Mach Learn (2012) 88:47–68
DOI 10.1007/s10994-011-5270-7

# Multilabel classification with meta-level features in a learning-to-rank framework

**Yiming Yang · Siddharth Gopal**

Received: 1 October 2010 / Accepted: 2 November 2011 / Published online: 15 December 2011


**Abstract** Effective learning in multi-label classification (MLC) requires an appropriate level of abstraction for representing the relationship between each instance and multiple categories. Current MLC methods have focused on learning-to-map from instances to categories in a relatively low-level feature space, such as individual words. The fine-grained features in such a space may not be sufficiently expressive for learning to rank categories, which is essential in multi-label classification. This paper presents an alternative solution by transforming the conventional representation of instances and categories into meta-level features, and by leveraging successful learning-to-rank retrieval algorithms over this feature space. Controlled experiments on six benchmark datasets using eight evaluation metrics show strong evidence for the effectiveness of the proposed approach, which significantly outperformed other state-of-the-art methods such as Rank-SVM, ML-kNN (Multi-label kNN), IBLR-ML (Instance-based logistic regression for multi-label classification) on most of the datasets. Thorough analyses are also provided for separating the factors responsible for the improved performance.

**Keywords** Multilabel classification · Learning to rank

Editors: Grigorios Tsoumakas, Min-Ling Zhang, and Zhi-Hua Zhou.

The work is supported, in part, by the National Science Foundation (NSF) under grant IIS_0704689. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

Y. Yang
Language Technologies Institute & Machine Learning Department, Carnegie Mellon University, Pittsburgh, USA
e-mail: yiming@cs.cmu.edu

S. Gopal (✉)
Language Technologies Institute, Carnegie Mellon University, Pittsburgh, USA
e-mail: sgopal1@andrew.cmu.edu

## 1 Introduction

Multi-label classification (MLC) refers to the problem of instance labeling where each instance may have more than one correct label. MLC has a broad range of applications. For example, a news article could belong to multiple topics, such as politics, finance and economics, and also could be related to China and USA as the regional categories. An image could have flower as the object type, yellow and red as the colors, and outdoor as the background category. A computer trouble report could be simultaneously related to a networking issue, a software problem, an urgency-level category, a regional code, and so on. MLC is technically challenging as it goes beyond the scope of well-studied binary classifiers, such as two-class Support Vector Machines (SVM), Naive Bayes probabilistic classifiers, etc. Approaches to MLC typically reduce the problem into two sub-problems: the first is learning to rank categories with respect to each input instance, and the second is learning to place a threshold on each ranked list for a yes/no decision per category. The first sub-problem is the most challenging part and therefore has been the central focus in MLC. A variety of approaches has been developed and can be roughly divided into two types: binary-classifier based methods versus global optimization methods, and the latter can be further divided into model-based and instance-based methods. Binary-classifier based methods are the simplest. A representative example is to use a standard SVM (binary-SVM) (Vapnik 2000) to learn a scoring function for each category independently from the scoring functions for other categories. Other kinds of binary classifiers could also be used for such a purpose, such as logistic regression, Naïve Bayes probabilistic classifiers, boosting algorithms, neural networks etc. In the testing phase, the ranked list of categories is obtained for each test instance by scoring each category independently with its binary classifier and then sorting the scores. Binary-classifier based methods have been commonly used due to their simplicity, but also have been criticized for the lack of global optimization in category scoring. These methods are common baselines in benchmark evaluations of stronger methods for MLC.

Elisseeff and Weston (2001) proposed a large-margin approach, namely Rank-SVM, which is a representative example of model-based methods. Unlike conventional binary SVM which maximizes the margin for each category independently, Rank-SVM maximizes the sum of the margins for all categories, conditioned on partial-order constraints. That is, ranking the relevant categories of each instance higher than the irrelevant categories is an explicit optimization criterion in this method. The scoring function is parameterized by the weights of input features for every category. Experiments by the authors of Rank-SVM showed improved performance of this method over binary SVM in gene classification on a micro-array dataset (namely 'the yeast dataset'). Zhang and Zhou (2007) proposed an instance-based approach which is named as Multi-label k-Nearest Neighbor (ML-kNN). Cheng and Hüllermeier (2009) proposed another variant called Instance-Based Logistic Regression (IBLR). Multi-label versions of kNN have been studied in text categorization for a long time and commonly used as strong baselines in benchmark evaluations (Creecy et al. 1992; Yang 1994, 1999). ML-kNN and IBLR are relatively new variants which are similar to each other in the sense that both use Euclidean distance to identify the k nearest neighbors for each test instance, but also differ from each other in how the local probabilities are estimated for categories. ML-kNN gives an equal weight to each label occurrence in the neighborhood of the input instance while IBLR varies the weight of each label according to how distant it is to the test instance. Further, ML-kNN assumes independence among category occurrences while IBLR explicitly models pairwise dependencies among categories using logistic regression. IBLR also makes combined use of instance-based features (such as the similarity score of each neighbor) and conventional features (such as words in the test document) in the logistic regression). The evaluations by the authors of ML-kNN showed

its superior performance over Rank-SVM, BoosTexter (Schapire and Singer 2000) and Adaboost.MH (Schapire and Singer 1999) , and the experiments by the authors of IBLR showed further performance improvement by IBLR over the results of ML-kNN on multiple datasets (Cheng and Hüllermeier 2009).

MLC methods, including the ones discussed above, have been focused on learning-to-map using low level features. Binary SVM and Rank-SVM for text categorization, for example, use words in the document vocabulary as typical features. Rank-SVM for image classification, as another example, uses colors and intensity of pixels as typical features. Although these low-level features are directly observable and handy to process they may not be sufficiently expressive for characterizing the instance-category relationship, and may not provide a proper granularity level for systems to effectively learn the instance-to-category mapping. An open question for research therefore is: Can we transfer the lower-level features into higher-level features with which MLC classifiers or learning-to-rank algorithms would work more effectively? We answer this question in this paper by the following means:

1. We propose a general framework that supports automated transformation of a conventional instance representation (such as a bag of words per instance and a set of training instances per category) into meta-level features, enables a broad range of learning-to-rank algorithms in information retrieval (IR) to be leveraged for category ranking in MLC, and invokes supervised learning for instance-based threshold optimization.
2. We instantiate our framework with several state-of-the-art learning-to-rank algorithms, including RankSVM-IR (Joachims 2002), SVM-MAP (Yue and Finley 2007), LambdaRank (Burges et al. 2007) and ListNet (Cao et al. 2007), to illustrate the generality of our framework and to examine the effectiveness of these instantiations.
3. We conduct controlled experiments on multiple benchmark datasets to evaluate our approach in comparison with other state-of-the-art methods in MLC, including Rank-SVM, ML-kNN, IBLR and Binary SVM. Our approach outperforms the other methods significantly on most of the datasets, with p-values at the 5% level or smaller.

The rest of the paper is organized as follows. Section 2 describes our new approach. Section 3 outlines the design of our experiments. Section 4 presents the main results with discussions. Section 5 summarizes our findings and future work.

This paper is the extended version of our paper in ACM SIGIR 2010 (Gopal and Yang 2010). Here we present the same new framework for MLC, with larger and more thorough empirical evaluations and in-depth analyses for a broader range of learning-to-rank algorithms.

## 2 Method

Our approach consists of three components: supervised extraction of meta-level features, learning to rank categories based on meta-level features, and learning to optimize the threshold over ranked lists on a per-instance basis.

### 2.1 Meta-level feature extraction

We define the meta-level features for MLC based on both the original representation of each input instance and a training set of labeled instances. Let $\mathbf{X}$ be the input space of all possible instances, $\mathbf{C}$ be the set of categories over which the ranked lists of categories are formed, $m$ be the number of categories, and $\phi(x, c)$ be the meta-level representation (a vector) of instance $x \in \mathbf{X}$ with respect category $c \in \mathbf{C}$. The desirable properties of the meta-level representation are the following:

1. $\phi(x, c)$ should be highly informative about the relation of instance $x$ to category $c$, and discriminative in separating the positive instances from the negative instances of category $c$.
2. The features in vector $\phi(x, c)$ should be automatically computable given instance $x$ and a training set of labeled instances.
3. The transformed training data should allow a broad range of learning-to-rank algorithms to be used for category ranking per test instance in MLC.

Based on these criteria we define a set of vectors for representing each instance ($x$) based on its $k$-nearest-neighbor distances from the training instances in category $c$ as the following:

- $\phi_{L_2}(x, c) = (d_{L_2}(x, \hat{x}_1), d_{L_2}(x, \hat{x}_2), \ldots, d_{L_2}(x, \hat{x}_j), \ldots, d_{L_2}(x, \hat{x}_k))$, a $k$-dimensional vector, where $\hat{x}_j \in kNN_{L_2}(x, c)$ is the $j^{th}$ nearest neighbor ($j = 1, 2, \ldots, k$) of $x$ among those instances which belong to $c$ in the training set, and $d_{L_2}(x, \hat{x}_j) = \|x - \hat{x}_j\|_2$ is the $L_2$ distance between the two vectors.
- $\phi_{L_1}(x, c) = (d_{L_1}(x, \hat{x}_1), d_{L_1}(x, \hat{x}_2), \ldots, d_{L_1}(x, \hat{x}_j), \ldots, d_{L_1}(x, \hat{x}_k))$, a $k$-dimensional vector, where $\hat{x}_j \in kNN_{L_1}(x, c)$ is the $j^{th}$ nearest neighbor ($j = 1, 2, \ldots, k$) of $x$ among those instances which belong to $c$ in the training set, and $d_{L_1}(x, \hat{x}_j) = \|x - \hat{x}_j\|_1$ is the $L_1$ distance between the two vectors.
- $\phi_{cos}(x, c) = (d_{cos}(x, \hat{x}_1), d_{cos}(x, \hat{x}_2), \ldots, d_{cos}(x, \hat{x}_j), \ldots, d_{cos}(x, \hat{x}_k))$, a $k$-dimensional vector, where $\hat{x}_j \in kNN_{cos}(x, c)$, and $d_{cos}(x, \hat{x}_j)$ is the cosine similarity distance between the two vectors.
- $\phi_{mod}(x, c) = (d_{L_2}(x, \bar{x}_c), d_{cos}(x, \bar{x}_c))$, a 2-dimensional vector, where $\bar{x}_c$ is the centroid (vector average) of all instances that belong to category $c$.

