Making Use of Category Structure for Multi-class Classification

Inaugural-Dissertation

zur
Erlangung des Doktorgrades der
Mathematisch-Naturwissenschaftlichen Fakultät
der Heinrich-Heine-Universität Düsseldorf

vorgelegt von

Hieu Quang Le

aus Vietnam

März 2010

Aus dem Institut für Informatik
der Heinrich-Heine-Universität Düsseldorf

Gedruckt mit der Genehmigung der
Mathematisch-Naturwissenschaftlichen Fakultät der
Heinrich-Heine-Universität Düsseldorf

Referent:	Prof. Dr. Stefan Conrad Heinrich-Heine-Universität Düsseldorf
Koreferent:	Prof. Dr. Martin Mauve Heinrich-Heine-Universität Düsseldorf

Tag der mündlichen Prüfung: 07.04.2010

Nothing is more valuable than independence and freedom.
Ho Chi Minh

Acknowledgments

My PhD study was long and not easy for me. I know I would not able to finish this thesis without the help of many people.

First of all, I would like to express my gratitude to Prof. Stefan Conrad, my advisor and first referee, for his patient guidance and the friendly working environment throughout my study and research, as well as for his help in my thesis writing. I would also like to express my appreciation to Prof. Martin Mauve, my second referee, for taking time to read this thesis and finishing his review in a short time.

In this world of billions people, this is a rare chance for me to meet and work with the old and new colleagues at the Databases and Information Systems group. My special thanks go to Johanna Vompras and Marga Potthoff for helping me not only at work but also in my private issues, to Guido Königstein for fulfilling all my technical and networking needs, and to Sabine Freese for her administrative assistance.

I was supported by the Vietnamese Ministry of Education and Training, and by the DAAD – German Academic Exchange Service – during my study. I sincerely thank them for giving me the opportunity to stay and study in Germany – the country of the great people I admire and where I met friends all over the world.

I deeply thank to the Yoga masters, who are the authors of the books and articles I have read. Their practices help me to gain necessary health, concentration and creativity so that I can complete my research.

This thesis is dedicated to my family. Their unconditional love, continuous encouragement and support have turned all my difficulty into opportunity so that I can grow up and become what I might be. I would also like to thank to Tran Thanh Hai and my other Vietnamese friends for their help of all kinds.

My PhD journey is about to end. Looking back, the words of Buddha echo in my mind: “There is no way to happiness. Happiness is the way”, which I translate from

the German “Es gibt keinen Weg zum Glück. Glück ist der Weg”. So the wish from my true heart to all of us is that we are always happy on the way we are going.

AUM

Abstract

Multi-class classification is the task of organizing data samples into multiple predefined categories. In this thesis, we address two different research problems of multi-class classification, one specific and the other general.

The first and specific problem is to categorize structured data sources on the Web. While prior works use all features, once extracted from search interfaces, we further refine the feature set. In our approach, we use only the text content of the search interfaces. We choose a subset of features, which is suited to classify web sources, by our feature selection technique with a new metric and selection scheme. Using the aggressive feature selection approach, together with a multi-class Support Vector Machine categorizer, we obtained high classification performance in an evaluation over real web data.

The second and general task is to develop a multi-label classification algorithm. In a multi-label classification problem, a data sample can be assigned to one or more categories. Given a multi-label problem of m categories, the commonly used One-Vs-All (OVA) approach transforms the problem into m independent binary classifications between each category and the rest (the category's complement). Based on the OVA approach, we propose a new method named Multi-Pair (MP). This MP method decomposes further each of the OVA binary classifications into multiple smaller and easier pair comparisons between a category and a subset of the category's complement. Furthermore, we incorporate the SCutFBR.1 thresholding strategy into the MP method. In our experiments with three benchmark text collections, the MP method outperforms the OVA approach in both cases with and without SCutFBR.1.

A common aspect of our works is that we make use of category structure in our feature selection and multi-label classification methods. This is the aspect that distinguishes our works from prior researches.

Zusammenfassung

Multi-Class–Klassifikation bezeichnet die Aufgabe, Datenobjekte mehreren vorgegebenen Kategorien zuzuordnen. In dieser Dissertation werden ein spezielles und ein allgemeines Klassifikationsproblem aus diesem Bereich behandelt.

Die erste Problemstellung besteht in der Kategorisierung strukturierter Datenquellen im Web. Während frühere Arbeiten alle Eigenschaften (Features) verwenden, die von den Anfrageschnittstellen der Datenquellen extrahiert werden können, verfeinern wir die Menge der Eigenschaften. In unserem Ansatz verwenden wir nur den Textinhalt der Anfrageschnittstellen. Wir wählen mit Hilfe unserer Feature-Selection–Technik, einer neuen Metrik und einem neuen Selection-Schema eine Teilmenge der Eigenschaften aus, die geeignet ist die Web-Quellen zu klassifizieren. Unter Einsatz dieses "aggressive feature selection"–Ansatzes zusammen mit einem Multi-Class Support Vector Machine–Kategorisierer erhalten wir eine hohe Klassifikationsgenauigkeit in der experimentellen Evaluation mit realen Daten aus dem Web.

Die zweite Aufgabe ist es einen Multi-Label–Klassifikationsalgorithmus zu entwickeln. In einem Multi-Label-Klassifikationsproblem kann ein Datensatz zu einer oder mehreren Kategorien zugeordnet werden. Für ein gegebenes Multi-Label–Problem mit m Kategorien transformiert der allgemein verwendete One-Vs-All–Ansatz (OVA) das Problem in m unabhängige binäre Klassifiaktionsprobleme zwischen jeder Kategorie und dem Rest (d.h. dem Komplement dieser Kategorie). Ausgehend vom OVA-Ansatz schlagen wir eine neue Methode vor, die wir Multi-Pair (MP) nennen. Diese MP-Methode zerlegt die binären OVA-Klassifikationen weiter in kleinere und leichtere Vergleichspaare zwischen einer Kategorie und einer Teilmenge ihres Komplements. Darüber hinaus nutzen wir die SCutFBR.1-Thresholding–Strategie in unserer MP-Methode. In unseren Experimenten mit drei Benchmark-Text-Kollektonen ist die MP-Methode sowohl mit als auch ohne SCutFBR.1 dem OVA-Ansatz überlegen.

Das gemeinsame Merkmal unserer Arbeiten ist, dass wir die Struktur der Kategorien sowohl in unserem Feature-Selection- als auch in unserem Multi-Label-

Klassifikatonsansatz ausnutzen. Hierin unterscheiden wir uns deutlich von anderen Forschungsarbeiten auf dem Gebiet.

Contents

List of Abbreviations

BC	Binary Classifier
BEP	Break Even Point
DAG	Decision Directed Acyclic Graph
$\mathbf{F}_1$	F-measure
FS	Feature Selection
MP	Multi-Pair
MPC	Multi-Pair Classifier
OVA	One-Vs-All
SVM	Support Vector Machine
SWS	Structured Web Source

Chapter 1

Introduction

1.1 Motivation and Contributions

Multi-class classification is the task of organizing data samples into multiple predefined categories. A traditional example is that in a library, books are grouped by their subjects. Nowadays, another common example is found in the Internet, where an electronic news can be assigned simultaneously to several categories on a website. In this thesis, we address two different research problems of multi-class classification, one specific and the other general.

Categorizing structured web sources is the first and specific problem. A structured web source is a website that stores information in form of structured data with attribute-value pairs; and it usually provides search interfaces so that users can query its database [Chang et al., 2004]. There are a large number of such web sources, forming an important part of the huge Deep Web [Bergman, 2001]. On the one hand, integrated access over multiple sources is needed. For examples, a user may want to compare prices of a book in different online shops; or s/he may buy air tickets and book a room in a hotel online while preparing for a trip. On the other hand, there have been researches on data integration of a relative small number of heterogeneous sources [Chawathe et al., 1994, Levy et al., 1996]), as well as on large-scale search over multiple text databases [Callan et al., 1999, Ipeirotis et al., 2001]. Consequently, projects aiming at providing integrated data access to a large number of structured web sources, such as MetaQuerier [Chang et al., 2005] and WISE [He et al., 2005], have been born. Building and maintaining such large-scale accessing services involves a number of tasks: source finding and categorization, schema mapping and query translation, data extraction and integration, and so on. The categorization task is an integral part of these projects, as sources that have been collected must be grouped according to similarity before other tasks, such as

schema mapping [He and Chang, 2003] or query interface integration [Wu et al., 2004], can be performed.

In this task of categorizing structured web sources, our two contributions are as follows. First, we propose a feature selection (FS) technique with a new metric and selection scheme. Second, in term of classification approach, we use either our or others' [Yang and Pedersen, 1997] FS technique to refine features, once extracted from the search interfaces of the web sources. Meanwhile, prior works [He et al., 2004, Lu et al., 2006, Barbosa et al., 2007] use all features without further selection. Using the aggressive feature selection approach, together with a multi-class Support Vector Machine categorizer [Crammer and Singer, 2001], we obtained high classification performance in an evaluation over real web data.

Developing a multi-label classification algorithm is the second and general task. In a multi-label classification problem, a data sample can be assigned to one or more categories. This situation usually happens in text categorization. As in the mentioned Internet example, a news of the U.S. health care reform can be classified into three categories *Politics*, *Health* and *Business* on Yahoo! News website. Multi-label problems are also found in medical diagnosis, protein function classification, music categorization and semantic scene classification [Tsoumakas and Katakis, 2007].

In this task of developing an algorithm, we focus on improving the One-Vs-All (OVA) approach that is commonly used in researches, for instance, Joachims [1998], Yang [2001], Bekkerman et al. [2003], Lewis et al. [2004], Fan and Lin [2007]. Given a multi-label problem of m overlapping categories, the OVA approach transforms the problem into m independent binary classification tasks between each category and the rest (the category's complement). Based on the OVA approach, we develop a new method named Multi-Pair (MP). This MP method decomposes further each of the OVA binary tasks into multiple smaller and easier pair comparisons between a category and a subset of the category's complement. Via this decomposition, we aim at making use of category structure existing in the complement. The decomposition also helps to reduce the problem of imbalanced training data in the OVA approach. The MP method is our first contribution.

