Multilabel Classification with Meta-level Features

Siddharth Gopal
Carnegie Mellon University
Pittsburgh PA 15213
sgopal1@andrew.cmu.edu

Yiming Yang
Carnegie Mellon University
Pittsburgh PA 15213
yiming@cs.cmu.edu

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 been focused on learning-to-map from instances to ranked lists of categories in a relatively high-dimensional space. The fine-grained features in such a space may not be sufficiently expressive for characterizing discriminative patterns, and worse, make the model complexity unnecessarily high. This paper proposes an alternative approach by transforming conventional representations of instances and categories into a relatively small set of link-based meta-level features, and leveraging successful learning-to-rank retrieval algorithms (e.g., SVM-MAP) over this reduced feature space. Controlled experiments on multiple benchmark datasets show strong empirical evidence for the strength of the proposed approach, as it significantly outperformed several state-of-the-art methods, including Rank-SVM, ML-kNN and IBLR-ML (Instance-based Logistic Regression for Multi-label Classification) in most cases.

Categories and Subject Descriptors
I.5.2 [**PATTERN RECOGNITION**] Design Methodology; Classifier design and evaluation; H.1.0 [**INFORMATION SYSTEMS**]: General;

General Terms
Algorithms, Design, Experimentation, Performance

Keywords
Multi-label classification; model design; learning to rank; comparative evaluation

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 both topics *politics* and *economics*, and also could be related to *China* and *USA* as the regional categories. An image (picture) 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 hardware failure, 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 two-way classifiers, such as binary Support Vector Machines (SVM), Naïve 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") [17] 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 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 in MLC.

Elisseeff and Weston [5] proposed a large-margin approach, 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 simultaneously, conditioned on *partial-order constraints*. That is, ranking the relevant categories of each training 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; thus, the number of parameters in Rank-SVM is the product of the number of categories and the size of the feature set. In other words, the model complexity (measured using the number of free parameters) is the same as that of *m* binary-SVMs where *m* is the number of target categories. Experiments by the authors of Rank-SVM show that this method significantly outperformed binary SVM in gene classification on a micro-array dataset (namely "the *yeast* dataset").

Zhang and Zhou [1] proposed an instance-based approach which is named as Multi-label k-Nearest Neighbor (ML-kNN).

SIGIR'10, July 19–23, 2010, Geneva, Switzerland.

Cheng and Hullermeier proposed another variant called Instance-Based Logistic Regression (IBLR) [3]. Multi-label versions of kNN have been studied in text categorization for a long time and commonly used as strong baselines in benchmark evaluations [2][20][22]. ML-kNN and IBLR are relatively new variants which are similar in the sense that both use Euclidean distance to identify the *k* nearest neighbors for each test instance, but differ 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 [14] and Adaboost.MH [15] , and the experiments by the authors of IBLR showed further performance improvement by IBLR over the results of ML-kNN on multiple datasets [3].

MLC methods, including the ones discussed above, have been focused on learning-to-rank in a relatively high-dimensional space. In Rank-SVM for text categorization the features are words in the training-set document vocabulary. In IBLR for text categorization the feature set is the union of word-level features and kNN-based features (used to model the interdependencies among categories). Both methods learn a linear function per category, so the total number of features is *md* where *d* is the feature-set size and *m* is the number of categories. In machine learning, it is generally understood that when the number of model parameters is unnecessarily large, the model tends to overfit training data and does not generalize well on test data. To what extent would this be an issue in current MLC methods? Further, can we find a better solution for MLC by transforming lower-level features to higher-level ones, and then learning to rank more effectively in the reduced space? Thorough investigation is needed to answer these questions, and such is the primary focus of this paper. Specifically, we address these questions by the following means:

1) We propose a generic framework that allows automated transformation of a conventional data representation (such as a bag of words per instance and a set of training instances per category) into meta-level features. This enables a broad range of learning-to-rank algorithms in information retrieval (IR) to be deployed for ranking categories in MLC.
2) We use SVM-MAP, a large-margin method for optimizing ranked lists of documents with respect to the Mean Average Precision in IR, as the choice of algorithm in this paper to illustrate effective MLC learning with meta-level features. For convenience we call this instantiation of our framework SVM-MAP-MLC in distinction from the use of SVM-MAP in ad-hoc information retrieval.
3) We conduct controlled experiments on multiple benchmark datasets to evaluate SVM-MAP-MLC in comparison with other state-of-the-art methods, including Rank-SVM, ML-kNN, IBLR and Binary SVM.
4) We provide strong empirical evidence for the strengths of the proposed method, with p-values at the 1% level or smaller on all the datasets in statistical significance tests for comparing our approach with the other state-of-the-art methods.

