerno. Topic taxonomy adaptation for group profiling. *ACM Trans. Knowl. Discov. Data*, 1(4):1–28, 2008.

[24] L. Tang, J. Zhang, and H. Liu. Acclimatizing taxonomic semantics for hierarchical content classification. In *KDD '06: Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 384–393, 2006.

[25] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector machine learning for interdependent and structured output spaces. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 104, New York, NY, USA, 2004. ACM.

[26] G. Tsoumakas and K. Ioannis. Multi label classification: An overview. *International Journal of Data Warehousing and Mining*, 3:1–13, 2007.

[27] G. Tsoumakas and K. Ioannis. Random k-labelsets: An ensemble method for multilabel classification. In *ECML*, 2007.

[28] N. Ueda and K. Saito. Parametric mixture models for multi-labeled text. In *NIPS*, pages 721–728, 2002.

[29] R. Yan, J. Tesic, and J. R. Smith. Model-shared subspace boosting for multi-label classification. In *KDD '07: Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 834–843, 2007.

[30] Y. Yang. A study of thresholding strategies for text categorization. In *SIGIR '01: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 137–145, New York, NY, USA, 2001. ACM.

[31] K. Yu, S. Yu, and V. Tresp. Multi-label informed latent semantic indexing. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 258–265, New York, NY, USA, 2005. ACM.

[32] B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699, 2002.

[33] M.-L. Zhang and Z.-H. Zhou. Ml-knn: A lazy learning approach to multi-label learning. *Pattern Recogn.*, 40(7):2038–2048, 2007.

[34] S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *SIGIR*, 2005.