The MP method can be considered complementary to other multi-label approaches that have an OVA technique in their cores, for instance, Godbole and Sarawagi [2004], Fan and Lin [2007], Tang et al. [2009]. In this direction, we step further to incorporate SCutFBR.1, reportedly one of the most effective thresholding strategies [Lewis et al., 2004], into the MP method. The result is a combined method, our second contribution.

To evaluate our new approach, we conducted experiments with three benchmark text collections. In all the three text collections, the MP method outperforms the OVA approach in both cases with and without SCutFBR.1. Besides, source code to reproduce our classification results is available at http://dbs.cs.uni-duesseldorf.de/research/mp/ .

Making use of category structure is the common thread running through our works; and it distinguishes our works from prior researches. In a multi-label problem of m categories, the complement of a category is composed of around $(m-1)$ other categories. When merging samples together to create binary classification tasks, the OVA approach ignores this category structure. Meanwhile, the MP method takes it into account by decomposing OVA binary tasks. A similar difference exists between our and others' FS techniques for multi-class single-label classification. In our FS technique, one metric is defined for each feature by using "top" two categories. Also, features are divided into their respective categories before being selected. In others' FS technique [Yang and Pedersen, 1997], a score is defined by using every category and its complement. Given m categories, there are m scores for each feature; and the highest or average value is chosen. Then, all features are sorted and selected together.

1.2 Outline of this Thesis

The remainder of this thesis is organized as follows. Chapter 2 is an introduction to the classification process. In Section 2.1, we describe different classification types, that is, the categorization of classification problems themselves. In Section 2.2, we discuss our choice of the linear Support Vector Machine (SVM) method for the classification tasks at hand and available SVM implementation for each classification type. In this section, we also briefly introduce SVM basics. After that, Section 2.3 describes a general classification process, as well as related concepts such as training and testing data, data representation and normalization, training and prediction, and parameter tuning. Section 2.4 gives the definitions of the performance measures used in this thesis. Lastly, in Section 2.5, we describe the procedures of experimental evaluation with or without cross-validation, as well as the procedure of parameter tuning with cross-validation.

Chapter 3 is for the task of categorizing structured web sources; and chapter 4 is for the Multi-Pair method. These two chapters share the similar organizational structure. We open each chapter with the description of prior researches. This description provides a context to introduce our work subsequently. Then, in the first section of each chapter, we review related works with additional information and in wider scope. Next come the

chapter's main contents in the second and third sections that describe and explain our works at each task in detail. After that, we present our experimental setup, results and discussion. We close each chapter with a summary.

Chapter 5 is the last chapter of this thesis. In this chapter, we briefly review our study and highlight the important aspects of our works. Finally, we conclude this thesis with future work.

Chapter 2

Overview of Classification Process

This chapter is an introduction to classification process and related concepts: classification types, training and testing data, Support Vector Machine, performance measures, cross-validation and so on, for they are frequently used in this thesis.

2.1 Classification Types

Classification problems themselves can be divided into different types:

- **Binary classification** involves two categories; and a data sample is assigned to only one category. For example, we classify emails into two categories "*Spam*" and "*Not-spam*".
- **Multi-class single-label classification** involves two or more categories; and a sample is also assigned to only one category. An example is found in handwriting recognition, in which we identify an unique alphanumeric character for each digital image. Thus, binary classification is the special case of multi-class single-label classification, where the number of categories is two; and they are both ***single-label classification***.
- In a **multi-label classification** problem, there are two or more categories; but a sample can be assigned to more than one category. And we have already mentioned the example of Internet news. Though the number of categories can be two, multi-labeled datasets normally have more than two categories. For example, all multi-labeled datasets used in this thesis have 20 or more than 100 categories. Multi-label classification, together with multi-class single-label one, is ***multi-class classification*** which normally involves more than two categories and thus differs from binary classification.

2.2 Support Vector Machine Method and Implementation

A classification task usually involves training and testing data. ***Training data*** are data samples with their pre-assigned categories, while ***testing data*** are samples without categories. The goal of a categorizer is to learn a model from the training data in order to predict the target categories of the testing data.

For each classification task at hand, we have to select a suitable categorizer. In this thesis, we use the linear Support Vector Machines (SVMs) [Cortes and Vapnik, 1995]. This is because, on the one hand, all classification tasks in this thesis are either themselves or transformed into ***text categorization*** problems, or the tasks of organizing text documents into predefined categories [Sebastiani, 2002]. On the other hand, the linear SVM method has been applied successfully to text categorization [Joachims, 1998, Dumais et al., 1998].

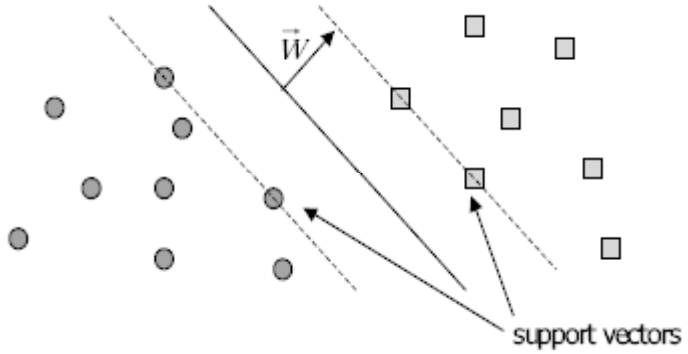

Figure 2.1: Linear Support Vector Machine [source: Dumais et al., 1998].

Linear SVM. We now briefly introduce SVM basics. [1] Let us consider a binary classification task whose training data consists of l data samples $\vec{x}_i$ with corresponding categories $y_i \in \{+1, -1\}$ $(i = 1, \ldots, l)$; each data sample is represented as a vector in a multi-dimensional feature space. In the linear form, the SVM method finds a plane that separates the set of positive samples from the set of negative samples with the maximum margin – see Figure 2.1. This is done through solving the following optimization

[1] Users interested in Support Vector Machines in more detail could read Bennett and Campbell [2000], an excellent SVM tutorial.

problem [Cortes and Vapnik, 1995]:

$$\begin{aligned} \min_{\vec{w},b,\vec{\xi}} \quad & \frac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{l}\xi_i \\ \text{subject to} \quad & y_i(\vec{w}\cdot\vec{x}_i - b) + \xi_i \geq 1\ , \\ & \xi_i \geq 0, \quad i = 1,\ldots,l\ ; \end{aligned}$$

where C is the trade-off between minimizing training error and maximizing margin. The result is the vector $\vec{w}$ that determines the orientation of the plane and the scalar b that determines the offset of the plane from the origin. And the classification function, or the learned model, is $y_* = sign(\vec{w}\cdot\vec{x}_* - b)$, where $\vec{x}_*$ is a testing sample.

SVM Implementation. To apply the linear (or nonlinear) SVM method to a classification problem, one can use available SVM implementation. For binary classification, there are a number of SVM binary categorizers in software packages, for example, SVMLIGHT [Joachims, 1999], LIBSVM [Chang and Lin, 2001], LIBLINEAR [Fan et al., 2008] or LIBOCAS [Franc and Sonnenburg, 2009].

For multi-class single-label classification, one can use in the SVM^{multiclass} [Tsochantaridis et al., 2004] or LIBLINEAR software package, which has the implementation of the multi-class SVM by Crammer and Singer [2001]. In LIBLINEAR, there is also the implementation of the single-label One-Vs-All approach [see Rifkin and Klautau, 2004]. The single-label One-Vs-All approach, as well as Decision Directed Acyclic Graph [Platt et al., 2000] and Error-correcting output codes [Dietterich and Bakiri, 1995], transforms a multi-class single-label problem into multiple binary classification tasks. One can easily implement these approaches oneself by using a prebuilt binary categorizer which can be the SVM or other classification methods.

For multi-label classification, to our knowledge, there is no stable implementation available. As mentioned in Chapter 1, the multi-label One-Vs-All approach is commonly used, and can be easily implemented by using a prebuilt binary categorizer which can be the SVM or other classification methods. In Chapter 4, we implement and compare this approach with our Multi-Pair method.

In this thesis, we treat the prebuilt SVM categorizers used as black box programs, providing them with input data and required parameters, and running them to obtain output result. For an SVM categorizer with a linear kernel (that is, a linear SVM categorizer), the only relevant parameter is C. However, this parameter is scaled differently in the SVM categorizers we used. Therefore, we first consulted the software manuals to

have a reasonable parameter range. Then we experimentally determined (or tuned) a suitable value for C through a cross-validation procedure – see Section 2.5.

2.3 Classification Process

In this section, we describe classification process by using an example, which is the simplified task of classifying emails into two categories "*Spam*" and "*Not-spam*". Though it is a binary task, its classification process in general is similar to those of multi-class single-label and multi-label classification tasks.

Now let us suppose that we have received many emails through our system each day; and we want to filter out junk emails by using only the text contents of the emails. We treat this task as a binary text categorization problem with the two predefined categories "*Spam*" and "*Not-spam*". Hence, we use a linear SVM binary categorizer, for example, available in LIBLINEAR. The classification process, which includes manual assignment, data representation, training and prediction, is described as follows.

Manual assignment. First of all, we have to create training data by selecting and manually classifying a part of the emails received into "*Spam*" or "*Not-spam*". The more manually classified emails, the better.

Data representation. Then we represent each training sample as a vector in the format required by the categorizer used. For our example, we use the common bag-of-words representation, and weight words (or terms) by the TF (term frequency) scheme [see Baeza-Yates and Ribeiro-Neto, 1999]. That is, the categories "*Spam*" and "*Not-spam*" are coded as $+1$ and -1 respectively. Every word is indexed uniquely, for instance:

```
..., corpus:15, ..., dataset:50, ..., newsgroup:500, ...
..., offer:12003, ..., reuters:16000, ..., special:20900, ...;
```

and the number of the word's appearance in the text contents of each email is counted.