The rest of the paper is organized as follows. Section 2 introduces the meta-level features for MLC. Section 3 describes the method for category ranking optimization, i.e., SVM-MAP-MLC and the strategy for optimizing the threshold on each ranked list. Section 4 presents design of controlled experiments. Section 5 reports the main results. Section 6 summarizes our findings and conclusion.

2. META-LEVEL FEATURES FOR MLC

Following a standard notation in machine learning, we define *X* as the input space (of all possible instances), *Y* as the output space (of all possible ranked lists of target variables), and $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ as a training set of *n* pairs of $x_i \in X$ and $y_i \in Y$. The learning-to-rank problem is generally defined as to find the optimal mapping $f : X \to Y$ given *T*. Now we define a transformation from $x \in X$ to $z \in Z$ with the following properties:

1) The feature-set size of *Z* should be relative small.
2) The features in $z \in Z$ should be highly informative about how each instance is related to *multiple* categories, and should discriminate relevant instances from irrelevant instances with respect to any possible subset of categories.
3) The transformation from $x \in X$ to $z \in Z$ should be learnable based on a labeled training set and automatically computable for each test instance on the fly.
4) The transformed training data should allow a broad range of learning-to-rank algorithms to be used for MLC optimization with respect to various loss functions.

Based on these criteria we define vector z as the concatenation of the sub-vectors below, which are defined for each test instance x_0 and category $c_j \in C$ for $j = 1,2,\cdots,m$ as:

- $v_{L_2}(x_0, c_j) = \left(d(x_1^j, x_0), d(x_2^j, x_0), \cdots, d(x_k^j, x_0)\right)$, a k-dimensional vector where $x_i^j \in kNN(x_0, c_j)$ is the *i*th nearest neighbor ($i = 1,2,\cdots,k$) of x_0 among the members of c_j in the training set, and $d(x_i^j, x_0) \equiv || x_i^j - x_0 ||_2$ is the L_2-norm of the difference between the two vectors[1];
- $v_{L_1}(x_0, c_j) = \left(d'(x_1^j, x_0), d'(x_2^j, x_0), \cdots, d'(x_k^j, x_0)\right)$, a k-dimensional vector where $d'(x_i^j, x_0) \equiv || x_i^j - x_0 ||_1$ is the L_1-norm of the difference between the two vectors;

[1] For rare categories, since the number of training instances in each category is small, there might not be *k* nearest neighbors. In such a case we duplicate the furthest neighbor so that the sub-vector reaches size *k*.

- $v_{\cos}(x_0, c_j) = \left(\cos(x_1^j, x_0), \cos(x_2^j, x_0), \cdots, \cos(x_k^j, x_0)\right)$, a k-dimensional vector whose elements are the cosine similarity of the corresponding vector pairs;
- $v_{\text{mod}}(x_0, c_j) = \left(d(\bar{x}_j, x_0), \cos(\bar{x}_j, x_0))\right)$, a 2-dimensional vector where $\bar{x}_j$ is the centroid (vector average) of all the positive training examples in category c_j.

Of course the listed features are not necessarily exhaustive for all possibly informative features but rather a set of concrete examples for illustrating the principle in our approach. The number of features in vector z is $3k+2$ per category, and the total of $(3k+2)m$ for m categories. Parameter k can be tuned on a held-out validation dataset which is typically in the range from 10 to 100. The size of such a meta-level feature set is much smaller than those typically found in current MLC methods.