We define $\phi(x, c)$ as the concatenation of the above vectors:

$$\phi(x, c) = [\phi_{L_2}(x, c), \phi_{L_1}(x, c), \phi_{cos}(x, c), \phi_{mod}(x, c)] \tag{1}$$

Finally, the full meta-level representation of instance $x$ with respect to all categories is defined as:

$$\mathbf{z_i} = (\phi(x_i, c_1), \phi(x_i, c_2), \ldots, \phi(x_i, c_m)) \tag{2}$$

Of course the meta-level features above are not exhaustive with respect to all possibly informative features; rather they are a set of concrete examples to illustrate the principle underlying our approach. Note that vector $\phi(x, c)$ has a dimensionality $3k + 2$ and each $\phi_{cos}(x, c)$, $\phi_{L_1}(x, c)$, $\phi_{L_2}(x, c)$ have dimensionality $k$.[1] Parameter $k$ can be tuned on a held-out validation dataset; $k$ is typically in the range from 10 to 100.

The meta-level features make a combined use of local information (through kNN-based features) about each instance as well as global information (through category centroids) about output categories in the training set and represent a feature space for constructing discriminative patterns across categories. Figure 1 illustrates the concept geometrically in a 2-D space. For simplicity we only plot $\phi_{L_2}(x, c)$ in this graph.

[1] When $k$ is larger than the number of training instances in a category, we repeatedly append the value of the largest NN distance to the meta-level feature representation, making the resulting vector with dimension fixed $k$.

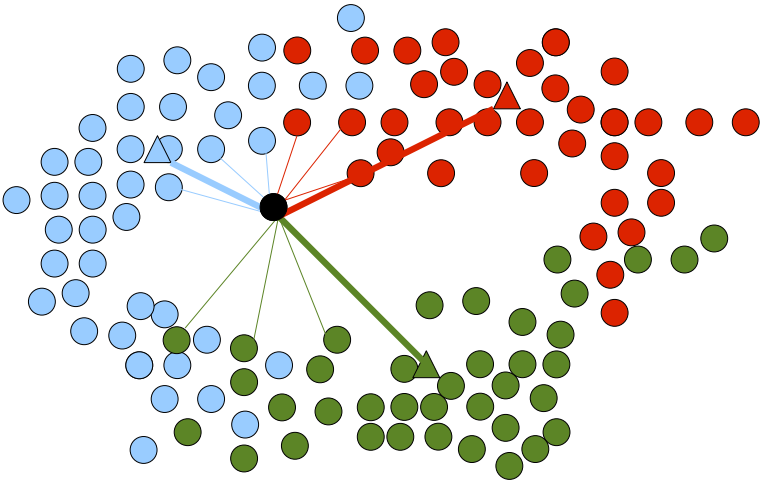

**Fig. 1** The feature representation of one particular instance, the *black dot in the center*, is shown in relation to each of the 3 categories. Each category is represented using its positive examples (*points in the same color*) in the training set and its centroid (*triangle*). The relation between the instance and a category is represented using the distance to the category centroid (shown by *thick lines*) and the distance to each of the k nearest neighbors (shown by the *thin lines*) from the same category. Note that the number of nearest neighbors is set to 3

## 2.2 Learning to rank categories for MLC

In analogy to the standard notation in the learning-to-rank literature, we define $\mathbf{Z}$ as the space of all possible instances which are represented using meta-level features, $\mathbf{Y}$ as the space of all possible ranked lists of categories, and $D = \{(z_i, y_i)\}_{i=1,..n}$ as a training set of $n$ pairs of $z_i \in \mathbf{Z}$ and $y_i \in \mathbf{Y}$. Our goal is to find the optimal mapping $f : \mathbf{Z} \rightarrow \mathbf{Y}$ given $D$. In principle, any learning to rank method can be used to learn the ranking function. Learning to rank methods so far are designed for ranking documents with respect to *ad-hoc* queries where *ad-hoc* means that queries can be any combination of words and are not fixed in advance. Most learning-to-rank algorithms in text retrieval rely on a shared representation between queries and documents, i.e., a bag-of-words per query and per document. Such a shared representation facilitates a natural way to induce features for discriminating the relevance of query-document pairs. Typical features include the count of shared terms in the query and the document, the length of the document, the BM25 score of the document (Qin et al. 2010) etc. In order to apply any learning-to-rank retrieval algorithm to MLC in general, we need to find discriminative features to represent instance-category pairs. The meta-level features $\phi(x, c)$ we introduced in the previous section are exactly designed for such a purpose, allowing a broad range of learning-to-rank algorithms in IR to be deployed for MLC.

We follow the well-established learning-to-rank paradigm; in the training phase, a learning to Rank method is used to optimize model parameters with respect to partial-order preference between pairs of relevant and non-relevant category labels for each training document. In the testing phase, the trained system produces a ranked list of categories (in complete order) for each test document. A variety of learning-to-rank algorithms have been developed recently for ad-hoc retrieval, optimizing different loss functions and model selection criteria. For example, SVM-MAP (Yue and Finley 2007) is a large-margin approach

designed to maximize the Mean Average Precision (MAP) of ranked lists, AdaRank.NDCG (Xu and Li 2007) is a boosting based method designed to optimize Normalized Discounted Cumulative gain (NDCG) of ranked lists. Methods which focus on other optimization criteria include MCRank (Li et al. 2007), FRank (Tsai et al. 2007), ListNet (Cao et al. 2007), LambdaRank (Burges et al. 2007). All these methods can be used in our framework as "plug-in" learning algorithm in principle, to perform the task of ranking categories given an instance. We choose a subset of them as concrete examples for the instantiation of our approach and for thorough experiments (Sects. 4 and 5). These choices are briefly outlined below:

RankSVM-IR was proposed by Joachims (2002) for ranking documents given a query, originally named RankSVM. We rename it as 'RankSVM-IR' in order to make a distinction from the 'RankSVM' proposed by Elisseeff and Weston (2001) for multi-label classification. RankSVM-IR is a large-margin approach that enforces partial order constraints among documents given a query, i.e., a relevant document should be ranked higher than any irrelevant document. Its objective function optimizes the ROC-Area from a classification point of view. RankSVM-IR has been a strong baseline in benchmark evaluations of learning-to-rank methods for IR. We name its application to MLC with meta-level features as MLC-RankSVM-IR.

SVM-MAP was proposed by Yue and Finley (2007), as another large-margin approach and a specific instantiation of the more general SVM-struct (Tsochantaridis et al. 2006). SVM-MAP is designed to optimize ranked lists with respect to the Mean Average Precision (MAP) which has been a conventional metric in the evaluations of retrieval systems. It has shown superior performance over RankSVM-IR and other methods when MAP is the choice of metric. We call its application to MLC with meta-level features as MCL-SVM-MAP.

LambdaRank was proposed by Burges et al. (2007), and has become one of the most popular and widely used learning-to-rank methods. It uses a neural network type of learning, and can be easily adapted to work with several IR metrics such as *MAP*, *NDCG*, Mean Reciprocal Rank etc. (Donmez et al. 2009). It has exhibited state-of-the-art performance in IR evaluations. We name its application to MLC with meta-level features as MLC-LambdaRank.

ListNet is proposed by Cao et al. (2007) as another neural network based method. It uses a probabilistic model to define its loss function and its authors have shown improved performance over RankSVM-IR, RankBoost (Freund et al. 2003), RankNet (Burges et al. 2005), etc., in evaluations on benchmark datasets (Qin et al. 2010). We name its application to MLC with meta-level features as MLC-ListNet.