As the result, we may have two (among many) emails represented in the LIBLINEAR's format as below:

```
−1  15:1  50:1  500:1  16000:1
+1  12003:3  20900:4
```

That is, the first email contains words 'corpus', 'dataset', 'newsgroups' and 'reuters', each word one time; and it is assigned to "*Not-spam*". The second email contains words 'offer' three times and 'special' four times; and it is assigned to "*Spam*".

For text categorization, we normally normalize data vectors, making their absolute values one. So after normalization, these two emails are represented as below:

```
−1  15:0.5  50:0.5  500:0.5  16000:0.5
+1  12003:0.6  20900:0.8
```

In the last step, we save all vectorized samples, for example, to a file `train_file`.

We note that in data representation for text categorization, we usually preprocess data, that is, lowercasing words, removing stop words such 'a' and 'the', and so on [see also Baeza-Yates and Ribeiro-Neto, 1999]. Depending on each problem, we may also apply a feature selection technique so as to have a better subset of features (see Chapter 3).

Training the categorizer. Having vectorized the training data, we use the data to train the categorizer, or in other words, to learn a new model. For example, with the above `train_file`, we call the `train` module of LIBLINEAR as below:

```
> train  -s 3  -c 1.0  -B 1  train_file  model_file
```

The result is a learned model, which is saved to the file `model_file`.

As can be seen in the above command, there are several parameters such as s (the type of solver) and c (trade-off between training error and margin). These parameters are normally determined through a cross-validation procedure (see Section 2.5).

Making prediction. After the training process has been completed, we are ready to predict (or classify) unassigned or new samples, that is, testing data. As for the training samples, we have to represent the testing samples as vectors. Since the true category of a testing sample is unknown, we simply use an arbitrary value, for example +1. Then, we employ the categorizer to classify the testing samples. Suppose that all testing samples are saved to a file `test_file`. We call the `predict` module of LIBLINEAR, with the above `model_file`, as below:

```
> predict  test_file  model_file  result_file
```

and obtain predicted results, which are saved to the file `result_file`.

2.4 Performance Measures

In this section, we define performance measures for each category individually and all categories together. For all categories, we first consider multi-label classification then single-label classification, for the latter case uses notations that are introduced in the former case.

Measures for a category. Following Sebastiani [2002], for an individual category, let TP_i denote true positives (the number of the samples that in fact belong to this category and are correctly assigned to this category); FP_i denote false positives (the number of the samples that in fact do not belong to this category, but are falsely assigned to this category); FN_i denote false negatives (the number of the samples that in fact belong to this category, but are falsely not assigned to this category); and TN_i denote true negatives (the number of the samples that in fact do not belong to this category and are correctly not assigned to this category). We define the five performance measures precision (P), recall (R), accuracy, break even point (BEP) and F-measure (F_1) for this category as below:

$$P_i = \frac{TP_i}{TP_i + FP_i} , \qquad R_i = \frac{TP_i}{TP_i + FN_i} , \qquad accuracy_i = \frac{TP_i + TN_i}{TP_i + FN_i + TN_i + FP_i} ,$$

$$BEP_i = \frac{P_i + R_i}{2} , \qquad F_{1i} = \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} = \frac{2 \cdot TP_i}{2 \cdot TP_i + FP_i + FN_i} .$$

Thus, BEP_i is the unweighted mean of P_i and R_i. Meanwhile, F_{1i} is called the harmonic mean. With the same BEP_i, the more balanced P_i and R_i, the higher F_{1i}.

Measures for multi-label classification. To evaluate performance of all m categories in a multi-label classification task together, we have two averaging approaches. The macro-F_1 is the average value of category F-measures:

$$macro\text{-}F_1 = \frac{1}{m} \cdot \sum_{i=1}^{m} F_{1i} .$$

For micro-average measures, all individual predictions are taken into account together:

$$micro\text{-}P = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FP_i)} , \qquad micro\text{-}R = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FN_i)} ,$$

$$micro\text{-}BEP = \frac{micro\text{-}P + micro\text{-}R}{2} , \qquad micro\text{-}F_1 = \frac{2 \cdot \sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(2 \cdot TP_i + FP_i + FN_i)} .$$

As discussed in Lewis et al. [2004], macro-averaging is dominated by small categories, whereas micro-averaging by large categories.

Measures for single-label classification. To evaluate performance of all m categories in a single-label classification task together, we use overall accuracy:

$$accuracy = \frac{TP}{TP + FN} = \frac{\sum_{i=1}^{m} TP_i}{\sum_{i=1}^{m}(TP_i + FN_i)} .$$

This overall accuracy can be proved to be equal to micro-P, micro-R, micro-BEP and micro-F_1 in the case of single-label classification. Hence, it is conventionally used in researches, for instance, Bekkerman et al. [2003], Gabrilovich and Markovitch [2004]. Besides, we note that in the case of binary classification ($m = 2$), this overall accuracy is the same as the accuracy defined for each (positive or negative) category.

2.5 Cross-validation Procedure

In this section, we describe cross-validation procedures [see Tan et al., 2005], which are used in parameter tuning and experimental evaluation.

Parameter tuning. As mentioned in the training step of the classification process, we usually have to determine one or more parameters required by the SVM categorizer used. This procedure is called parameter tuning, a major issue when working with SVMs as well with most of the other inductive learning algorithms [Bekkerman et al., 2003].

As the true categories of testing samples are not available, we tune required parameters through a cross-validation procedure over training data. Let us take the C parameter (trade-off between training error and margin) for LIBLINEAR as an example. First of all, we fix a small set of feasible values for the C parameter. As suggested in Hsu et al. [2009], we may use the set of $[2^{-5}, 2^{-3}, \ldots, 2^{15}]$ or a smaller subset of the set. Afterward, for each value of the feasible set, we evaluate its classification performance (for example, measured in accuracy) as follows. We divide the training data into f parts; so we have a f-fold cross-validation schema. Sequentially, we use $(f-1)$ parts to train the categorizer with the current parameter value, and employ the trained categorizer to test the other part. Thus, in the f-fold cross-validation schema, each sample of the whole training set is predicted once. Using this prediction result, we calculate the classification performance of the current parameter value. Among all values of the feasible set, we select the one that gives the best classification performance. Finally, we train the categorizer by using the whole training set and with the found parameter; and then classify testing samples.

When there are two parameters, for instance feature selection level (see Section 3.3.2) and C, we may tune the first while fixing the second (by setting it to a default value); and then determine the second once the first is found. Or we may search for the both parameters at the same time in a grid-search over all possible parameter pairs [see also Hsu et al., 2009].

Experimental evaluation. Let us now suppose that we want to experimentally evaluate performance of a classification method by using a benchmark dataset, which contains data samples together with their pre-assigned categories. There are two schemes to carry out experiments.

The first scheme is cross-validation, which is slightly different from that of parameter tuning. That is, we split the benchmark dataset into f parts, or in other words, we employ a f-fold cross-validation schema. For each fold, $(f-1)$ parts are used as the training set, while the other part is used as the ***validation set*** (that is, the fold's testing set). We use the training set to tune required parameters and train with the method. After that, we categorize the validating samples and compute the classification performance of the method on the validation set. We then report the average performance of all f folds as the final result.

The second scheme is that we separate the benchmark dataset into two parts. One part is used as the training set. The other part is used as the validation set on which we calculate and report classification performance. Furthermore, instead of one training-testing pair, we may generate two or more pairs, each consists of a training set and a corresponding testing set. In this case, we report the average classification performance of the testing sets.

In this thesis, we will use both the evaluation schemes. This is because we use preprocessed datasets made available by prior researches. And for each dataset, we follow the research that is the source of the dataset.

Chapter 3

Categorizing Structured Web Sources Using Aggressive Feature Selection

As discussed in Section 1.1, the task of categorizing structured web sources in an integral part of the projects that provide integrated data access to a large number of these web sources. In this chapter, we address this categorization task.

The search interfaces of structured web sources, which serve as the "entrances" to underlying databases, are used as the main source for the categorization task. In He et al. [2004], the authors argued that the form labels of search interfaces (for instance, 'Title', 'ISBN(s)', ... in an Amazon's search form) are the right "representatives" for structured sources, and used only them. Subsequently, in addition to form labels as the most important feature, Lu et al. [2006] identified and utilized other features such as form values (for instance, 'hardcover', 'paperback') and other regular text terms. In these two works, features inputed to their clustering algorithms must be extracted from HTML pages by another technique [see also Lu et al., 2006]. In contrast, Barbosa et al. [2007] argued that such an extraction task is hard to automate, so they used all the text (bag-of-words) of a search interface, which is partitioned into text of the form and text of the page, together with backlinks pointing to the interface.

A common issue in the prior works is that features, once extracted, are all used without any further selection. However, it is not difficult to see that in a search interface, words that help in distinguishing categories (for instance, 'author', 'textbooks') mingle with many more other words. Indiscriminative or noisy terms (for instance, 'sort by', 'state') also occur inside forms, as observed by Lu et al.. For these reasons, we investigate on how to identify features suitable for categorizing structured web sources, that is, the feature selection (FS) problem.

Our classification approach employs a filtering FS technique in text categorization [Sebastiani, 2002], together with a multi-class Support Vector Machine categorizer [Crammer and Singer, 2001]. In our research, we use only the text content of a search interface. To choose a suitable subset of terms, we conducted experiments with others' FS metrics and techniques [Yang and Pedersen, 1997, Mladenic, 1998, Forman, 2003, Gabrilovich and Markovitch, 2004], and with ours as well (see Section 3.3.2). The FS technique that we propose includes a new metric and selection scheme. [1] As pointed out in Barbosa et al. [2007], it is prone to make clustering mistakes among domains with overlapping vocabulary such as *Movies* and *Musics*, and for domains with a highly heterogeneous vocabulary. This new FS technique is designed to tackle these issues.