More importantly, these meta-level features are induced in a supervised manner, taking the labels of training instances into account. Also, the meta-level features make a combined use of local information (through kNN-based features) and global information (through category centroids) in the training set. Vector $z \in Z$ synthetically represents a pattern for each instance about how it is related to multiple categories. Figure 1 illustrates the concept geometrically in a 2-D space. For simplicity we only plot the L2-norm links in this graph and omit the other types of links.

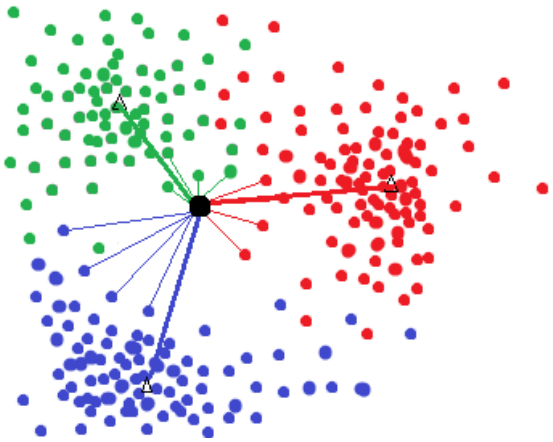

Figure 1: The link-based representation of one particular instance (the dot in the center) in relation to multiple categories is shown. Each category is represented using its positive examples (points in the same color) in the training set and its centroid (triangle). Each instance is represented using the "average links" (i.e., the distance to each category centroid, shown by thick lines) and multiple "single links" (i.e., the distance to each of the k nearest neighbors, shown by the thin lines) per category. These links together portray an informative picture about how the instance is related to multiple categories.

3. OUR APPROACH: SVM-MAP-MLC

3.1 Learning to rank for MLC

Our goal now is to solve the mapping $f^{MLC}: Z \to Y$ based on the transformed training set

$$T^{MLC} = \{(z_1, y_1), (z_2, y_2), \cdots, (z_n, y_n)\}$$

where $z_i = z(x_i) \in Z$ is an transformed input vector whose elements are meta-level features, and $y_i \in Y$ is a true ranked list of categories for the input[2]. Any learning-to-rank algorithm could be deployed in principle; among the successful ones, we choose SVM-MAP in the remainder of this paper.

SVM-MAP is originally designed for ranking documents with respect *ad-hoc* queries where ad-hoc means that queries could be any combination of words, not fixed in advance. The learning-to-rank problem is to find the optimal mapping $f^{IR}: Q \to Y$ where Q the input space (of all possible queries), Y as the output space (of all possible ranked lists of documents), and optimal means to minimize the training-set loss as well as the model complexity. A variety of learning-to-rank algorithms have been developed recently in machine learning for ad-hoc retrieval, with different loss functions and model selection criteria. SVM-MAP [24] is designed to maximize the Mean Average Precision (MAP) of ranked lists, which is a common metric for method comparison in IR evaluations. Methods focusing on other optimization criteria include a multivariate version of SVM [8] that maximizes ROC-Area for classification, MCRank and AdaRank [11] [19] that use boosting algorithms to maximize the Normalized Discounted Cumulated Gain (NDCG) of ranked lists, and so on.

Most learning-to-rank algorithms in ad-hoc 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. For example, SVM-MAP uses a conventional search engine (Indri) to produce the cosine similarity and language-model based scores for each query-document pair, discretizes those scores into bins, and treats the bins as the features in a dimension-reduced vector space where each query-document pair with relevance judgment is treated as a training instance. Optimizing the mapping from queries to ranked lists of documents therefore reduces to the learning of feature weights in the dimension-reduced vector space.

In order to apply SVM-MAP to MLC, or to apply any learning-to-rank retrieval algorithm to MLC in general, we need to find discriminative features to represent instance-category pairs and the one-to-many mapping from each instance to categories. The meta-level features 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. The category-specific links ($3k+2$ per category) in vector z (Section 2) and the corresponding label (yes or no with respect to the category) is treated as an instance-category pair in the training set. We focus

[2] There may be more than one true ranked list for an instance. That is, any list that places all the relevant categories above all the irrelevant categories is truly correct.

on SVM-MAP in this paper because it explicitly optimizes MAP which is one of the primary metrics we use in our evaluation of MLC methods (Section 4.3). SVM-MAP and other learning-to-rank retrieval methods have not been used for MLC before, to our knowledge. We name our novel application of SVM-MAP as SVM-MAP-MLC, in contrast to its conventional use in ad-hoc information retrieval.