### 2.3 Learning to threshold for MLC

In order to enable the system to make classification decisions in MLC, we need to apply a threshold to the ranked list of categories for each test instance. A variety of thresholding strategies have been studied in the literature (Elisseeff and Weston 2001; Yang 2001), and the choice of thresholding strategy may depend on the nature of classifiers. For binary-SVM, the natural and common choice is to set the threshold to zero, while for probabilistic binary classifiers (such as logistic regression or Naïve Bayes classifiers), the default choice of threshold is 0.5. However, learning-to-rank methods are very different from binary classifiers, and many of them do not produce probabilistic scores. Obviously, neither zero nor 0.5 nor any fixed constant is the optimal threshold on the ranked list of categories given an instance. We take an alternative strategy which was originally proposed

by Elisseeff and Weston (2001) for Rank-SVM for MLC, where the threshold is optimized on a per-instance basis, conditioned on each ranked list. The system uses a training set to learn a linear mapping $g$ from an arbitrary ranked list of categories to a threshold as:

$$g : L \rightarrow T$$

where $L \subseteq \mathbf{R}^m$ is the space of all possible vectors of system-scored categories, and $T \subseteq \mathbf{R}$ is the space of all possible thresholds. The optimal mapping is defined as the linear-least-squared-fit (LLSF) solution given a training set of ranked lists with the optimal threshold per list. The optimal threshold given a list is defined as the threshold that minimizes some pre-defined metric (such as error-rate, $F_1$ etc.). In our experiments we chose the optimization criteria to select the threshold that minimizes the sum of false positives and false negatives. The training set can be automatically generated by (1) using a learning-to-rank algorithm to rank (score) all categories conditioned on each input instance, and (2) finding the optimal threshold on the ranked list of categories with respect to pre-specified metric. The LLSF function is learned from such a training set and then is applied in the testing phase, to the system-scored categories conditioned on each test instance. The categories whose scores are above or at the threshold receive a yes decision, and the categories whose scores are below the threshold receive a no decision. As a modification of the original learning-to-threshold method, we rescale the scores of categories for each test instance to make them sum to one; we have found the score normalization improves performance in our experiments.

## 3 Evaluation design

### 3.1 Datasets

We use six datasets in our experiments, namely *emotion, scene, yeast, citeseer, Reuters21578*, and *vowel*. Table 1 summarizes the statistics of the datasets. These datasets form a representative sample across different fields and they vary in training-set size and feature-space size. All the datasets have been used in previous evaluations of multi-label or multi-class classification methods, with conventional train-test splits. We follow such conventions in order to make our results comparable to the previously published ones. We briefly outline these datasets below:

*Emotions* is a multi-label audio dataset (Trohidis et al. 2008), in which each instance is a song, indexed using 72 features such as amplitude, beats per minute etc. The songs have been classified under six moods such as sad/lonely, relaxing/calm, happy/pleased, amazed/surprised, angry/aggressive and quiet/still.

*Scene* is an image classification dataset (Boutell et al. 2004). The images are indexed using a set of 294 features which decompose each image into smaller blocks and represent the color of each block. The images are classified based on the scenery (Beach, Sunset etc.) they portray.

*Yeast* dataset (Elisseeff and Weston 2001) is a biomedical dataset. Each instance is a gene, represented using a vector whose features are the micro-array expression levels under various conditions. The genes are classified into 14 different functional classes.

*Citeseer* is a set of research articles we collected from the Citeseer web site. Each article is indexed using the words in its abstract as the features, with a feature-set size of 14,601. We use the top level of 17 categories in the Citeseer classification hierarchy as the labels

**Table 1** Dataset statistics

| Dataset Name | Training Size | Testing Size | #Categories | Avg #Categories per instance | #Features |
|---|---|---|---|---|---|
| *Vowel* | 528 | 462 | 11 | 1 | 10 |
| *Emotions* | 391 | 202 | 6 | 1.87 | 72 |
| *Scene* | 1211 | 1196 | 6 | 1.07 | 294 |
| *Yeast* | 1500 | 917 | 14 | 4.24 | 103 |
| *Citeseer* | 5303 | 1326 | 17 | 1.26 | 14601 |
| *Reuters-21578* | 7770 | 3019 | 90 | 1.23 | 18637 |

in this dataset, and randomly split 80% of the corpus into training and the rest as testing instances. The dataset is publicly available.[2]

*Reuters-21578* is a benchmark dataset in text categorization evaluations. The instances are Reuters news articles during the period 1987 to 1991, and labeled using 90 topical categories. We follow the same train-test split used by Yang (2001).

*Vowel* is a multi-class audio vowel recognition dataset (Ganapathiraju et al. 1998). There are 11 types of vowel sounds which form the output class labels. The training set consists of 528 speech data frames from eight speakers and the test set consists of 462 frames from seven speakers. The speech data was digitized and is represented using 10 features.

In our experiments, we use the *scene, emotions, yeast, citeseer, Reuters21578* datasets for comparative evaluation of our approach and other state-of-the-art methods in MLC, as well as for evaluating our own approach with different learning-to-rank algorithms (MLC-RankSVM-IR, MLC-SVM-MAP, MLC-LambdaRank and MLC-ListNet). We used the *Vowel* dataset for an additional experiment, i.e., to assess the usefulness of our meta-level features in solving a multi-class classification problem.

### 3.2 MLC methods for comparison

We conduct controlled experiments to compare our approach with the following MLC methods which have performed strongly in benchmark evaluations:

1. Binary-SVM is a standard version of SVM for one-versus-rest classification, and a common baseline in comparative evaluation of classifiers (including MLC methods) (Elisseeff and Weston 2001; Joachims 1999).
2. Rank-SVM, the method proposed by Elisseeff and Weston (2001), is representative of the model-based methods which explicitly optimize ranked lists of categories for MLC.
3. IBLR, the instance-based method recently proposed by Cheng and Hüllermeier (2009). The method has two versions, one uses kNN-based features only (IBLR-ML) and another (IBLR-ML+) uses word-level features in conjunction with kNN- based features. We tested both versions and found that IBLR-ML performed consistently better than IBLR-ML+, which agrees with the conclusion by the authors of IBLR (Cheng and Hüllermeier 2009). We therefore use the results of IBLR-ML for method comparison in the rest of the paper. We used the Mulan (Tsoumakas et al. 2010) implementation provided by the authors.

[2] http://nyc.lti.cs.cmu.edu/clair/datasets.htm.

4. ML-kNN, the instance-based method proposed by Zhang and Zhou (2007) is another strong baseline (Cheng and Hüllermeier 2009). We used the publicly available Mulan implementation (Tsoumakas et al. 2010) of this method in our experiments.

All the systems produce scores for candidate categories given a test instance. Applying a threshold to those scores yields yes/no assignments of categories with respect to the instance. In Binary-SVM, we used the conventional threshold of zero for each category. In ML-kNN, IBLR-ML and Rank-SVM, we follow the same thresholding strategies as proposed by the authors of those methods. Specifically, for ML-kNN and IBLR we set the threshold to 0.5 since the score of each category is the system-estimated probability for category to be relevant to the given the test instance. For Rank-SVM we use the LLSF solution to predict a threshold for each test instance (Sect. 2.3) as proposed by Elisseeff and Weston (2001).

### 3.3 Evaluation metrics

We select five standard metrics for evaluating ranked lists, and three standard metrics for evaluating classification decisions.

- *Mean Average Precision* (*MAP*) (Voorhees 2003) is a traditional metric in IR evaluations for comparing ranked lists. It is defined as the average of the per-instance (or per-ranked-list) Average precision (AP) over all test instances. Let $D = \{x_i\}_{i=1,2,..n}$ be the test set of instances, $L_i$ be the ranked list of categories for a specific instance, $r_i(c)$ be the rank of category $c$ in list and $R_i$ be the set of categories relevant to instance. MAP is defined as:

$$MAP(D) = \frac{1}{n}\sum_{i=1}^{n} AP(x_i) \tag{3}$$

$$AP(x_i) = \frac{1}{|R_i|}\sum_{c\in R_i} \frac{|\{c' \in R_i \text{ s.t. } r_i(c') < r_i(c)\}|}{r_i(c)} \tag{4}$$

- *Ranking Loss* (*RankLoss*) is a popular metric for comparing MLC methods in ranking categories (Schapire and Singer 2000; Elisseeff and Weston 2001; Cheng and Hüllermeier 2009; Zhang and Zhou 2007). It measures the average number of times an irrelevant category is ranked above a relevant category in a ranked list:

$$Rankloss(D) = \frac{1}{n}\sum_{i=1}^{n} RL(x_i) \tag{5}$$

$$RL(x_i) = \frac{1}{|R_i||\bar{R}_i|}\left|\{(c, c') \in R_i \times \bar{R}_i \text{ s.t. } r_i(c) > r_i(c')\}\right| \tag{6}$$

- *Normalized Discounted Cumulative Gain* (NDCG) (Järvelin and Kekäläinen 2000) is a popular metric in recent IR evaluations. NDCG uses a logarithmic discounting factor to place a larger penalty when a relevant object is placed lower in the ranked list. For evaluating ranked lists of categories, NDCG is defined as:

$$NDCG(D) = \frac{1}{n}\sum_{i=1}^{n} \frac{DCG(x_i)}{IDCG(x_i)} \tag{7}$$

$$DCG(x_i) = \sum_{c\in R_i} \frac{1}{\log(1 + r_i(c))} \tag{8}$$

$$IDCG(x_i) = \sum_{i=1}^{|R_i|} \frac{1}{\log(1+i)} \tag{9}$$

- *Micro-averaged* $F_1$ (Micro-$F_1$) is a conventional metric for evaluating classifiers in category assignments to test instances (Lewis et al. 1996; Yang 1999; Yang and Pedersen 1997). The system-made decisions on test set $D$ with respect to a specific category $c \in \mathbf{C} \equiv \{c_1, \ldots, c_m\}$ can be divided into four groups: True Positives ($TP_c$), False Positives ($FP_c$), True Negatives ($TN_c$) and False Negatives ($FN_c$), respectively. The corresponding evaluation metrics are defined as:

$$\text{Global Precision } P = \frac{\sum_{c\in\mathbf{C}} TP_c}{\sum_{c\in\mathbf{C}}(TP_c + FP_c)} \tag{10}$$

$$\text{Global Recall } R = \frac{\sum_{c\in\mathbf{C}} TP_c}{\sum_{c\in\mathbf{C}}(TP_c + FN_c)} \tag{11}$$

$$\text{Micro-averaged } F_1 = \frac{2PR}{P+R} \tag{12}$$

- *Macro-averaged* $F_1$ (Macro-$F_1$) is also a conventional metric for evaluating classifiers in category assignments, defined as:

$$\text{Category-specific Precision } P_c = \frac{TP_c}{TP_c + FP_c} \tag{13}$$

$$\text{Category-specific Recall } R_c = \frac{TP_c}{TP_c + FN_c} \tag{14}$$

$$\text{Macro-averaged } F_1 = \frac{1}{m}\sum_{c\in\mathbf{C}} \frac{2P_cR_c}{P_c + R_c} \tag{15}$$