In the mentioned experiments with the existing and new FS techniques, we obtained high classification performance with the subsets selected, which is significantly higher than the performance obtained when using the much larger set of all features. This result does not only show that our aggressive feature selection approach has its own strength, but is also a convincing evidence that extracted features should be further selected. In addition, we achieved the best performance with the new FS technique.

The rest of this chapter is organized as follows. In the following section, we review related work. Then we describe the classification process for structured web sources in Section 3.2, the existing and new FS techniques in Section 3.3. In section 3.4, we present our experimental results and discussion. Finally, we conclude with a summary in Section 3.5.

This chapter is based on the paper Le and Conrad [2010a].

3.1 Related Work

In this section, we relate our work to other categorization problems, to the researches on the same topic of structured web source (SWS) categorization and on feature selection.

Firstly, SWS categorization is related to text database classification [Callan et al., 1999, Ipeirotis et al., 2001], for they work with the Deep Web's sources. A text database normally stores documents of multiple categories. Its interfaces contain simple search forms, which often have only one label and field, and little information about stored documents. Therefore, to have the representative summary of a text database, query

[1] We use the phrase of "FS metric" to indicate a scoring formula, and "FS technique/method" to indicate a scoring formula with a selection scheme.

submission techniques that send queries to sources and analyze returned results are needed. In contrast, a structured database stores data objects of a single domain. Its search interfaces provide much information, such as form labels and values (describing exported schema), advertised products (that is, data items), which can be used to classify the database. Furthermore, it is necessary that the domain of a structured source, which contains complex and multi-label search forms, is known before a query submission technique can be applied. Since utilizing search interfaces, SWS categorization is also related to web page categorization [Chakrabarti et al., 1998, Zamir and Etzioni, 1998] which uses terms and links, and to text categorization [Joachims, 1998], which uses only terms. However, the goal of SWS categorization is not to classify a search page (that is, an HTML page) itself but the database, to which the page's search form serves as an "entrance". As a result, discriminative features extracted from or related to search forms are most important to the task.

Secondly, the mentioned works of He et al., Lu et al., Barbosa et al., together with ours, are on the same topic of SWS categorization. While the others' works show that it is feasible to determine the domain of a source by using discriminative features extracted from source's search interfaces, we take a step further. We refine the set of features once extracted, and use the refined set in order to increase classification performance. In addition, the prior studies employed clustering (that is, unsupervised learning) algorithms, while we use a classifying (that is, supervised learning) method. One reason is that we group web databases so that other integrating tasks can be performed. There is no online requirement as, for example, in search engines where web documents may need to be clustered dynamically within a few seconds in response to a user's need [Zamir and Etzioni, 1998]. Thus, the emphasis of SWS categorization is on accuracy. The other reason is that our goal is to categorize a large number of sources. It is appropriate that we build an initial domain hierarchy, either by manual assignment or by clustering together with manually checking, from a small number of sources. Afterward, we classify the rest so as to make use of the data samples better through a learning process.

Lastly, the problem of feature selection in text categorization has been intensively studied, for example, in Yang and Pedersen [1997], Mladenic [1998], Soucy and Mineau [2001], Forman [2003], Gabrilovich and Markovitch [2004]. These and our studies use the filtering approach [see Sebastiani, 2002]. As our FS technique are proposed for multi-class single-label classification, our work is most closely related to the work of Yang and Pedersen. In Section 3.3, we will present our FS technique, along with its most similar one – χ^2 [Yang and Pedersen, 1997].

3.2 Classification Process for Structured Web Sources

In our research, each complete search interface is treated simply as a text document, that is, a bag-of-words extracted from its HTML content. Similar to the prior studies [He et al., 2004, Lu et al., 2006, Barbosa et al., 2007], we assume that one web database, together with its search interfaces, belongs to one category. Since a document representing a web source belongs to one category, and there are multiple categories, our problem is equivalent to a single-label multi-class text categorization problem [see Sebastiani, 2002]. [2]

The process of categorizing structured web sources in general is similar to the classification process presented in Section 2.3. In the following paragraphs, we describe its specific details.

Search interface processing. Following [Barbosa and Freire, 2005], we identify terms in forms that are between $FORM$ tags, and terms in pages that are within $HTML$ tags. We further use terms in titles that are between $TITLE$ tags separately. After parsing HTML search interfaces to extract words in these three types, we do not apply a stemming algorithm [for instance, Porter, 1997] to reduce words with the same stem to a common form. The reason is that, for example, the word 'book' appears frequently in both $Airfares$ and $Books$ domains, while the word 'book$s$' is often found in $Books$ but not in $Airfares$. Therefore, these two words should be seen as two different discriminative features instead of being merged by a stemming algorithm. In addition, terms that appear less than some small K times in every category are to be eliminated. This technique is to remove noisy words.

Feature selection (FS). After processing search interfaces, we apply a multi-class FS technique in order to identify terms (that is, processed words) that are suited to categorize web sources. The multi-class FS techniques we used are presented in Section 3.3. Basically, these techniques score terms by a metric and then select top N ranked terms by a selection scheme. Or in other words, they are the filtering approach, as mentioned in Section 3.1.

We rank and select terms of each form, page and title feature types separately. After that, we put all selected terms of these feature types into a common feature space. (So the same term in different feature types will be represented and chosen differently in our

[2] Our approach can be easily adapted to a multi-label case, in which an interface is to be classified into several categories, by using the multi-label One-Vs-All or Multi-Pair method (see Chapter 4).

classification approach.) In our implementation, we use the same top N parameter for all the three feature types, where N is determined through a cross-validation procedure.

Term weighting and vector normalization. We weight terms by a non-weighted feature scheme and then normalize each data vector, for we obtain the best performance with this simple technique. In the non-weighted feature scheme, if a feature appears in a sample, the feature's weight in the data vector representing the sample is 1; and if the feature does not appear, its weight is 0. We observe that the distinctive terms of a domain (for instance, 'ISBN', 'paperback' and 'hardcover' for *Books*) often appear only once in a search interface as its form labels or values. This observation helps to explain why the non-weighted feature scheme is suitable for the categorization task at hand.

Categorizer and kernel selection. After feature selection and data representation, we move onto the training and prediction steps. For these steps, we choose the SVM method with a linear kernel. As mentioned in the Section 2.2, this choice has been used successfully in text categorization.

3.3 Feature Selection Techniques

In this section, we first describe χ^2 – an existing multi-class FS method related to our new T2CS-CHI technique. After that, we present the T2CS-CHI technique.

3.3.1 χ^2 (CHI)

Let $\{A, B, \ldots\}$ be the set of m ($m \geq 2$) categories. The χ^2 technique [Yang and Pedersen, 1997] is described as follows.

χ^2 metric. For a feature t, its score is calculated in the next two steps:

1. For each category, say C, compute the category-specific score of t with regard to C by the below formula [Sebastiani, 2002]:

$$\chi^2(t, C) = |Tr| \cdot \frac{[P(t, C) \cdot P(\bar{t}, \overline{C}) - P(t, \overline{C}) \cdot P(\bar{t}, C)]^2}{P(t) \cdot P(\bar{t}) \cdot P(C) \cdot P(\overline{C})} ,$$

   where $|Tr|$ denotes the total number of samples in the training set, $P(t, C)$ the probability that a random document contains a feature t and belongs to a category C, and $\overline{C}$ the complement of a category C.

2. Choose the highest value of the category-specific scores as the final score of t:

$$\chi^2_{max}(t) = \max_{C \in \{A,B,\ldots\}} \chi^2(t, C) \ .$$

When calculating the score of a feature t, instead of the highest value of the category-specific scores, we can choose the weighted average value as the final score of t. As reported in Rogati and Yang [2002], the highest value performs better than the average one across categorizers and text collections. Therefore, in this thesis, we do not use the average value.

Selection scheme. After calculating the scores of all features, the features are sorted together in descending order and selected top-down.

3.3.2 T2CS-CHI

In this section, we present the T2CS-CHI (Top-two-category separation – χ^2) technique which includes a new metric and selection scheme.

T2CS-CHI metric. We first describe the rational behind the T2CS-CHI metric through an example with three categories $Airfares$, $Books$ and $Musics$. When a document contains the word 'music', together with other words, such as 'title', 'elvis' and 'surround', we can normally determine quickly that its category is either $Musics$ or $Books$, but not $Airfares$. We observe that $Musics$ and $Books$ are the two categories whose documents most frequently contain the word 'music'. Moreover, their vocabularies usually have a large overlap. Since the main categorization difficulty is in deciding between them, we are interested in how much 'music' helps to classify documents into $Musics$ or $Books$. We thus calculate the score of 'music' in two steps. At first, we select only the "top" two categories $Musics$ and $Books$ by the above observation that the word 'music' most frequently appears in them. Afterward, we compute the score of 'music' with regard to these "top" two categories by a formula that is derived from the category-specific χ^2 scoring formula.

Formally, let $\{A, B, \ldots\}$ be the set of m ($m \geq 2$) categories; $P(t|C)$ be the conditional probability that a random sample in a category C contains a feature t; $P(C|t)$ be the conditional probability that a random sample containing a feature t belongs to a category C; $P(C)$ be the probability that a random sample belongs to a category C. Let C_1 and C_2 denote the two categories that have the first and second highest values of all probabilities $P(t|C)$ for $C \in \{A, B, \ldots\}$. (In the case of two categories with the

same probability $P(t|C)$, we choose the one with the higher probability $P(C)$.) The score of a feature t is given in the following formula:

$$T2CSCHI(t) = [P(t|C_1) - P(t|C_2)] \cdot [P(C_1|t) \cdot P(C_2|\bar{t}) - P(C_2|t) \cdot P(C_1|\bar{t})] \,. \quad (3.1)$$

We now compare the χ^2 metric with the T2CS-CHI metric. It can be proved that:

$$T2CSCHI(t) = \frac{[P(t, C_1) \cdot P(\bar{t}, C_2) - P(t, C_2) \cdot P(\bar{t}, C_1)]^2}{P(t) \cdot P(\bar{t}) \cdot P(C_1) \cdot P(C_2)} \,.$$

So in the case of binary classification ($m = 2$), T2CS-CHI is equivalent to χ^2, for they differ only in $|Tr|$ (the number of training samples) which is the same for all features. However, in the case of multi-class classification ($m > 2$), T2CS-CHI and χ^2 are different from each other. That is, the T2CS-CHI metric is defined from the "top" two categories, whereas the χ^2 metric from a category and its complement.