3.2 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 [21],[5], among which we choose the linear regression approach proposed in [5]. Unlike binary-SVM where the natural choice of threshold is zero and probabilistic classifiers (such as logistic regression, Naïve Bayes, IBLR, etc.) where the default choice of threshold is 0.5, SVM-MAP-MLC produces non-probabilistic scores to rank categories with partial-order preferences. Obviously, neither 0 nor 0.5 is the appropriate optimal threshold on a ranked list of categories given an instance. The strategy proposed in Rank-SVM [5] is designed for optimizing the threshold conditioned on each ranked list. That is, the system uses a training set to learn a linear mapping from an arbitrary ranked list of categories to the optimal threshold as:

$$g : L \rightarrow T$$

where $L \subseteq R^m$ is the space of all possible vectors of system-scored categories, and $T \subseteq R$ 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 total error, i.e. the sum of type-I errors (the false positives) and type-II errors (the false negatives). Such a training set can be automatically generated by 1) learning a SVM-MAP-MLC model to score all categories conditioned on each input instance, and 2) finding the optimal threshold for each vector of scored categories given an instance. The LLSF function is applied in the testing phase, to the system-scored categories conditioned on each test instance. The resulting threshold is then applied to the corresponding ranked list of categories: the categories above or at the threshold receive a yes decision, and the categories below the threshold receive a no decision. We modified the original method by rescaling scores of each instance to make it in the unit norm.

4. EVALUATION DESIGN

4.1 Methods for Comparison

We conducted controlled experiments to evaluate SVM-MAP-MLC[3] in comparison with the following methods:

- **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) [5] [9][4].
- **Rank-SVM,** the method proposed by [5], is representative among the model-based methods which explicitly optimize ranked lists of categories for MLC[5].
- **IBLR,** the instance-based method recently proposed by [3], is a new method in MLC evaluations. The method has two versions, one using kNN-based features only (IBLR-ML) and another (IBLR-ML+) using world-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 [3]. We therefore use the results of IBLR-ML for method comparison in the rest of the paper. We used the Weka [7] implementation provided by the authors.
- **ML-kNN,** the instance-based method proposed by [1] is another strong baseline in MLC evaluations [3]. We used the publicly available Weka implementation [7] of this method in our experiments.

All the systems produce a ranked list of categories given a test instance**.** The ranked lists can be directly evaluated using rank-based metrics, and indirectly evaluated based on binary classification decisions (yes or no on each category) after applying a threshold to each ranked list.

The choice of thresholding strategy depends on the nature of each classification method. In Binary-SVM, for each category, we used the conventional threshold of zero to obtain a yes/no decision for that category. In ML-kNN, IBLR-ML and Rank-SVM we follow the same thresholding strategies as proposed by the authors. Specifically, for ML-kNN and IBLR we set the threshold to 0.5, and for Rank-SVM we use the linear least squares fit solution as the threshold, as proposed in [5].

4.2 Datasets

We used five datasets[6] in our experiments, namely *emotion*, *scene*, *yeast*, *Reuters-21578* and *Citeseer*. These datasets form a representative sample across different fields and they vary in training-set size and feature-space size. All the datasets except *Citeseer* have been used in previous evaluations of MLC methods, with conventional train-test splits. We follow such conventions in order to make our results comparable to the previously published ones. Table 1 summarizes the datasets statistics:

- *Emotions* [16] is a multi-label audio dataset, 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* [1] is an image dataset. 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.

[3] We used the publicly available SVM-MAP software at http://projects.yisongyue.com/svmmap/ as the core algorithm.

[4] We used the publicly available SVMlight software package at http://svmlight.joachims.org/ in our experiments.

[5] We thank the authors of ML-kNN for sharing their implementation of Rank-SVM.