  Both micro-averaged and macro-averaged are informative for method comparison. The former gives the performance on each instance an equal weight in computing the average; the latter gives the performance on each category an equal weight in computing the average.
- *Hamming loss* (*HLoss*) (Schapire and Singer 1999) is a generalization of error-rate to the case of multilabel classification:

$$HLoss(D) = \frac{\sum_{c\in\mathbf{C}}(FP_c + FN_c)}{n \times m} \tag{16}$$

- *One-error* measures the average number of times the top most label in the ranked list is irrelevant. The IR equivalent of One-error is $1 - prec@1$. Let $c_i^{(1)} \in \mathbf{C}$ denotes the topmost category in the ranked list for $x_i$, we have:

$$OneError(D) = \frac{1}{n}\sum_{i=1}^{n} I\left(c_i^{(1)} \notin R_i\right) \tag{17}$$

- *Coverage* measures the average depth that one needs to go down the ranked list for a test instance in order to retrieve all the relevant category of that instance. In IR terms, it

**Table 2** CPU seconds for computing the nearest neighbors for all instances in a batch mode on each dataset

| | Emotions | Yeast | Scene | Citeseer | Reuters-21578 |
|---|---|---|---|---|---|
| Time Taken (secs) | 0.17 | 3.78 | 11.4 | 249 | 357 |

measures the average rank at 100% recall. It is defined as:

$$Coverage(D) = \frac{1}{n}\sum_{i=1}^{n} \max_{c \in R_i} r_i(c) - 1 \tag{18}$$

We choose the above metrics to evaluate the performance of both the ranking algorithms as well as the classification results after thresholding. *MAP*, *RankLoss*, *NDCG*, One-error and Coverage are metrics for evaluating ranked lists, while Micro-$F_1$, Macro-$F_1$ and *HLoss* are metrics for evaluating classification results. Among the eight metrics, *MAP*, *NDCG*, Micro-$F_1$ and Macro-$F_1$ are more commonly used in benchmark evaluations than the rest. We include the less common ones because they have been used in some of the recent MLC works (Elisseeff and Weston 2001; Zhang and Zhou 2007; Cheng and Hüllermeier 2009).

### 3.4 Experimental setting details

For term weighting in Citeseer and Reuters documents, we use the conventional TF-IDF scheme (namely 'ltc'). On the Emotions dataset, each feature was rescaled to a unit variance representation since we observed that the original values of the features are not comparably scaled. We did not use feature selection on any of the datasets for any method. All parameter tuning is done through a five-fold cross validation on the training set for each corpus. The tuned parameters include the number of nearest neighbors for the induction of meta-level features, ML-kNN and IBLR-ML, and the regularization parameter in Ranking-MLC (if any), Binary-SVM, Rank-SVM, as well as the learning rate in Ranking-MLC (if any). We set the number of epochs for stochastic gradient based methods to 50. For the number of nearest neighbors we tried values from 10 to 100 with the increments of 10; for the regularization parameter we tried 20 different values between $10^{-6}$ to $10^{6}$. We increased the range if cross-validation chose values on the boundaries.

Since our framework relies on the computation of meta-level features for each test instance, the scalability of our approach depends on how fast the nearest neighbors can be calculated. In our experiments, for computing the $L_1$, $L_2$ distances we used the publicly available *ANN* (Approximate Nearest Neighbor) library, which has the option to compute approximate as well as exact nearest neighbors efficiently[3]; we used the 'exact' option. Table 2 shows the CPU seconds for computing the nearest neighbors for all instances in a batch mode on the benchmark datasets. Run times could be considerably reduced using the approximate option. There is a rich body of work in scalable search of nearest neighbors (Kleinberg 1997; Arya et al. 1998; Roussopoulos et al. 1995; Yianilos 1993).

For the learning-to-rank methods based on our MLC framework and Rank-SVM, we use the same training set to induce the classification models and to learn the regression function for instance-based thresholding (*non-splitting* strategy), rather than having a primary training set to learn the classification model and a secondary training set to train the threshold

[3] http://www.cs.umd.edu/~mount/ANN.

regression function (*splitting* strategy). The reason for this strategy is to avoid data sparsity issues. Many categories in benchmark datasets have a relatively small number of positive training instances. For example, the 'Vowel' training set has 48 positive instances per category on average; the 'Emotions' training set has 67 positive instances per category on average. The Reuters collection has a highly skewed category distribution, meaning that only a few categories have a significant number of positive training instances but the majority of categories do not. Thus splitting the training data into a primary training set, a secondary training set and then a tertiary set as the hold-out validation set would make the data sparsity issue more severe. We tested both strategies with 5-fold cross validation on our training sets (without using test data), and found the non-splitting strategy to exhibit better results in both category ranking (measured in *MAP*) and in category assignments (measured in $F_1$).

## 4 Results

We obtain four sets of results from our controlled experiments:

- The first set focuses on the comparison of our approach (using MLC-ListNet as a specific choice of the learning-to-rank algorithm) with other state-of-the-art MLC methods, including Binary-SVM, RankSVM, IBLR and ML-kNN. We use 5 datasets and 8 metrics for both rank-based and classification-based evaluations (Table 3).
- The second set focuses on the comparison of the above MLC methods under the condition that all the methods use the same thresholding strategy, i.e., instance-based regression for thresholding. We use 5 datasets and 3 metrics for classification-based evaluations (Tables 4 and 5).
- The third set focuses on the comparison of different instantiations of our framework, including MLC-SVM-MAP, MLC-RankSVM-IR, MCL-LambdaRank and MLC-ListNet. We use 5 datasets and 8 metrics for both rank-based and classification-based evaluations (Table 6).
- The fourth set examines the usefulness of our meta-level features compared to using the original features. We use the average performance of 2 metrics (Micro-$F_1$ and *MAP*) across 5 datasets for evaluations. (Fig. 2).
- The fifth set analyzes the usefulness of our meta-level features in solving a multi-class classification problem, which is related to multi-label classification. We test binary SVM on the *vowel* dataset, using conventional features and our meta-level features as alternatives; we also compare the results with 10 methods which were previously evaluated on that dataset (Table 6).

We discuss each of the result sets in detail in the next sections.

### 4.1 Relative performance of the MLC methods

Table 3 summarizes the main results that compare our approach (MLC-ListNet) with other state-of-the-art methods in MLC, on the five datasets (*scene, emotions, yeast, citeseer* and *Reuters21578*) with respect to the eightmetrics (*Micro-F1, Macro-F1, MAP, RankLoss, NDCG, One-Error, Coverage* and *HLoss*). We rank the methods for each dataset and metric, and we total the ranks at the bottom or the table. The rank total of our method, MLC-ListNet, is clearly superior to the other methods (58 vs 127 or higher). MLC-ListNet is the best in 32 out of the total 40 lines in the table, while Binary-SVM is the best on 6 lines, and IBLR and RankSVM are only best in one line each. Comparing Rank-SVM with binary-SVM, both

**Table 3** Comparison of MLC methods on five datasets using eight metrics. The bold-faced numbers indicate the best system on a particular dataset given the metric; the numbers in parentheses are the ranks of the systems accordingly. The *Rank Total* is the sum of all ranks for each method across data sets and metrics. A $^{*}$indicates a p-value of 5% or lower and $^{**}$indicates a p-value of 1% or lower in the statistical significance test for performance comparison against MLC-ListNet