Let us consider the above example with the words 'music', 'elvis', 'surround', 'title' again. Statistics on the dataset we used shows that 'music' appears very frequently in $Musics$, relatively frequently in $Books$; 'elvis' relatively frequently and only in $Musics$; 'surround' rarely and only in $Musics$; 'title' very frequently in $Books$ as well in $Musics$. With the T2CS-CHI metric, 'surround' and 'title' have low ranks and will usually be discarded. With the χ^2 metric, similar situation happens to 'surround', but not to 'title'. The χ^2 metric scores 'title' high, as it may distinguish $Books$ (and $Musics$) from $Airfares$. Meanwhile, the T2CS-CHI metric scores 'title' low as its usage may increase the mis-classification between $Books$ and $Musics$.

Lastly, we give here an intuitive explanation for the scoring formula (3.1). That is, with the first term of this formula, the more frequently a feature t appears in the category C_1 (or the higher $P(t|C_1)$), the bigger its score. (Note that C_1 is the category with the highest of all probabilities $P(t|C)$.) Likewise, with the second term of this formula, the higher the chance that a sample containing t belongs to C_1 (or the higher $P(C_1|t)$), the bigger the score of t; and also the less the chance that a sample not containing t belongs to C_1 (or the smaller $P(C_1|\bar{t})$), the bigger the score of t. In addition, we note that the first term $[P(t|C_1) - P(t|C_2)]$ is equal to $[P(t|C_1) \cdot P(\bar{t}|C_2) - P(t|C_2) \cdot P(\bar{t}|C_1)]$, for the sum of $P(t|C)$ and $P(\bar{t}|C)$ is 1.

Selection scheme. After calculating one score for each feature, we go on to choose a subset of features. While the straightforward selection scheme that is used by the χ^2 technique ranks and selects features altogether regardless of categories, we approach

differently. In the investigation of the straightforward selection scheme with a metric, T2CS-CHI or χ^2, we put each feature t of the top N features selected into the category C, among samples of which t most frequently appears (that is, $P(t|C)$ is the highest). Or in other words, we assign the feature t to the category C that t represents best. We observed that some categories get many more features assigned than other categories (for instance, *Automobiles* around 10 times higher than *Books*). This imbalance may make classification more error prone, since there may not be enough discriminative features to accept or reject whether a sample belongs to a category with the small number of features. The category with the small number, in turn, usually has a highly heterogeneous vocabulary. Therefore, we propose a new selection scheme, aiming at balancing the number of features representing each category, as follows:

1. Compute the scores of features by the T2CS-CHI metric; and assign each feature t to the category C with maximum $P(t|C)$. In the case of two categories with the same probability $P(t|C)$, we choose the one with the higher probability $P(C)$.
2. Sort features in each category by their scores in descending order; then re-rank all features together first by their relative ranks in their categories in ascending order, and second by their scores in descending order.
3. Select top N ranked features from the set of all features, where N is a parameter determined through a cross-validation procedure (see Section 2.5).

3.4 Experiments

In this section, we describe the dataset and multi-class SVM implementation we used, as well as other FS metrics and techniques that we experimented with. After that, we present the experiment results and use them to show the effect of feature selection and to compare the FS techniques used.

3.4.1 Dataset and Experimental Settings

We use the TEL-8 dataset of the UIUC Web integration repository [UIUC, 2003], which contains the search interfaces of 447 structured web sources classified into 8 domains. After converting HTML pages to text documents and manually checking, we kept 431 sources. The other sources were not usable because the offline contents of search pages required an online update while the pages themselves no longer exist on the Web. Table 3.1 describes the dataset that we used. We conducted experiments in a 4-fold

Domain	*# of sources*	*Domain*	*# of sources*
Airfares	43	Hotels	34
Automobiles	80	Jobs	50
Books	66	Movies	71
CarRentals	21	Musics	66

Table 3.1: Dataset of 431 web sources in 8 domains

cross-validation scheme. In the form processing step, we ignored words that appear less than three times in every category (that is, $K = 3$), so as to remove noisy words. For the problem at hand, web sources are to be discovered automatically by a crawler [see, for instance, Barbosa and Freire, 2005]. Due to the dynamic nature of the Web, we assume that there is no prior information in regard to the category of a new web source to be classified. [3] Therefore, in the implementation of FS metrics, we assigned the value $(1/m)$ to the probability $P(C)$ that a random document belongs to a category C, where m is the total number of categories.

For the multi-class SVM implementation, we used the $SVM^{multiclass}$ [Tsochantaridis et al., 2004]. Besides top N (FS level), the only parameter for the SVM linear kernel we use is *svm-C* (trade-off between training error and margin). We determine these two parameters through a 10-fold cross-validation procedure on each training set. [4]

Besides χ^2 and T2CS-CHI, we also conducted experiments with other FS metrics and techniques. The other FS metrics experimented are the binary version of Information Gain [Gabrilovich and Markovitch, 2004], Bi-normal separation [Forman, 2003], Odds Ratio [Mladenic, 1998], for they were reported as the most effective metrics by these authors. Like the χ^2 metric, these FS metrics are defined between a category and its complement. Therefore, we apply the FS procedure of χ^2 to these metrics; that is, we use one of these metrics to calculate the category-specific score of a feature for every category, selecting the maximum value of the category-specific scores as the final feature's score, and then sorting and selecting all features together. We also experimented with the multi-class Information Gain (IG) and Document Frequency (DF) techniques [Yang and Pedersen, 1997]. In the subsequent sections, we will discuss the T2CS-CHI, χ^2, IG

[3] In the TEL-8 dataset used by us and He et al., Barbosa et al., the ratio between the source number of the *Books* domain and that of the *Jobs* domain is 66 : 50 (1.32), while in another dataset used by Lu et al. the ratio is 85 : 20 (4.25). This substantial difference in the ratio of sources is additional support for our assumption.

[4] Specifically, the FS levels are in every 240 features, and for the *svm-C* parameter the feasible set is $\{20000, 40000, 60000, 80000, 100000\}$.

Method	*Accuracy*	*# of features*
All features after form processing	92.1 %	8030
T2CS-CHI	94.9 %	1320
χ^2	94.4 %	1020
IG	93.5 %	4650

Table 3.2: Classification performance.

and DF techniques, while omitting the others for their performance is either similar to or lower than that of χ^2 or IG.

3.4.2 The Effect of Feature Selection

Table 3.2 shows how aggressive feature selection affects classification performance. We report results at the FS level and *svm-C* parameters determined by using only training sets, and the number of features which is the average value of the training sets. Also, we report overall accuracy, as the problem at hand is multi-class single-label classification (see Section 2.4). When no FS methods are applied, that is, using all of around 8030 features after form processing, the accuracy obtained is 92.1%. This good result reflects a common observation that SVM categorizers can cope quite well with many redundant features [Joachims, 1998]. When applying our T2CS-CHI technique, the accuracy is 94.9%, which is the highest result. Those of existing FS techniques, χ^2 and IG, are 94.4% and 93.5% respectively. The FS levels of T2CS-CHI and χ^2 are not higher than 1320 features. Thus, these two FS techniques improve classification performance significantly while using the subsets that are much smaller than the set of all features.

Table 3.3 presents the detailed classification results of two techniques "All features after form processing" (column "a") and T2CS-CHI (column "b"), where Af, Am, Bk, Cr, Ht, Jb, Mv and Ms are abbreviation for 8 domains $Airfares$, $Automobiles$, $Books$, $CarRentals$, $Hotels$, $Jobs$, $Movies$ and $Musics$ respectively. In Table 3.3, for example, value 2 in the cell row Cr column Af-b means that two web sources, which in fact belong to $CarRentals$, have been assigned into $Airfares$ when using the T2CS-CHI technique. (A cell value is the sum of the results given by validation sets.) As can be observed in Table 3.3, the T2CS-CHI technique gives better or equal performance over all domains. Specifically, for the group of closely related domains $\{Airfares, CarRentals, Hotels\}$, it

	Af		*Am*		*Bk*		*Cr*		*Ht*		*Jb*		*Mv*		*Ms*	
	a	b	a	b	a	b	a	b	a	b	a	b	a	b	a	b
Af	38	41	0	0	0	0	3	2	2	0	0	0	0	0	0	0
Am	0	0	79	79	0	0	0	0	0	1	0	0	1	0	0	0
Bk	0	0	0	0	64	65	0	0	0	0	1	0	1	1	0	0
Cr	4	2	0	0	0	0	16	18	1	1	0	0	0	0	0	0
Ht	4	1	0	0	0	0	1	0	29	33	0	0	0	0	0	0
Jb	0	0	0	0	0	0	0	0	0	0	50	50	0	0	0	0
Mv	0	1	0	0	2	3	0	0	2	0	0	0	62	62	5	5
Ms	0	0	0	0	0	0	0	0	0	0	0	0	7	5	59	61

(a) All features after form processing, (b) T2CS-CHI.

Table 3.3: Detailed classification results.

is significantly better. And for *Books*, a domain with a highly heterogeneous vocabulary, T2CS-CHI shows a further improvement over the good result when using all features.