[6] The *emotions*, *scene* and *yeast* datasets were obtained from http://mlkd.csd.auth.gr/multilabel.html

Table 1. Dataset Characteristics

Dataset Name	Training Size	Testing Size	#Categories	Avg #Categories per instance	#Features
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

- *Yeast* dataset [5] 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[7]. 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 in this dataset, and randomly split 80% of the corpus into training and the rest as testing instances. The dataset will be made publicly available along with the publication of this paper.
- *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 in [21].

4.3 Metrics

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

- **Mean Average Precision** (MAP) [18] is a popular metric in traditional 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 $D = \{x_i\}_{i=1,\cdots,n}$ 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 L_i, and R_i be the set of categories relevant to instance x_i. MAP is defined as:

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

$$AP(x_i) = \frac{1}{|R_i|}\sum_{c\in R_i}\frac{\left|\{c' \in R_i, s.t., r_i(c') < r_i(c)\}\right|}{r_i(c)}$$

- **Ranking Loss (**RankLoss**)** is a popular metric for comparing MLC methods in ranking categories [14][3][5][1]. It measures the average number of times an irrelevant category is ranked above a relevant category in a ranked list as:

$$RankLoss(x_i) = \frac{1}{|R_i||\overline{R}_i|}\left|\{(c,c') \in R_i \times \overline{R}_i, s.t. r_i(c) > r_i(c')\}\right|$$

[7] http://citeseer.ist.psu.edu/directory.html

- **Micro-averaged** F_1 is a conventional metric for evaluating classifiers in category assignments to test instances [10],[22],[23] . The system-made decisions on test set D with respect to a specific category $c \in C \equiv \{c_1, \cdots, c_m\}$ 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:

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

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

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

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

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

Category-specific Recall $R_c = \frac{TP_c}{TP_c + FP_c}$

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

Both micro-averaged F_1 and macro-averaged F_1 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.

We choose the evaluation measures so as to evaluate the performance of both the ranking algorithms as well as the thresholding strategies. MAP and RankLoss measure how well a system ranks the categories; Micro-F_1 and Macro-F_1 evaluate the effectiveness of the thresholding strategies for making classification decisions.

4.4 Experimental Setting Details

All parameters are tuned to optimize MAP and the 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 in SVM-MAP-MLC, ML-kNN and IBLR-ML, and the regularization parameter in SVM-MAP-MLC, Binary-SVM, Rank-SVM. 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^{-3} to 10^{6}. For IBLR, on *Citeseer* and *Reuters-21578*, the best choice for the number of nearest neighbors through cross validation was found to be 300 and 190 respectively.

Feature selection was not performed on any of the datasets for any method. For term weighting in *Citeseer* and *Reuters* documents, we used a conventional TF-IDF scheme (namely "ltc") [22]. 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.

5. RESULTS

The results of all the systems on the five datasets are summarized in Table 2. The methods with the best scores are highlighted in bold, and the relative ranks among the methods on each dataset in a specific metric are provided inside parentheses. We report the value of 1-RankLoss instead of RankLoss just to make the scores consistent to each other, i.e., higher values are better. The total rank of each system is provided at the bottom line of the table.

5.1 Comparative Analysis

Let us call each line in Table 1 a *case*, which corresponds to a particular dataset and a specific metric. SVM-MAP-MLC is the best in 18 out of the 20 cases while Binary-SVM is the best on the two remaining cases. That is, the proposed approach consistently outperformed other methods in most cases.

Comparing Rank-SVM with binary-SVM, both are large-margin methods but the former outperformed the latter on 12 out of the 20 cases. These results are consistent with the previously reported evaluation on the Yeast dataset [5], showing some success of Rank-SVM by reinforcing partial-order preferences among categories. Comparing Rank-SVM with SVM-MAP-MLC, on the other hand, the latter outperformed the former in 19 out of the 20 cases although both methods have partial-order preferences in their objective functions for optimization. The advanced

Table 2. Results Summary: Each method is evaluated on five datasets using four 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.