| | MLC-ListNet | Binary-SVM | RankSVM | IBLR | ML-kNN |
|---|---|---|---|---|---|
| Micro-$F_1$ | | | | | |
| *Scene* | **0.74210** (1) | 0.66502$^{**}$ (4) | 0.64072$^{**}$ (5) | 0.71138$^{**}$ (2) | 0.69864$^{**}$ (3) |
| *Emotions* | **0.72960** (1) | 0.64591$^{**}$ (3) | 0.62162$^{**}$ (5) | 0.69262$^{**}$ (2) | 0.63900$^{**}$ (4) |
| *Yeast* | **0.67633** (1) | 0.63132$^{**}$ (5) | 0.65620$^{**}$ (2) | 0.63714$^{**}$ (3) | 0.63681$^{**}$ (4) |
| *Citeseer* | **0.61711** (1) | 0.53448$^{**}$ (3) | 0.56677$^{**}$ (2) | 0.46148$^{**}$ (5) | 0.52678$^{**}$ (4) |
| *Reuters-21578* | 0.81658 (2) | **0.87084** (1) | 0.80416$^{*}$ (4) | 0.72808$^{**}$ (5) | 0.80484 (3) |
| Macro-$F_1$ | | | | | |
| *Scene* | **0.75746** (1) | 0.66975$^{*}$ (5) | 0.67308$^{*}$ (4) | 0.71452 (2) | 0.69162$^{*}$ (3) |
| *Emotions* | **0.72188** (1) | 0.66000$^{*}$ (3) | 0.61644$^{*}$ (4) | 0.68103$^{*}$ (2) | 0.61392$^{*}$ (5) |
| *Yeast* | **0.46425** (1) | 0.32405$^{**}$ (5) | 0.36645$^{**}$ (3) | 0.37093$^{**}$ (2) | 0.33614$^{**}$ (4) |
| *Citeseer* | **0.61326** (1) | 0.53523$^{**}$ (3) | 0.56477$^{**}$ (2) | 0.44948$^{**}$ (5) | 0.50330$^{**}$ (4) |
| *Reuters-21578* | **0.56074** (1) | 0.52763$^{**}$ (2) | 0.40487$^{**}$ (3) | 0.31100$^{**}$ (5) | 0.37045$^{**}$ (4) |
| MAP | | | | | |
| *Scene* | **0.87586** (1) | 0.85679$^{**}$ (4) | 0.85962$^{**}$ (2) | 0.85803$^{**}$ (3) | 0.85118$^{**}$ (5) |
| *Emotions* | **0.82357** (1) | 0.76850$^{**}$ (5) | 0.80223$^{*}$ (3) | 0.81478 (2) | 0.78969$^{**}$ (4) |
| *Yeast* | **0.76654** (1) | 0.74659$^{**}$ (5) | 0.75678$^{**}$ (4) | 0.75999$^{**}$ (2) | 0.75846$^{**}$ (3) |
| *Citeseer* | **0.76948** (1) | 0.73615$^{**}$ (3) | 0.75086$^{**}$ (2) | 0.69643$^{**}$ (5) | 0.73295$^{**}$ (4) |
| *Reuters-21578* | 0.92375 (4) | **0.95432** (1) | 0.93332 (2) | 0.85415 (5)$^{**}$ | 0.92492 (3) |
| RankLoss | | | | | |
| *Scene* | **0.06885** (1) | 0.08344$^{**}$ (4) | 0.07680$^{*}$ (2) | 0.08263$^{**}$ (3) | 0.09308$^{**}$ (5) |
| *Emotions* | **0.14002** (1) | 0.19391$^{**}$ (5) | 0.17716$^{**}$ (4) | 0.15989$^{*}$ (2) | 0.17137$^{**}$ (3) |
| *Yeast* | **0.16187** (1) | 0.19894$^{**}$ (5) | 0.17238$^{**}$ (3) | 0.16816$^{*}$ (2) | 0.17492$^{**}$ (4) |
| *Citeseer* | **0.07008** (1) | 0.10429$^{**}$ (4) | 0.07835$^{**}$ (2) | 0.10622$^{**}$ (5) | 0.08411$^{**}$ (3) |
| *Reuters-21578* | **0.00502** (1) | 0.00517$^{**}$ (2) | 0.00669$^{**}$ (4) | 0.02683$^{**}$ (5) | 0.00626$^{**}$ (3) |
| NDCG | | | | | |
| *Scene* | **0.90871** (1) | 0.87444$^{**}$ (4) | 0.89711$^{*}$ (4) | 0.89534$^{**}$ (2) | 0.89022$^{**}$ (5) |
| *Emotions* | **0.88139** (1) | 0.84263$^{**}$ (5) | 0.85147$^{**}$ (4) | 0.85827$^{**}$ (2) | 0.85694$^{**}$ (3) |
| *Yeast* | **0.85786** (1) | 0.84887$^{**}$ (3) | 0.85512$^{**}$ (1) | 0.85734 (2) | 0.85050$^{**}$ (5) |
| *Citeseer* | **0.83128** (1) | 0.80530$^{**}$ (2) | 0.81693$^{**}$ (3) | 0.77507$^{**}$ (5) | 0.80986$^{**}$ (4) |
| *Reuters-21578* | 0.94587 (4) | **0.96849** (1) | 0.95235 (2) | 0.89371$^{**}$ (5) | 0.94697 (3) |
| One-Error | | | | | |
| *Scene* | **0.20903** (1) | 0.23829$^{**}$ (4) | 0.23829$^{**}$ (4) | 0.23746$^{**}$ (2) | 0.24248$^{**}$ (5) |
| *Emotions* | **0.24753** (1) | 0.32673$^{**}$ (5) | 0.31188$^{**}$ (4) | 0.25743$^{*}$ (2) | 0.30693$^{**}$ (3) |
| *Yeast* | 0.24100 (4) | 0.23991 (3) | **0.23228** (1) | 0.23337 (2) | 0.24428$^{**}$ (5) |
| *Citeseer* | **0.32579** (1) | 0.34465$^{**}$ (2) | 0.35294$^{**}$ (3) | 0.41327$^{**}$ (5) | 0.36576$^{**}$ (4) |
| *Reuters-21578* | 0.11560 (4) | **0.06062** (1) | 0.09374 (2) | 0.18119$^{**}$ (5) | 0.10401 (3) |

**Table 3** (*Continued*)

| | MLC-ListNet | Binary-SVM | RankSVM | IBLR | ML-kNN |
|---|---|---|---|---|---|
| Coverage | | | | | |
| *Scene* | **0.44147** (1) | 0.52258$^{**}$ (4) | 0.48579$^{*}$ (2) | 0.51421$^{**}$ (3) | 0.56856$^{**}$ (5) |
| *Emotions* | **1.74257** (1) | 2.02970$^{**}$ (5) | 1.87129$^{*}$ (3) | 1.81683$^{*}$ (2) | 1.91584$^{**}$ (4) |
| *Yeast* | **6.14395** (1) | 7.12323$^{**}$ (5) | 6.42857$^{**}$ (4) | 6.35006$^{**}$ (2) | 6.41440$^{**}$ (3) |
| *Citeseer* | **1.50151** (1) | 2.15762$^{**}$ (5) | 1.65008$^{**}$ (2) | 2.10860$^{**}$ (4) | 1.77677$^{**}$ (3) |
| *Reuters-21578* | **0.89069** (1) | 0.99139$^{**}$ (2) | 1.04439$^{**}$ (3) | 3.8$^{**}$ (5) | 1.12918$^{**}$ (4) |
| HLoss | | | | | |
| *Scene* | 0.09783 (2) | 0.11357$^{**}$ (4) | 0.13963$^{‡}$ (5) | **0.09337** (1) | 0.09894 (3) |
| *Emotions* | **0.19142** (1) | 0.20957$^{*}$ (3) | 0.23102$^{*}$ (5) | 0.20957$^{*}$ (3) | 0.21535$^{**}$ (4) |
| *Yeast* | **0.19676** (1) | 0.19879 (4) | 0.20463$^{*}$ (5) | 0.19808 (3) | 0.19801 (2) |
| *Citeseer* | 0.05439 (3) | **0.05221** (1) | 0.05536$^{†}$ (4) | 0.05749$^{*}$ (5) | 0.05292 (2) |
| *Reuters-21578* | 0.00485 (3) | **0.00334** (1) | 0.00520$^{*}$ (4) | 0.00690$^{**}$ (5) | 0.00462 (2) |
| Rank Total | 58 | 136 | 127 | 127 | 147 |

are large-margin methods but the former outperforms the latter on 21 out of the 38 lines (the remaining two performed equally). These results are consistent with the previously reported evaluation on the Yeast dataset (Elisseeff and Weston 2001), showing some success of Rank-SVM by reinforcing partial-order preferences among categories. Comparing Rank-SVM with MLC-ListNet, on the other hand, the latter outperforms the former in 37 out of the 40 lines showing the advantage of using of the meta-level features in the learning-to rank framework. Comparing MLC-ListNet, ML-kNN and IBLR-ML, these methods have one property in common: they are either fully instance-based or partially instance-based leveraging kNN-based features. IBLR-ML outperforms ML-kNN in 23 out of the 40 cases; this is more or less consistent with the previous report by Cheng and Hüllermeier (2009) in terms of the relative performance of the two methods. Nevertheless, both IBLR-ML and ML-kNN underperform MLC-ListNet in 38 and 35 cases.