3.4.3 Comparison of FS Techniques

Figure 3.1 compares the accuracy of our T2CS-CHI technique and those of the existing χ^2, IG and DF techniques at different FS levels. To draw the graph, at each FS level we choose the best accuracy among the feasible values of the parameter *svm-C* in each cross-validation set, then use the average of accuracy across all cross-validation sets. Or in other words, the result shown in the graph is obtained by tuning *svm-C* over the whole dataset. Thus the previous accuracies reported in Table 3.2 can not be exactly seen from this graph, for they are calculated at the FS level and *svm-C* determined by using only training sets.

It can be observed in Figure 3.1 that T2CS-CHI and χ^2 perform with high accuracy around 95% in the optimal range of 500-2000 features. Next comes IG with the accuracy around 94% at the FS level of 4000 features. DF performs worst. In addition, T2CS-CHI maintains stable accuracy around its optimal level of 1000. Meanwhile, the accuracy of the χ^2 method decreases sharply after its peak at the level of 500, before increasing again around the level of 2000. The stability of the T2CS-CHI technique helps to explain why it gives the better accuracy at the automatically determined FS level than χ^2 does (see Table 3.2), although the highest accuracies of the two techniques shown on the graph are almost the same.

Consistently with prior studies in text categorization [Yang and Pedersen, 1997, Rogati and Yang, 2002], we find that rare terms are not important. Let us consider the DF

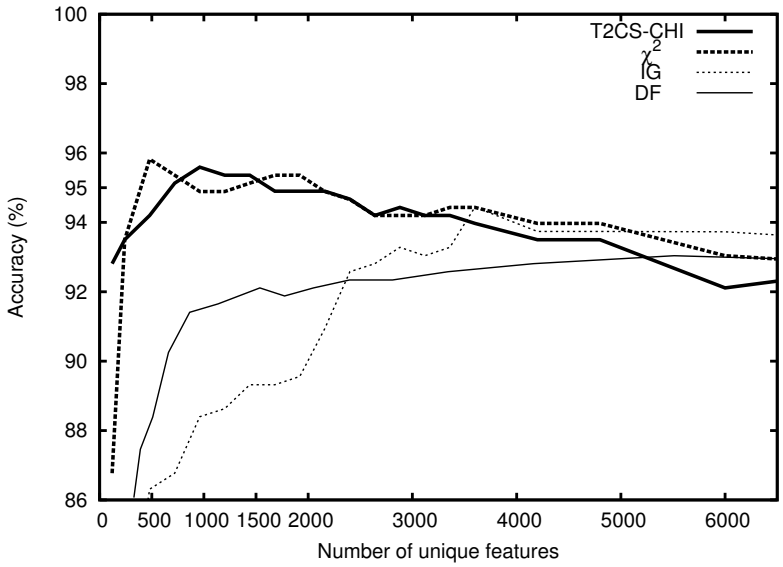

Figure 3.1: Comparison of FS techniques.

technique which simply counts, for each feature, the number of documents containing the feature in the train set and then select features whose number is higher than a given threshold. (We did not eliminate noisy words when selecting features with DF.) As shown in Figure 3.1, the DF technique has an accuracy of 92.6% at 3340 features or the equivalent DF threshold of 10, and it maintains accuracy around this value throughout higher FS levels. As a result, a large number of "rare" features can be removed without any loss of performance, as the total number of features is around 39100.

3.5 Chapter Summary

In this chapter, we study the problem of categorizing structured web sources by using their search interfaces. Our approach employs a filtering feature selection (FS) technique together with a multi-class Support Vector Machine categorizer. In our research, we use only the text contents of the search interfaces. We conducted experiments with our FS technique which includes a new metric and selection scheme, as well with existing FS methods. The experimental results indicate that: (a) aggressive feature selection improves classification performance significantly; (b) our classification approach and FS technique are effective. Our research also points out that rare words are not important to the categorization task.

Chapter 4

Multi-Pair: A Muti-label Method Making Use of Category Structure

In this chapter, we address the task of developing a multi-label classification algorithm. As mentioned in Section 1.1, we focus on improving the One-Vs-All approach, or the OVA approach for short, that is commonly used in researches, for instance, Joachims [1998], Yang [2001], Bekkerman et al. [2003], Lewis et al. [2004], Fan and Lin [2007]. This approach is also known as One-Vs-Rest. Let us consider a multi-label problem of m overlapping categories like *Politics*, *Health*, *Business*, *Science*, *Entertainment*, *Sports* and so on. The OVA approach transforms the problem into m independent binary classifiers, abbreviated as BCs, between each category and the rest (the category's complement) labeled as positive and negative respectively. A new article will be labeled as *Politics*, *Health* and *Business* if it is given positive scores by the BCs of these three categories, and negative scores by the other $(m-3)$ classifiers. The OVA approach has some drawbacks that lead to different research directions.

The first drawback is that the OVA approach does not take into account the correlation among categories, as its BCs run independently from one another. However if an article is already labeled as *Entertainment*, the chance that it in fact belongs additionally to *Sports* is higher than the chance that it belongs additionally to *Science*. Various methods have been proposed so as to make use of such correlation, for instance, in Godbole and Sarawagi [2004], Ghamrawi and McCallum [2005], Zhu et al. [2005], Tsoumakas and Vlahavas [2007], Tang et al. [2009]. Many among these authors, such as Godbole and Sarawagi, Tang et al., use an OVA technique as their first phase, and have a second phase that uses the correlation to consolidate the results output by the first phase.

The second OVA weakness is the imbalance of training data. As a kind of Zip's law for category size, there are often few large and many more small categories [Dumais

et al., 1998]. However, the OVA approach makes training data even more skewed as it increases the ratio between negative and fewer positive samples in its BCs $(m-1)$ times on average, where m is the number of categories. Since learning algorithms are normally optimized for accuracy, the increase of data skewness causes more samples to be assigned to the majority parts of the negative samples. Among multi-label studies, thresholding strategies [Yang, 2001] indirectly address this problem. They usually allow more samples to be labeled as positive by lowering categorizers' default thresholds [Fan and Lin, 2007].

The main focus of this chapter is the third issue, which to our knowledge has not been studied elsewhere in prior works. That is, when merging samples to create the BC of a category, the OVA approach ignores the fact that the category's complement is composed of around $(m-1)$ other categories. To make use of this existing category structure, intuitively, we reformulate a question, for example, "Does an article belong to the *Entertainment* category or not?". We ask more specific questions, like "Does the article more likely belong to *Entertainment* than to *Science*?", "Does the article more likely belong to *Entertainment* than to *Sports*?" and so on. Technically, we decompose each of OVA binary classification tasks further into multiple smaller and easier pair comparisons between a category and a subset of its complement. This decomposition also reduces the imbalance of training data created by the OVA approach directly. We name the new method Multi-Pair, or MP for short.

In this chapter, we also combine SCutFBR.1, which was reported one of the most effective thresholding strategies [Lewis et al., 2004], with the MP method. This is because, on the one hand, we observed that for very small categories (for instance, around five samples), the OVA and MP approach performed similarly. On the other hand, it was reported that thresholding strategies helped the OVA approach to increase macro-F_1 (defined in Section 2.4) for various datasets [Fan and Lin, 2007]. Via this combination, our goal thus is to improve the macro-F_1 performance of the MP method. This combination also indicates that the MP method can be considered complementary to other multi-label approaches that have an OVA technique in their cores.

As mentioned in Section 1.1, we experimentally evaluate our new approach by using three benchmark text collections RCV1-V2, 20 Newsgroups and Reuters-21578. For each collection, we use the preprocessed dataset made available by prior works and their results as baseline. In our experiments, the MP method outperforms the OVA approach in both cases with and without SCutFBR.1.

The organization of this chapter is similar to that of the previous chapter. In the next section we review related work. Then we present the MP method in Section 4.2, and combine it with ScutFBR.1 in Section 4.3. Experimental setup and classification results of the OVA and MP methods are shown in Section 4.4 and 4.5 respectively. Finally, in Section 4.6, we conclude with a summary.

This chapter is based on the paper Le and Conrad [2010b].

4.1 Related Work

In this section, we relate our work to other studies on multi-label, as well as multi-class single-label, classification.

Firstly, there are many multi-label approaches, which can be classified into two groups: (a) problem transformation (PT), and (b) algorithm adaptation (AA) [Tsoumakas and Katakis, 2007]. AA techniques, for example Adaboost.MH [Schapire and Singer, 2000] or CML [Ghamrawi and McCallum, 2005], are the extensions of single-label algorithms. They also normally involve some kinds of data transformation. PT methods, for instance OVA [Joachims, 1998, Yang, 2001, Bekkerman et al., 2003, Lewis et al., 2004, Fan and Lin, 2007] or RAKEL [Tsoumakas and Vlahavas, 2007], transform a multi-label problem into one or more single-label tasks, then solve them by a single-label classification method such as Support Vector Machines [Cortes and Vapnik, 1995] or Gaussian processes [Rasmussen and Williams, 2006]. OVA is the most commonly used PT approach. Our MP method is an extension of the OVA approach (see Section 4.2.2), and hence a new PT method.

As discussed in the begining of this chapter, on the one hand, the major part of existing approaches, AA or PT, aim at exploiting correlation among categories [Godbole and Sarawagi, 2004, Ghamrawi and McCallum, 2005, Zhu et al., 2005, Tsoumakas and Vlahavas, 2007, Tang et al., 2009]. On the other hand, our goal is to make use of the boundaries among categories to improve the OVA approach. In this regard, our approach complements the existing ones.

Secondly, multi-label classification is closely related to multi-class single-label categorization. The multi-label OVA approach has its single-label counterpart with the same name [see Rifkin and Klautau, 2004]. Likewise, the MP method has its sibling the Decision Directed Acyclic Graph method [Platt et al., 2000], as discussed in Section 4.2.3.