Dataset	Metric	SVM-MAP-MLC	ML-kNN	Rank-SVM	Binary-SVM	IBLR-ML
Emotions	MAP	**0.8286** (1)	0.7936 (3)	0.7819 (5)	0.7842 (4)	0.8147 (2)
	1-Rankloss	**0.8631** (1)	0.8262 (3)	0.8228 (4)	0.8184 (5)	0.8469 (2)
	Micro-F1	**0.7285** (1)	0.6693 (3)	0.6423 (4)	0.6309 (5)	0.6738 (2)
	Macro-F1	**0.7201** (1)	0.6562 (3)	0.6096 (4)	0.5875 (5)	0.6593 (2)
Scene	MAP	**0.8796** (1)	0.8511 (5)	0.8519 (4)	0.8567 (3)	0.8580 (2)
	1-Rankloss	**0.9315** (1)	0.9069 (5)	0.9193 (2)	0.9179 (3)	0.9173 (4)
	Micro-F1	**0.7416** (1)	0.6986 (3)	0.6611 (5)	0.6753 (4)	0.7094 (2)
	Macro-F1	**0.7563** (1)	0.6916 (3)	0.6687 (5)	0.6742 (4)	0.7122 (2)
Yeast	MAP	**0.7662** (1)	0.7584 (3)	0.7577 (4)	0.7465 (5)	0.7599 (2)
	1-Rankloss	**0.8373** (1)	0.8284 (4)	0.8288 (3)	0.8010 (5)	0.8307 (2)
	Micro-F1	**0.6730** (1)	0.6249 (5)	0.6515 (2)	0.6313 (4)	0.6371 (3)
	Macro-F1	**0.4306** (1)	0.3361 (4)	0.3597 (3)	0.3240 (5)	0.3709 (2)
Citeseer	MAP	**0.7690** (1)	0.7329 (4)	0.7508 (2)	0.7357 (3)	0.6926 (5)
	1-Rankloss	**0.9298** (1)	0.9143 (3)	0.9216 (2)	0.8904 (5)	0.8906 (4)
	Micro-F1	**0.5955** (1)	0.5139 (4)	0.5673 (2)	0.5142 (3)	0.4478 (5)
	Macro-F1	**0.5969** (1)	0.4924 (3)	0.5640 (2)	0.4919 (4)	0.4352 (5)
Reuters	MAP	0.9423 (2)	0.9248 (4)	0.9390(3)	**0.9543** (1)	0.8537 (5)
21578	1-Rankloss	**0.99485** (1)	0.9935 (4)	0.9939(3)	0.99483 (2)	0.9691 (5)
	Micro-F1	0.8225 (3)	0.8140 (4)	0.8278(2)	**0.8708** (1)	0.7280 (5)
	Macro-F1	**0.5451**(1)	0.2864 (5)	0.4420(3)	0.5266 (2)	0.3093 (4)
Rank Total		23	75	64	73	65

Table 3. P-Values in signed-rank tests for comparing SVM-MAP-MLC with other methods ('*' denotes lower performance of SVM-MAP-MLC).

Dataset/Method	ML-kNN	Rank-SVM	Binary-SVM	IBLR-ML
Emotions	0.003	7.490e-4	2.50e-04	0.243
Scene	6.975e-07	0.001	.001	4.976e-04
Yeast	1.706e-04	7.603e-04	3.056e-06	0.005
Citeseer	9.815e-12	3.109e-04	2.776e-15	3.156e-26
Reuters-21578	1.970e-10	.452	.999*	3.399e-72

performance of SVM-MAP-MLC, evidentially, comes from the use of meta-level features instead of the conventional features as that in Rank-SVM.

Comparing SVM-MAP-MLC, 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 outperformed ML-kNN in 13 out of the 20 cases in our experiments; this is more or less consistent with the previous report by [3] in terms of the relative performance of the two methods. Nevertheless, both IBLR-ML and ML-kNN underperformed SVM-MAP-MLC in all the 20 cases, showing the advantage of using of the meta-level features in the learning-to-rank framework with SVM-MAP.