### 4.2 Effectiveness of thresholding strategies in the MLC methods

Each of the MLC methods we compare in Table 3 has its own thresholding strategy, and the different strategies could be partly responsible for the difference in performance between these methods. In order to measure the effects of the thresholding strategies, we evaluate the performance of each method under two conditions: using its default thresholding strategy, or using the instance-based regression method. Table 4 shows the results on 5 datasets; since thresholding strategies do not affect ranked lists but influence classification decisions, we only need the classification-based metrics (Micro-$F_1$, Macro-$F_1$ and *HLoss*) for the evaluation. We can clearly see in the results that by using instance-based regression for thresholding, the performance of all the methods improved substantially in both Micro-$F_1$ and Macro-$F_1$ across almost all datasets. On average, there is 6.15% improvement in Micro-$F_1$ and a 9.78% improvement in Macro-$F_1$ respectively. However, the *HLoss* scores present an inconclusive picture. Further investigation is required regarding the nature of the evaluation metric and its consistence with other well-reserved metrics. The results shown in Table 5 compare the performance of all the five MLC methods where the same instance-based regression is used for thresholding in each method. Although the results of Binary-SVM,

**Table 4** Results of using instance-based regression to threshold in Binary-SVM, RankSVM, IBLR and ML-kNN. In the two columns under each method, one shows the result of using the method's default thresholding strategy and the other shows the results of using the instance-based regression

| Threshold | Binary-SVM | | RankSVM | |
|---|---|---|---|---|
| | Default | Regression | Default | Regression |
| Micro-$F_1$ | | | | |
| *Scene* | 0.66502 | **0.71575** | 0.64072 | **0.73531** |
| *Emotions* | 0.64591 | **0.67250** | 0.62162 | **0.67526** |
| *Yeast* | 0.63132 | **0.64323** | 0.65620 | **0.65696** |
| *Citeseer* | 0.53448 | **0.59922** | 0.56677 | **0.59866** |
| *Reuters-21578* | **0.87084** | 0.86494 | 0.80416 | **0.84812** |
| Macro-$F_1$ | | | | |
| *Scene* | 0.66975 | **0.72482** | 0.67308 | **0.74592** |
| *Emotions* | 0.66 | **0.67216** | 0.61644 | **0.67522** |
| *Yeast* | 0.32405 | **0.35742** | 0.36645 | **0.43306** |
| *Citeseer* | 0.53523 | **0.59700** | 0.56477 | **0.60018** |
| *Reuters-21578* | 0.52763 | **0.543498** | 0.40487 | **0.46308** |
| HLoss | | | | |
| *Scene* | 0.11357 | **0.10438** | 0.13963 | **0.10103** |
| *Emotions* | **0.20957** | 0.21782 | 0.23102 | **0.22855** |
| *Yeast* | **0.19879** | 0.20003 | **0.20463** | 0.20556 |
| *Citeseer* | **0.05221** | 0.05962 | **0.05536** | 0.05709 |
| *Reuters-21578* | **0.00334** | 0.00353 | 0.00520 | **0.00402** |
| Threshold | ML-kNN | | IBLR-ML | |
| | Default | Regression | Default | Regression |
| Micro-$F_1$ | | | | |
| *Scene* | 0.69864 | **0.72973** | 0.71138 | **0.72860** |
| *Emotions* | 0.63900 | **0.70709** | 0.69262 | **0.69977** |
| *Yeast* | 0.63681 | **0.65124** | 0.63714 | **0.65512** |
| *Citeseer* | 0.52678 | **0.58627** | 0.46148 | **0.54350** |
| *Reuters-21578* | 0.80484 | **0.81287** | 0.72808 | **0.74788** |
| Macro-$F_1$ | | | | |
| *Scene* | 0.69162 | **0.73855** | 0.71452 | **0.73868** |
| *Emotions* | 0.61392 | **0.69597** | 0.68103 | **0.69068** |
| *Yeast* | 0.33614 | **0.38223** | 0.37093 | **0.41136** |
| *Citeseer* | 0.50330 | **0.58194** | 0.44948 | **0.54518** |
| *Reuters-21578* | 0.37045 | **0.39759** | 0.31100 | **0.33388** |
| HLoss | | | | |
| *Scene* | **0.09894** | 0.10675 | **0.09337** | 0.10382 |
| *Emotions* | 0.21535 | **0.21122** | 0.20957 | **0.20875** |
| *Yeast* | **0.19801** | 0.20447 | **0.19808** | 0.20159 |
| *Citeseer* | **0.05292** | 0.06149 | **0.05749** | 0.06796 |
| *Reuters-21578* | 0.00462 | **0.00460** | 0.00690 | **0.00675** |

**Table 5** Comparison of different MLC methods with the same thresholding strategy. A † beside the name indicates that the instance-based regression thresholding has been applied instead of the method's default thresholding strategy

| | MLC-ListNet | Binary-SVM† | RankSVM† | IBLR† | ML-kNN† |
|---|---|---|---|---|---|
| Micro-$F_1$ | | | | | |
| *Scene* | **0.74210** | 0.71575$^{**}$ | 0.73531$^{*}$ | 0.72860$^{**}$ | 0.72973$^{**}$ |
| *Emotions* | **0.72960** | 0.67250$^{**}$ | 0.67526$^{**}$ | 0.69977$^{**}$ | 0.70709$^{**}$ |
| *Yeast* | **0.67633** | 0.64323$^{**}$ | 0.65696$^{**}$ | 0.65512$^{**}$ | 0.65124$^{**}$ |
| *Citeseer* | **0.61711** | 0.59922$^{*}$ | 0.59866$^{*}$ | 0.54350$^{**}$ | 0.58627$^{**}$ |
| *Reuters-21578* | 0.81658 | **0.86494** | 0.84812 | 0.74788$^{**}$ | 0.81287 |
| Macro-$F_1$ | | | | | |
| *Scene* | **0.75746** | 0.72482$^{*}$ | 0.74592 | 0.73868 | 0.73855 |
| *Emotions* | **0.72188** | 0.67216$^{*}$ | 0.67522$^{*}$ | 0.69068$^{*}$ | 0.69597$^{*}$ |
| *Yeast* | **0.46425** | 0.35742$^{**}$ | 0.43306$^{**}$ | 0.41136$^{**}$ | 0.38223$^{**}$ |
| *Citeseer* | **0.61326** | 0.59700$^{**}$ | 0.60018$^{*}$ | 0.54518$^{**}$ | 0.58194$^{**}$ |
| *Reuters-21578* | **0.56074** | 0.54350$^{*}$ | 0.46308$^{**}$ | 0.33388$^{**}$ | 0.39759$^{**}$ |
| HLoss | | | | | |
| *Scene* | **0.09783** | 0.10438 | 0.10103 | 0.10382 | 0.10675 |
| *Emotions* | **0.19142** | 0.21782$^{*}$ | 0.22855$^{**}$ | 0.20875$^{**}$ | 0.21122$^{**}$ |
| *Yeast* | **0.19676** | 0.20003 | 0.20556$^{**}$ | 0.20159$^{**}$ | 0.20447$^{**}$ |
| *Citeseer* | **0.05439** | 0.05962$^{**}$ | 0.05709$^{**}$ | 0.06796$^{**}$ | 0.06149$^{**}$ |
| *Reuters-21578* | 0.00485 | **0.00353** | 0.00402 | 0.00675$^{**}$ | 0.00460 |
| Rank Total | 20 | 51 | 43 | 57 | 56 |

RankSVM, IBLR and ML-kNN have improved, MLC-ListNet remains the best performer on average. MLC-ListNet has the best performance in 10 out of the total 12 cases while Binary-SVM is the best in the remaining 2 cases. These results suggest that the superior performance of MLC-ListNet comes from the use of meta-level features and the effective learning-to-rank algorithm as the main factors.

### 4.3 Statistical significance tests

We perform significance tests on all the results in Table 3 and Table 5. We use the sign-test for Micro-$F_1$, and the Wilcoxon signed rank test for the rest of the evaluation metrics. The sign-test is suitable for comparing binary predictions of two systems on all category-document pairs (Liu 1999). Each prediction is considered as a random event. The null hypothesis is that both the systems are equally good; the alternative is that one of the systems is better. The Wilcoxon signed-rank test is a non-parametric statistical test for pairwise comparison of methods and a better alternative to the paired t-test when the performance scores are not normally distributed. For Macro-$F_1$, the $F_1$ scores on each category are used to compare two systems. In the case of *MAP*, *RankLoss*, *NDCG*, *OneError*, *Coverage* and *HLoss*, the performance scores on each test instance are used to compare systems. In both Table 3 and Table 5, we report the significance test results of comparing MLC-ListNet with every other method. A $^{*}$ indicates a p-value of 5% or lower, and a $^{**}$ indicates a p-value

**Table 6** Performance of different learning-to-rank methods in MLC. The 8 different blocks report the performance on each of the 8 evaluation metrics. The best method for each dataset and metric is highlighted in bold. The relative ranks between the methods are shown in parentheses