4.2 Multi-Pair Method

This section studies the Multi-Pair (MP) method: its (a) main algorithm, (b) partition schema, (c) multi-class single-label counterpart and (d) a feature selection technique for it. But first of all, we describe the One-Vs-All (OVA) approach in the next paragraph, for it is the starting point of our work.

Let us consider a multi-label classification problem of m categories A, B, C, ... in which data samples can be assigned to one or more categories, or in other words, the categories can overlap. Let $\overline{A}$ denote the set of samples not belonging to A; and $B\backslash A$ denote the set of samples that belong to a category B but not A. Given a base binary categorizer, the OVA approach classifies a new data sample $x*$ of the multi-label problem as follows:

1. For each category, say A, construct a binary classifier (BC), which decides if $x*$ belongs to A or not, in the following steps: create a classification task A-vs-$\overline{A}$ between A labeled as positive and the complement $\overline{A}$ labeled as negative; train the base categorizer for the binary task; and then classify $x*$.
2. The labels of $x*$ are the labels of categories into which $x*$ is classified by the BCs of these categories.

4.2.1 Main Algorithm

A naive observation is that for a category A, its complement $\overline{A}$ is composed of multiple categories; but the OVA approach ignores this category structure. Meanwhile, the MP method takes it into account by decomposing $\overline{A}$. A straightforward partition schema is that categories are taken out one by one in some order: $B_A = B \setminus A$, $C_A = (C \setminus A) \setminus B_A$ and so on. After that we do multiple pair comparisons A-vs-B_A, A-vs-C_A, ...so as to decide whether a sample $x*$ belongs to A. Comparing to the OVA approach, the MP method has two advantages as follows.

First, a MP pair comparison, say A-vs-B_A, is usually smaller and easier than its respective OVA binary classification A-vs-$\overline{A}$. This is because B_A is a subset of $\overline{A}$. The gap between A and B_A thus is larger than or at least equal to the gap between A and $\overline{A}$; or it is easier to separate A and B_A than A and $\overline{A}$. So we pursuit a "divide and conquer" strategy in the MP method in order to improve the OVA approach.

Second, the training data of a MP pair A-vs-B_A is more balanced than the training data of a respective OVA task A-vs-$\overline{A}$; and this is a good property of the MP method,

for the imbalance of data poses problems to categorizers [Chawla et al., 2002]. In the OVA approach, the ratio of positive and negative samples in A-vs-$\overline{A}$ is around $1/(m-1)$ on average, where m is the number of categories. The situation in fact is worse since there are often few large and many more small categories in a dataset [Dumais et al., 1998]. Via the decomposition, the MP method increases the ratio in A-vs-B_A k times (that is, to $k/(m-1)$) on average, where k is the number of the partitions formed by a MP partition schema. When all categories have the same size and $k = (m-1)$ (that is, each category has a corresponding partition formed), the ratio is about 1, or the training data of each MP pair is balanced.

Algorithm. Given the base binary categorizer, the MP method classifies a new data sample $x*$ of the multi-label classification problem as follows:

1. For each category, say A, construct a multi-pair classifier (MPC), which decides if $x*$ belongs to A or not, in the following steps:
   a) Partition the complement $\overline{A}$ into k subsets B_A, C_A, ... (see Section 4.2.2 for a partition schema).
   b) For each subset, say B_A, create a paired (or binary) classification task A-vs-B_A between A labeled as positive and B_A labeled as negative; train the base categorizer for the pair; and then classify $x*$.
   c) $x*$ is assigned to A if it is labeled as positive by all the k pair categorizers.
2. The labels of $x*$ are the labels of categories to which $x*$ is assigned by the MPCs of these categories.

So there are m MPCs in a MP instance; and they perform the same function as the m BCs of an OVA instance do. In Algorithm 1, there is the pseudo-code of the MP method, with a smallest-first partition schema described in the next section.

4.2.2 Partition Schema

A partition schema for $\overline{A}$, the complement of a category A, is an integral and important component of the MP method. When designing our schema, there are two considerations as follows.

First, a partition should be within the boundary of a category or the category itself. It is a simple constraint to make use of the existing category structure.

Input:
$\mathcal{X} = \{A, B, C, \ldots\}$: set of overlapping categories
$\widetilde{s}$: minimum partition size
`Train`, `Predict`: two imported functions of a base binary categorizer
$x*$: a new data sample
Output:
$L*$: category labels of $x*$

```
// Main algorithm
begin
    L* ← ∅
    for each U in 𝒳 do
        assigned ← True
        𝒫 ← Partition(U, 𝒳, s̃)
        for each V in 𝒫 do
            model ← Train(U as positive, V as negative)
            score ← Predict(model, x*)
            if score < 0 then
                assigned ← False
                exit for
        if assigned = True then
            L* ← Append LabelOf(U) into L*
end
// Smallest-first partition schema
function Partition(U, 𝒳, s̃):
    𝒫 ← ∅                                   // set of partitions returned
    V ← U                                   // category taken out at each time
    𝒴 ← 𝒳                                   // set of remaining categories
    repeat
        𝒴 ← Remove V from 𝒴
        for each W in 𝒴 do  W ← W \ V
        V ← ∅
        for each W in 𝒴 do
            if SizeOf(W) > s̃ then
                if (V = ∅) or (SizeOf(W) < SizeOf(V)) then
                    V ← W
        if V ≠ ∅ then  𝒫 ← Append V into 𝒫
    until V = ∅
    for each W in 𝒴 do  V ← V ∪ W    // last partition of samples left
    if V ≠ ∅ then  𝒫 ← Append V into 𝒫
return 𝒫
```

Algorithm 1: Multi-Pair method.

Second, a partition should be large enough, or in other words, its size should be bigger than a given value $\widetilde{s}$. Experimentally, without this key point, the MP will not perform well in a dataset with many small categories. Intuitively, it is hard, for either a person or a computer algorithm, to draw out a pattern from a few samples. Therefore, a pair categorizer A-vs-B_A, whose part B_A is small, often makes noisy prediction and should not be created. Likewise, in the case of a very small A, neither its $\overline{A}$ should be divided further.

Schema. When constructing the MPC of a category A, the partition schema for $\overline{A}$ given a minimum size $\widetilde{s}$ is as follows:

1. Take out categories, together with their samples, one by one from $\overline{A}$ to form respective partitions: $B_A = B \setminus A$, $C_A = (C \setminus A) \setminus B_A$ and so on; at each time, consider only the categories with the numbers of samples left bigger than $\widetilde{s}$, and choose the smallest one among them.
2. When there are no categories bigger than $\widetilde{s}$, use all the samples left to form the last partition.

In the above partition schema, instead of the smallest category we can choose the biggest one at each time. However, with the same $\widetilde{s}$, comparing to the biggest-first schema, the smallest-first schema forms more partitions. Its largest partitions are smaller and normally have fewer samples that belong additionally to other categories. Therefore, the smallest-first schema is more in line with the MP's "divide and conquer" strategy.

Now we consider the computational requirement of an OVA instance and a MP instance with a given $\widetilde{s}$. As there are m BCs like A-vs-$\overline{A}$, the total number of samples trained by the OVA instance is $(m \times N)$, where N is the number of all samples. Suppose for each category A, its MPC of the MP instance divides $\overline{A}$ into k partitions; so the number of training samples is $((k-1) \times N_A + N)$, where N_A is the size of A. For all m categories, the total number is around $((k-1) \times N + m \times N)$. (It is approximated for the categories overlap.) Since $k \leq (m-1)$, the total number of training samples of the MP instance is fewer than twice that of the OVA instance.

Lastly, with the introduction of the minimum size $\widetilde{s}$, we have to determine it, commonly through a cross-validation procedure. On the other hand, when setting $\widetilde{s}$ to a value bigger than the number of all samples, only one partition is formed by the MP schema, or we end up with an OVA instance. So an OVA instance is a special MP instance and can be used as an option during tuning $\widetilde{s}$. In practice, we should include this option so that it could be selected for very small categories. The reason is that the complement of a very small category should normally be kept undivided, as discussed above.

4.2.3 Multi-class Single-label Counterpart

There is not only the OVA approach for multi-label classification but also for multi-class single-label categorization [see Rifkin and Klautau, 2004]. Similarly, the single-label sibling of the MP method is the Decision Directed Acyclic Graph (DAG) method [Platt et al., 2000]. That connection is shown as below.

Suppose for a category A, its MPC of a MP instance divides the category's complement into k partitions. At this point we have a multi-class single-label problem of $(k + 1)$ categories, which is special for we are only interested in one question: if a sample belongs to A or not. This problem can be solved by a normal multi-class single-label method such as OVA, DAG, Error-correcting output codes [Dietterich and Bakiri, 1995] or multi-class SVM by Crammer and Singer [2001].

By using the DAG approach and placing A along the right edges of its decision graph [see Platt et al., 2000], we arrive at the MP method. That is, the classifying procedure used in a MPC is a stripped-down DAG algorithm. This stripped-down algorithm helps to reduce runtime. At most k pair comparisons between A and another category (or partition) are needed before we are able to decide if a sample belongs to A. Meanwhile, by employing the multi-class single-label OVA method, we always have to run $(k + 1)$ binary categorizers between a category and the rest composed of k categories.

4.2.4 Feature Selection for MP

As discussed in Section 4.2.3, in a MPC, there is a multi-class single-label classification problem of $(k+1)$ categories, where k is the number of partitions formed by the MP partition schema. So we can apply a multi-class FS technique to it. We select T2CS-CHI (see Section 3.3.2) instead of other multi-class FS techniques (see Sections 3.3.1 and 3.4.1) and a straightforward method in which a binary FS technique [see Sebastiani, 2002] is applied to each pair comparison of a MPC. This is because the T2CS-CHI technique gave the best overall performance (that is, the multi-label classification performance calculated from all categories) in several preliminary experiments we conducted.