Comparing Rank-SVM with ML-kNN, the former outperformed the latter in 13 out of the 20 cases. The relative performance of these two methods compared to each other is different from the previously reported evaluation in [1] where ML-kNN outperformed Rank-SVM on average. In order to clarify this issue, we compared the performance of Rank-SVM with different values of its regularization parameter which controls the balance between the training-set loss and model complexity. We found Rank-SVM performed suboptimally with the default parameter setting,, and performed better when this parameter was tuned through cross validation. Our results of Rank-SVM are based on the properly tuned parameters, and this should explain why Rank-SVM performed stronger than ML-kNN in our experiments. Comparing Rank-SVM with IBLR-ML, each method outperformed the other in 10 out of the 20 cases. Thus the two methods have comparable performance; both are strong baselines for method comparison in MLC.

6. SIGNIFICANCE TESTS

We used the *Wilcoxon signed-rank test* to compare the performance of each method with that of SVM-MAP-MLC. Signed-rank 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. Due to the space limit of the paper, we only present the tests based on the MAP scores of the systems. Each test instance is treated as a random event, and the average precision scores of system-generated ranked lists over all test instances are used to compare each pair of systems. The null hypothesis is that the system being compared with SVM-MAP-MLC is equally good; the alternative is that SVM-MAP-MLC is better. The p-values are presented in Table 3 where the null hypothesis is rejected with strong evidence in most cases. That is, the performance difference is statistically significant in 17 out of the 20 tests if using 1% p-value as the threshold.

We did not use ANOVA tests for multi-set comparison of systems because such tests have a normal-distribution assumption about the data which is inappropriate for the performance scores we use for system comparison. The Friedman test has been recently advocated for comparing classifiers on multiple datasets [4] [6], which does not have the normal assumption. 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[8].

6.1 Experiments with Feature Subsets

In order to analyze the usefulness of different types of meta-level features (links), we conducted experiments with the following combinations: using the single links in L_1-norm only, using the single links in L_2-norm only, using the single links in cosine similarity, and using all the links -- including those in L_1-norm, L_2-norm, cosine similarity and the two average links per category. Figure 2 compares the performance of SVM-MAP-MLC under these conditions. The different types of links have complementary effects: the single links in L_1-norm are more useful in the datasets (*Emotions*, *Scene* and *Yeast*) whose feature-

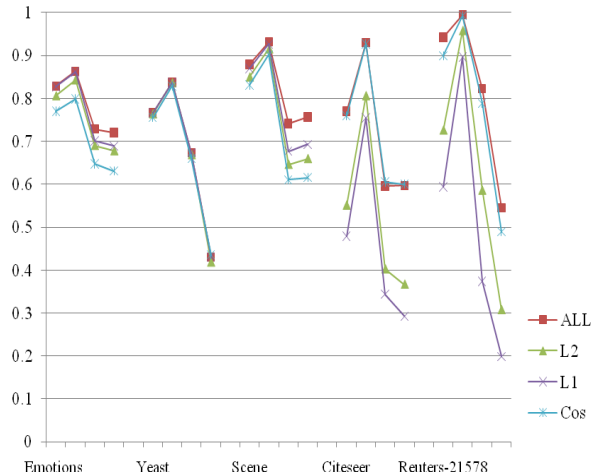

Figure 2. SVM-MAP-MLC with different feature sets (Performance in MAP, 1-RankLoss, Micro-F1, Macro-F1)

[8] This is according to personal communication with the author of [4].

set sizes are relatively small than they are on the datasets (*Citeseer* and *Reuters-21578*) whose feature-set sizes are large. On the other hand, cosine-similarity based single links have a different performance pattern; single links in L_2-norm have comparable performance across the datasets. Using all the features together performs the best, showing that SVM-MAP-MLC is able to assign appropriate weights to different features, and improve the robustness of its predictions by making a combined use of different features types.

7. CONCLUDING REMARKS

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 relationship from instances to categories in MLC, and by formulating the category ranking problem as a standard ad-hoc retrieval problem, our framework allows a broad range of learning-to-rank retrieval algorithms to be deployed for MLC optimization with respect to various performance metrics. Using SVM-MAP-MLC as a specific instantiation of this framework, and with controlled experiments on multiple benchmark datasets, the strength of the proposed approach is strongly evident, as it significantly outperformed all the state-of-the-art methods (Rank-SVM, ML-kNN and IBLR-ML) being evaluated in our experiments.