| | MLC-SVM-MAP | MLC-RankSVM-IR | MLC-LambdaRank | MLC-ListNet |
|---|---|---|---|---|
| Micro-$F_1$ | | | | |
| *Scene* | 0.73995 (3) | **0.74231** (1) | 0.73480 (4) | 0.74210 (2) |
| *Emotions* | 0.72895 (3) | **0.72960** (1) | 0.72875 (4) | **0.72960** (1) |
| *Yeast* | 0.67339 (4) | 0.67648 (2) | 0.67483 (3) | **0.67651** (1) |
| *Citeseer* | 0.60440 (4) | 0.60837 (2) | 0.60810 (3) | **0.61711** (1) |
| *Reuters-21578* | 0.82726 (2) | **0.83022** (1) | 0.81974 (3) | 0.81658 (4) |
| Macro-$F_1$ | | | | |
| *Scene* | 0.75697 (3) | **0.75838** (1) | 0.75233 (4) | 0.75746 (2) |
| *Emotions* | 0.72291 (3) | 0.72468 (2) | **0.72521** (1) | 0.72188 (4) |
| *Yeast* | 0.45438 (3) | 0.45594 (2) | 0.45074 (4) | **0.46425** (1) |
| *Citeseer* | 0.60039 (4) | 0.60354 (3) | 0.60699 (2) | **0.61326** (1) |
| *Reuters-21578* | 0.55967 (3) | **0.56229** (1) | 0.55509 (4) | 0.56074 (2) |
| MAP | | | | |
| *Scene* | **0.87964** (1) | 0.87787 (2) | 0.86966 (4) | 0.87586 (3) |
| *Emotions* | 0.82336 (2) | 0.82027 (4) | 0.82269 (3) | **0.82357** (1) |
| *Yeast* | 0.76610 (3) | 0.76579 (4) | 0.76626 (2) | **0.76654** (1) |
| *Citeseer* | 0.76875 (2) | 0.76377 (4) | 0.76422 (3) | **0.76948** (1) |
| *Reuters-21578* | **0.94238** (1) | 0.94195 (2) | 0.94083 (3) | 0.92375 (4) |
| RankLoss | | | | |
| *Scene* | 0.07048 (3) | **0.06844** (1) | 0.07425 (4) | 0.06885 (2) |
| *Emotions* | 0.14205 (4) | **0.13830** (1) | 0.14081 (3) | 0.14002 (2) |
| *Yeast* | 0.16310 (3) | 0.16268 (2) | 0.16320 (4) | **0.16187** (1) |
| *Citeseer* | 0.07012 (3) | **0.06996** (1) | 0.07148 (4) | 0.07008 (2) |
| *Reuters-21578* | 0.00515 (4) | **0.00486** (1) | 0.00502 (2) | 0.00502 (3) |
| NDCG | | | | |
| *Scene* | 0.90826 (2) | 0.90565 (3) | 0.90416 (4) | **0.90871** (1) |
| *Emotions* | 0.87932 (3) | 0.87329 (4) | 0.88130 (2) | **0.88139** (1) |
| *Yeast* | 0.85783 (2) | 0.85751 (4) | 0.85782 (3) | **0.85786** (1) |
| *Citeseer* | 0.82602 (3) | 0.82067 (4) | 0.82762 (2) | **0.83128** (1) |
| *Reuters-21578* | 0.95803 (2) | 0.95798 (3) | **0.95829** (1) | 0.94587 (4) |
| One-Error | | | | |
| *Scene* | 0.22993 (4) | **0.20903** (1) | 0.21488 (3) | **0.20903** (1) |
| *Emotions* | 0.24753 (1) | 0.24753 (1) | 0.24753 (1) | 0.24753 (1) |
| *Yeast* | **0.23991** (1) | 0.24100 (4) | **0.23991** (1) | 0.24100 (4) |
| *Citeseer* | **0.32504** (1) | 0.33333 (3) | 0.33560 (4) | 0.32579 (2) |
| *Reuters-21578* | **0.07983** (2) | **0.07983** (2) | 0.08281 (3) | 0.11560 (4) |

**Table 6** (*Continued*)

| | MLC-SVM-MAP | MLC-RankSVM-IR | MLC-LambdaRank | MLC-ListNet |
|---|---|---|---|---|
| Coverage | | | | |
| *Scene* | 0.45234 (3) | 0.45067 (2) | 0.47074 (4) | **0.44147** (1) |
| *Emotions* | 1.76733 (4) | 1.76733 (4) | 1.75248 (2) | **1.74257** (1) |
| *Yeast* | 6.19302 (4) | 6.16685 (3) | 6.15703 (2) | **6.14395** (1) |
| *Citeseer* | **1.49472** (1) | 1.51584 (3) | 1.51961 (4) | 1.50151 (2) |
| *Reuters-21578* | 1.01358 (4) | 0.99508 (3) | 0.91255 (2) | **0.89069** (1) |
| HLoss | | | | |
| *Scene* | 0.09601 (2) | **0.09532** (1) | 0.10089 (4) | 0.09783 (3) |
| *Emotions* | 0.18977 (3) | **0.18812** (2) | **0.18812**(2) | 0.19142 (4) |
| *Yeast* | 0.19808 (4) | 0.19731 (2) | 0.19785(3) | **0.19676** (1) |
| *Citeseer* | 0.05448 (3) | **0.05439** (2) | 0.05452 (4) | **0.05439** (2) |
| *Reuters-21578* | 0.00443 (1) | **0.00453** (2) | 0.00468 (3) | 0.00485 (4) |
| Rank Total | 108 | 91 | 118 | 79 |

of 1% or lower. It can be observed in these tables that MLC-ListNet significantly outperformed the other methods at the 5% level or lower in most cases and at the 1% level or lower in a large number of the cases, which is indeed strong evidence for the effectiveness of our approach. We do not use ANOVA tests for multi-set comparison of systems because ANOVA assumes a normal distribution on the data, but the performance scores we have for system comparison do not necessarily satisfy such an assumption. The Friedman test has been recently advocated for comparing classifiers on multiple datasets (Demšar 2006; Garcıa and Herrera 2008); This method does not assume a normal distribution. However, it treats each dataset as a random event, and requires a relatively large number of datasets for meaningful testing. It is not recommended to use the Friedman test when the number of datasets is 10 or less.[4]

### 4.4 Performance of different learning-to-rank methods in MLC

Table 6 summarizes the performance of using different learning to rank methods—MLC-RankSVM-IR, MLC-SVM-MAP, MLC-LambdaRank and MLC-ListNet on the five datasets of *scene, emotions, yeast, citeseer, Reuters21578*. Overall, the variance in performance among the learning-to-rank methods are smaller than those we observed in Table 3, and the performance mean is higher. These indicate that all the learning-to-rank methods we tested are relatively effective in working with the meta-level features. In most cases no method totally outperforms the others. Relatively, MCL-ListNet has the best score in Rank Total, and is particularly strong in the *NDCG* metric, and MLC-RankSVM-IR has the second best score in Rank Total, and is particularly strong in the *RankLoss* metric. The suboptimal performance of LambdaRank could be partially explained by the difficulty of jointly tuning all the parameters in the method (such as the learning rate, and its reduction as a function of iterations). The difficulty is amplified because the system needs to be tuned with respect to a variety of metrics and on many datasets.

[4]This is according to personal communication with Demšar (2006).

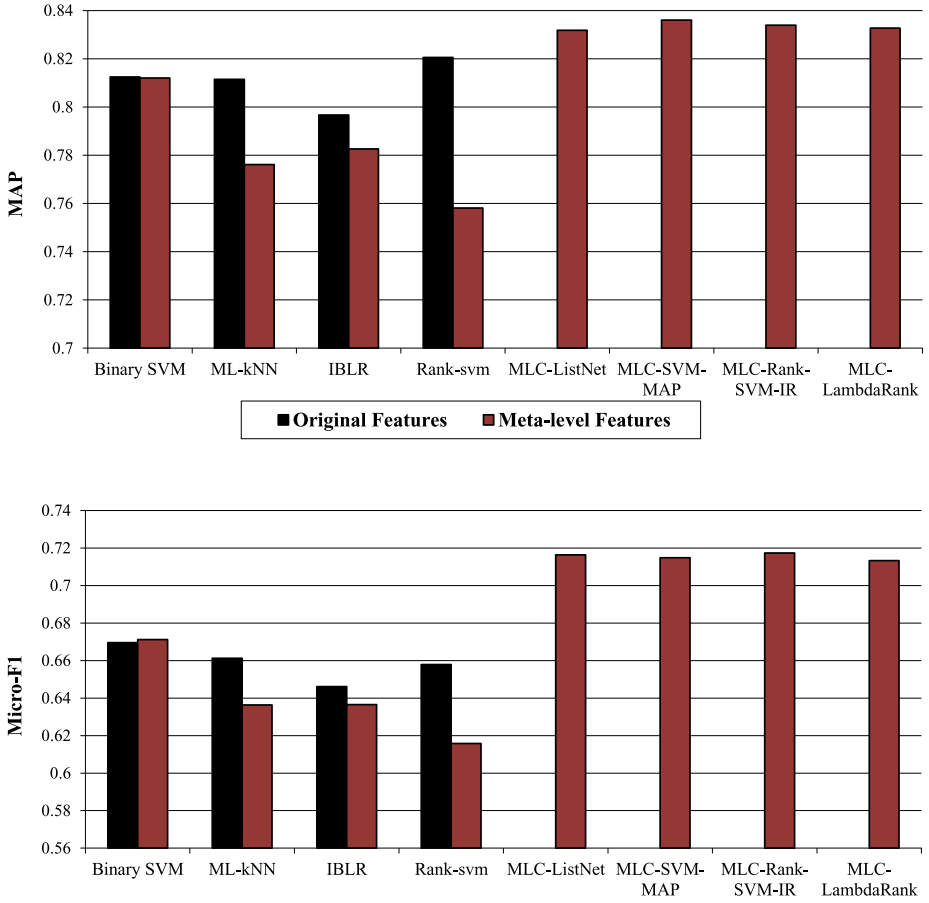


**Fig. 2** Each graph above compares the average performance of each of the methods with the original features and meta-level features. The *graph on top* compares the average Micro-$F_1$ and the *graph below* compares the average *MAP*. The performance of the learning to rank methods are also plotted

## 4.5 Effectiveness of meta-level features

Although the meta-level features are primarily designed to enable the use of learning-to-rank algorithms in solving the multi-label classification problem, it is also possible to use these meta-level features to produce an alternative representation of the input for conventional MLC methods, such as binary SVM, IBLR, ML-kNN and Rank-SVM. Instead of using the original features, such as words in text or pixels in images, one could use these meta-level features to represent each input instance in binary SVM, IBLR, ML-kNN and Rank-SVM. We examine whether or not this alternative representation would lead to significant performance improvements for those methods. Figure 2 summarizes our experimental results: Each method is evaluated on all the five datasets in two conditions, i.e., using the original features vs. using the meta-level features. The graph on top compares the performance in category assignments (classification decisions) with Micro-$F_1$, averaged over the five datasets; while the graph below compares the performance in category ranking with *MAP*, again averaged over the five datasets. For reference we also include the performance

**Table 7** Performance results on the *vowel* dataset. The results for the other methods were taken from page 444, Hastie et al. (2009)

| | Method | Error-rate |
|---|---|---|
| (1) | LDA | .56 |
| (2) | QDA | .53 |
| (3) | CART | .56 |
| (4) | CART (linear combination splits) | .54 |
| (5) | Single-layer Perceptron | .67 |
| (6) | Multi-layer Perceptron (88 hidden units) | .49 |
| (7) | Gaussian Node Network (528 hidden units) | .45 |
| (8) | Nearest Neighbor | .44 |
| (9) | FDA/BRUTO | .44 |
| (10) | FDA/MARS (degree=2) | .42 |
| (11) | SVM with plain features | .612 |
| (12) | SVM with only meta-level features | .420 |

of the learning-to-rank methods in these figures; notice that only the meta-level features are applicable to these methods. These results show that for the conventional MLC methods, meta-level features either hurt the performance (as for ML-kNN and IBLR) or have only a negligible performance improvement (as for Binary SVM) on average. In any case, the use of meta-level features does not enable the conventional MLC methods to close the performance gap compared to the learning-to-rank methods. This suggests that in order to fully leverage the meta-level features, it is important for the methods to have the ability to enforce partial-order constraints among categories or to optimize category ranking explicitly, which the learning-to-rank methods do but the conventional classifiers (except RankSVM) do not.

### 4.6 Effectiveness of meta-level features in multi-class classification on vowel

Given the encouraging results of our approach with the meta-level features in MLC, we are naturally interested in testing the effectiveness of such features in solving a similar problem: multi-class classification (MCC). MCC is similar to MLC in the sense that both problems require the learning methods to focus on one-to-many mapping from each instance to categories, and our meta-level features are designed to make such learning easier. On the other hand, MCC differs form MLC in the sense that each instance has one and only one correct category. Nevertheless, the similarity between MCC and MLC makes it interesting to try our meta-level features instead of the conventional features in MCC. In our experiments we took a common and simple approach to MCC: we train a standard binary SVM per category, and assign the category with the top score to each test instance. We evaluated SVM with two options: using original features, or using the meta-level features which we designed for MLC. We choose to use the *vowel* dataset for the experiments, which is a common benchmark in MCC and known as a difficult classification task. Table 7 shows the results of SVM using the two types of features, respectively; previously published results on this dataset are also included for comparison. Using the meta-level features reduced the error rate of SVM from 0.612 to 0.420. Whereas SVM with the original features is the worst-performing method, SVM with our meta-level features ties for second-best method.

## 5 Conclusion and future work

In this paper we produced a new approach for learning to rank categories in multi-label classification. By introducing meta-level features that effectively characterize the one-to-many

relationships from instances to categories in MLC, and by formulating the category ranking problem as a standard *ad-hoc* retrieval problem, our framework enables the application of a broad range of learning-to-rank retrieval algorithms for MLC optimization with respect to various performance metrics. Our controlled experiments with this new approach on six benchmark datasets with eight performance metrics shows consistent and significant performance improvements of our method over the previous state-of-the-art methods (Rank-SVM, ML-kNN and IBLR-ML).

This study provides useful insights into how to enhance the performance of MLC methods by improving the representation schemes for instances, categories and their relationships, and by creatively leveraging state-of-the-art algorithms for learning to rank. A logical extension of our approach is to explore different techniques for automated induction of meta-level features, including unsupervised methods (such as SVD or LDA) or supervised methods (such as sLDA or DiscLDA) or using a hybrid between meta-level features, instance-only features and category specific features. Another interesting direction for future research would be to extend our framework by jointly optimizing the learning-to-rank part and the learning-to-threshold part using a combined objective function, instead of treating them as two separate optimization problems.

## References


Arya, S., Mount, D., Netanyahu, N., Silverman, R., & Wu, A. (1998). An optimal algorithm for approximate nearest neighbor searching fixed dimensions. *Journal of the ACM* , *45*(6), 891–923.

Boutell, M., Luo, J., Shen, X., & Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771.

Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., & Hullender, G. (2005). Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on machine learning* (p. 96). New York: ACM.

Burges, C., Ragno, R., & Le, Q. (2007). Learning to rank with nonsmooth cost functions. *Advances in Neural Information Processing Systems*, *19*, 193.

Cao, Z., Qin, T., Liu, T., Tsai, M., & Li, H. (2007). Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on machine learning* (p. 136). New York: ACM.

Cheng, W., & Hüllermeier, E. (2009). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2–3), 211–225. doi:10.1007/s10994-009-5127-5, http://www.springerlink.com/content/m20342966250233x/.

Creecy, R., Masand, B., Smith, S., & Waltz, D. (1992). Trading MIPS and memory for knowledge engineering. *Communications of the ACM*, *35*(8), 48–64.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 30.

Donmez, P., Svore, K., & Burges, C. (2009). On the local optimality of LambdaRank. In *Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval* (pp. 460–467). New York: ACM.

Elisseeff, A., & Weston, J. (2001). Kernel methods for multi-labelled classification and categorical regression problems. In *Advances in neural information processing systems* (Vol. 14, pp. 681–687). Cambridge: MIT Press.

Freund, Y., Iyer, R., Schapire, R., & Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, *4*, 933–969.

Ganapathiraju, A., Hamaker, J., & Picone, J. (1998). Support vector machines for speech recognition. In *International conference on spoken language processing* (pp. 2923–2926). New York: ACM.

Garcıa, S., & Herrera, F. (2008). An extension on statistical comparisons of classifiers over multiple data sets for all pairwise comparisons. *Journal of Machine Learning Research*, *9*, 2677–2694.

Gopal, S., & Yang, Y. (2010). Multilabel classification with meta-level features. In *Proceeding of the 33rd international ACM SIGIR conference on research and development in information retrieval* (pp. 315–322). New York: ACM.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*.

Järvelin, K., & Kekäläinen, J. (2000). IR evaluation methods for retrieving highly relevant documents. In *Proceedings of the 23rd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 41–48). New York: ACM.

Joachims, T. (1999). Making large-scale support vector machine learning practical. In *Advances in kernel methods: support vector learning*.

Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 133–142). New York: ACM.

Kleinberg, J. (1997). Two algorithms for nearest-neighbor search in high dimensions. In *Proceedings of the twenty-ninth annual ACM symposium on theory of computing* (pp. 599–608). New York: ACM.

Lewis, D., Schapire, R., Callan, J., & Papka, R. (1996). Training algorithms for linear text classifiers. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 298–306). New York: ACM.

Li, P., Burges, C., Wu, Q., Platt, J., Koller, D., Singer, Y., & Roweis, S. (2007) McRank: Learning to rank using multiple classification and gradient boosting. *Advances in Neural Information Processing Systems*.

Qin, T., Liu, T., Xu, J., & Li, H. (2010) LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval*, 1–29.

Roussopoulos, N., Kelley, S., & Vincent, F. (1995). Nearest neighbor queries. In *ACM sigmod record* (Vol. 24, pp. 71–79). New York: ACM.

Schapire, R., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.

Schapire, R., & Singer, Y. (2000). BoosTexter: A boosting-based system for text categorization. *Machine Learning*, *39*(2), 135–168.

Trohidis, K., Tsoumakas, G., Kalliris, G., & Vlahavas, I. (2008). Multilabel classification of music into emotions. In *Proc. 9th international conference on music information retrieval (ISMIR 2008)*, Philadelphia, PA, USA (Vol. 2008).

Tsai, M., Liu, T., Qin, T., Chen, H., & Ma, W. (2007). FRank: A ranking method with fidelity loss. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (p. 390). New York: ACM.

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2006). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, *6*(2), 1453.

Tsoumakas, G., Vilcek, J., Spyromitros, E., & Vlahavas, I. (2010). Mulan: a Java library for multilabel learning. *Journal of Machine Learning Research*, *1*, 1–48.

Vapnik, V. (2000). *The nature of statistical learning theory*. Berlin: Springer.

Voorhees, E. (2003) Overview of TREC 2002. *NIST special publication SP* (pp. 1–16).

Xu, J., & Li, H. (2007). Adarank: A boosting algorithm for information retrieval. In *Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval* (p. 398). New York: ACM.

Yang, Y. (1994). Expert network: Effective and efficient learning from human decisions in text categorization and retrieval. In *ACM SIGIR conference on research and development in information retrieval* (pp. 13–22). New York: Springer.

Yang, Y. (1999). An evaluation of statistical approaches to text categorization. *Information Retrieval*, *1*(1), 69–90.

Yang, Y. (2001). A study of thresholding strategies for text categorization. In *Proceedings of the 24th annual international ACM SIGIR conference on research and development in information retrieval* (pp. 137–145). New York: ACM.

Yang, Y., Liu, X. (1999). A re-examination of text categorization methods. In *Proceedings of the 22nd annual international ACM SIGIR conference on research and development in information retrieval* (pp. 42–49). New York: ACM.

Yang, Y., & Pedersen, J. (1997) A comparative study on feature selection in text categorization. In *International conference in machine learning* (pp. 412–420). Citeseer.

Yianilos, P. (1993). Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the fourth annual ACM-SIAM symposium on discrete algorithms* (pp. 311–321). Philadelphia: Society for Industrial and Applied Mathematics.

Yue, Y., & Finley, T. (2007). A support vector method for optimizing average precision. In *Proceedings of SIGIR07* (pp. 271–278). New York: ACM.

Zhang, M., & Zhou, Z. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048.