We hope 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 dimensionality reduction. A line of future research would be to explore our framework with other learning-to-rank algorithms, using different dimensionality reduction techniques (such as SVD or LDA), and for different optimization metrics (such ach NDCG and other types of loss functions).

ACKNOWLEDGEMENTS

This 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.

REFERENCES

[1] M.R. Boutell, J.Luo, X.Shen and C.M. Brown. Learning multi-label scene classification. *Pattern Recognition* 2004, pages 1757—1771

[2] Robert H. Creecy, Brij M. Masand, Stephen J. Smith, David L. Waltz: Trading MIPS and Memory for Knowledge Engineering. *Communications of the ACM* 1992, pages 48-64 (1992)

[3] W.Cheng and E.Hüllermeier. Combining Instance-Based Learning and Logistic Regression for Multilabel Classification. *Machine Learning* 2009, pages 211-255.

[4] J. Demsar. Statistical comparisons for classifiers over multiple data sets. *Journal of Machine Learning Research* 2006, pages 1-30.

[5] A Elisseff and J. Weston. A kernel method for multi-labeled classification. *Advances in Neural Information Processing Systems* 2002, pages 681-687.

[6] S Garcıa and F. Herrera. An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. *Journal of Machine Learning Research* 2008, pages 2677-2694.

[7] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I.H. Witten; The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, Volume 11, 2009.

[8] T. Joachims: A support vector method for multivariate performance measures. *International Conference on Machine Learning* 2005, pages 377-384.

[9] T. Joachims, Making large-Scale SVM Learning Practical. *Advances in Kernel Methods - Support Vector Learning*, MIT-Press, 1999.

[10] D.D. Lewis, R.E. Schapire, J.P. Callan and R.Papka. Training algorithms for linear text classifiers. *ACM SIGIR* 1996, pages 298-306.

[11] F. Li and Y. Yang. A loss function analysis for classification methods in text categorization. *International Conference on Machine Learning 2003*, pages 472-479.

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

[13] S Har-Peled, D Roth and D Zimak. Constraint classification: a new approach to multiclass classification and ranking. *Advances in Neural Information Processing Systems* 2002, pages 365-379.

[14] R.E. Schapire and Y. Singer . BoosTexter: A boosting-based system for text categorization. *Machine learning* 2000, pages 135-168.

[15] R.E. Schapire and Y. Singer . Improved boosting algorithms using confidence-rated predictions. *Machine learning* 1999, pages 297-336.

[16] K. Trohidis ,G. Tsoumakas, G. Kalliris, I. Vlahavas. Multilabel classification of music into emotions *International Conference on Music Information Retrieval 2008.*

[17] V. Vapnik, *The nature of statistical learning theory,* Springer verlag, New York, 2005.

[18] E.Voorhees Overview of TREC 2002, *NIST Special Publication* SP 2003.

[19] *Jun Xu and* Hang Li. AdaRank: a boosting algorithm for information retrieval. *ACM SIGIR 2007*, pages 391-398.

[20] Y. Yang: Exper Network: Effective and Efficient Learning from Human Decisions in Text Categorization and Retrieval. *ACM SIGIR 1994*, pages 13-22.

[21] Y. Yang. A Study of thresholding strategies for text categorization. *ACM SIGIR 2001*, pages 137-145.

[22] Y. Yang. An evaluation of statistical approaches for text classification. *Information Retrieval 1999*. Vol 1, No. ½, pages 67-88.

[23] Y. Yang and J.O. Pederson. A comparative study of features selection in text categorization. *International Conference on Machine Learning* 1997, pages 412-420.

[24] Yisong Yue, T. Finley, F. Radlinski and T. Joachims. A Support Vector Method for Optimizing Average Precision. *ACM SIGIR Conference* 2007.

[25] ML Zhang and ZH Zhou. ML-kNN: A lazy learning approach to multilabel-learning. *Pattern Recognition* 2007, pages 2038-2048