4.3 Thresholding Strategy for Multi-Pair

In this section, we first describe the SCutFBR.1 thresholding strategy for the OVA approach, and refer to it as the SCutFBR.1 for OVA. Then, we extend the SCutFBR.1

strategy for the MP method, that is, the SCutFBR.1 for MP. In our description, we will use the performance measures F_1 and accuracy for a category, and macro-F_1 and micro-F_1 for all categories, which are defined in Section 2.4.

4.3.1 SCutFBR.1 for OVA

A binary classifying algorithm basically works in two steps. At first it computes a score for each data sample. Afterward it labels a sample as positive if the sample's score is higher than a predefined threshold, or negative otherwise. Changing the threshold thus affects the final outcome. For the OVA approach, a thresholding strategy such as PCut or SCut adjusts the default threshold of the base categorizer used in order to obtain better classification performance [Yang, 2001]. ScutFBR.1 is a SCut variant and was reported as one of the most effective strategies [Lewis et al., 2004]. Following Fan and Lin [2007], we now describe the ScutFBR.1 for OVA.

SCutFBR.1 works locally within the BC of each category of an OVA instance. As the true labels of testing samples are not available, SCutFBR.1 finds the new threshold of a BC through a cross-validation procedure. That is, available data are split into f parts; so we have a f-fold cross-validation schema. To optimize threshold for a fold:

1. $(f-1)$ parts are used to train the base categorizer used by the OVA instance.
2. The other part is used as a validation set, which supposedly contains l samples. After sorting the predicted scores of these l samples in descending order, $(l+1)$ thresholding levels are defined as follows: (a) the highest level is slightly higher than the biggest score; (b) $(l-1)$ levels are the average values of two adjacent scores; (c) the lowest level is slightly lower than the smallest scores. SCutFBR.1 selects the level that gives the best evaluation measure (such as F_1) on the validation set as the new threshold.
3. When the F_1 of the validation set does not achieve a given value called fbr, SCutFBR.1 sets the threshold of the fold to the highest level. This FBR heuristic is to address the data overfitting problem [see Yang, 2001].

Finally, the new threshold of the BC is the average value of the thresholds of all f folds.

To optimize for the global macro-F_1 measure, we select the thresholding level that gives the best F_1 for an individual category; and for the global micro-F_1 measure, the level

with the highest accuracy. [1] In the case of two levels with the same F_1 (or accuracy), we choose the lower one as its macro-F_1 (or micro-F_1) was usually a little better in our experiments. In addition, as pointed out by Fan and Lin, the FBR heuristic often lowers the thresholds of the BCs of very small categories; hence it increases the recalls of the categories and the macro-F_1 of all categories. Therefore, the new thresholds adjusted by re-optimizing for accuracy with SCutFBR.1 give a higher macro-F_1 than the default threshold of the base categorizer does.

4.3.2 SCutFBR.1 for MP

We apply ScutFBR.1 to each MPC of a MP instance, and use a cross-validation procedure to find a new threshold; that is, a technique similar to the SCutFBR.1 for OVA described in Section 4.3.1.

Suppose for a category A, its MPC consists of k pairs: A-vs-B_A, A-vs-C_A, ... So a MPC threshold is indeed an array of k thresholds, one for each pair. In this regard, an evaluation measure is a function of multiple variables needed to be optimized:

$$\max_{T_1, T_2, \ ..., T_k} measure(T_1, T_2, \ \ldots, T_k) \ ,$$

where k is the number of pairs, and T_i $(1 \leq i \leq k)$ is the threshold of the i-th pair.

We now present an optimizing algorithm through the following steps: (a) removing irrelevant samples, (b) defining thresholding levels and (c) searching in cycles.

First, when finding a threshold within a pair, we use only samples relevant to the pair. During training, the first pair A-vs-B_A uses only the samples of the category A and partition B_A. However, in validation, it shares the same validation set with the other pairs. Therefore, we discard samples that do not belong A or B_A from the validation set. This is because we consider the scores of these samples predicted by the pair A-vs-B_A are not reliable; and if we make use of them, there is likely a data overfitting problem. Let l_1 denote the size of the new smaller validation set of the first pair. Similarly for another i-th pair, the size of its new validation set is l_i, where $2 \leq i \leq k$. The decision procedure for the category A has to be changed accordingly. After the removal step, a sample is assigned to A if it is labeled as positive by all pairs whose validation sets contain the sample. That is, for a sample in fact belongs to A, the procedure is unchanged: the

[1] There is a more complicated threshold selection procedure for the micro-F_1 measure used by Fan and Lin; but their reported results are not higher than ours (see Section 4.5.1).

sample is placed (correctly) into A if it is predicted as positive by all pairs; but for a sample of $\overline{A}$, it is labeled (falsely) A if it is assigned to A by only the pair whose validation set contains the sample. (Note that we are in a cross-validation procedure using a training set; hence the true labels of all validating samples are known.)

Second, for a i-th pair ($1 \leq i \leq k$) whose validation set contains l_i samples, we define $(l_i + 1)$ thresholding levels in the same manner with the SCutFBR.1 for OVA. That is, after sorting the scores of the l_i samples, $(l_i - 1)$ levels are the average values of two adjacent scores; the highest and lowest levels are slightly higher and lower than the biggest and smallest scores respectively.

Third, in the optimization of the function $measure(T_1, T_2, \ldots, T_k)$, a method that scans all possible combinations of the thresholding levels of the k pairs is not feasible. A reasonable solution is that we sequentially adjust one T_i ($1 \leq i \leq k$) while fixing the others. A search round begins with T_1 and ends with T_k. We cycle over rounds until no improvement is made or the number of rounds is bigger than a predefined limit. The FBR heuristic is also used at the end: we reset the thresholds of pairs found to their highest levels if the F_1, which is calculated on the validation set of the category A, is smaller than a given fbr value.

We initialize searching by setting the thresholds of all pairs to their lowest thresholding levels. During adjusting a threshold T_i, if there are two levels with the same performance, we select the smaller one. The reason is that lower thresholds leave more options for subsequent adjustments. (Note that a sample is labeled A if it is predicted as positive by all pairs whose validation sets contain the sample.) In contrast, if we start searching with the highest levels of all pairs and select the bigger value in the case of two levels performing the same, we always end up with the result: all validating samples are categorized as not belonging to A.

As with the SCutFBR.1 for OVA, we optimize for macro-F_1 or micro-F_1 by tuning a MPC threshold for F_1 or accuracy respectively in each category. In addition, in the case of $k = 1$, the SCutFBR.1 for MP is similar to the SCutFBR.1 for OVA.

Algorithm 2 shows the pseudo-code of the SCutFBR.1 for MP. We note that in our experiments, the final numbers of search rounds were not bigger than two. We also observed that the order of pairs did not affect the result of the algorithm. [2]

[2] We conducted experiments in three options: pairs sorted by their size in ascending order, in descending order, and not ordered by size; and we obtained the same result.

Input:
$\mathcal{X} = \{A, B, C, \ldots\}$: set of overlapping categories;
fbr: lower bound of F-measure
Output:
One MPC threshold for each category

// Main program
for each *category* X **in** $\mathcal{X}$ **do**
  Construct a MPC of k pairs X-vs-P_i $(1 \leq i \leq k)$;
  Split each of X and P_i $(1 \leq i \leq k)$ into f folds;
  for *fold* $j = 1$ **to** f **do**
    Train a MP categorizer on other $(f-1)$ folds of data;
    $[S_1^j, \ldots, S_k^j] \leftarrow$ Predict samples' scores of the j-th fold by the MP categorizer;
    $[T_1^j, \ldots, T_k^j] \leftarrow$ `FindMPCThreshold(`fbr, k, $[S_1^j, \ldots, S_k^j]$`)`;
  Train the MPC by the MP categorizer; Set the threshold of the MPC to the average of f folds:
  $$[T_1, \ldots, T_k] = [\tfrac{1}{f}\sum_{j=1}^{f} T_1^j, \ldots, \tfrac{1}{f}\sum_{j=1}^{f} T_k^j];$$

// SCutFBR.1 within a fold
function `FindMPCThreshold(`fbr, k, $[S_1^j, \ldots, S_k^j]$`)`:
  for *pair* $i = 1$ **to** k **do**
    Select samples belonging to X or P_i from S_i^j;
    $L_i \leftarrow$ Compute thresholding levels from the scores of the selected samples;
    $T_i^j \leftarrow$ Lowest level of L_i; `// initialize pair threshold`
  $m_M \leftarrow -1$;
  for *round* $r = 1$ **to** $\widetilde{R}$ **do** `//` $\widetilde{R}$`: predefined limit of rounds`
    for *pair* $i = 1$ **to** k **do**
      $m_p \leftarrow -1$;
      for *level* $T =$ *lowest* **to** *highest level of* L_i **do**
        $m \leftarrow$ `Measure(`$T_1^j, \ldots, T_{i-1}^j, T, T_{i+1}^j, \ldots, T_k^j$`)`;
        if $m_p < m$ **then**
          $m_p \leftarrow m$;
          $T_i^j \leftarrow T$;
    if $m_M < m_p$ **then** $m_M \leftarrow m_p$;
    else exit for;
  if `Fmeasure(`$T_1^j, \ldots, T_k^j$`)` $< fbr$ **then** `// FBR heuristic`
    for *pair* $i = 1$ **to** k **do** $T_i^j \leftarrow$ Highest level of L_i;
return $[T_1^j, \ldots, T_k^j]$

Algorithm 2: SCutFBR.1 for Multi-Pair.

4.4 Experimental Setup

In this section, we describe the experimental setup used, including: datasets and their results reported by other authors, base categorizers and parameter tuning.

4.4.1 Multi-label Datasets and Previous Results

The three benchmark datasets that we use are RCV1-V2 subsets, 20 Newsgroups and Reuters-21578. For each of them, we briefly describe their data contents, previous OVA results obtained by other authors that we use as baseline, data representation and feature selection possibly used, as well as the highest results from prior works for these datasets that we collected.

RCV1-V2 Subsets

The RCV1-V2 corpus contains newswire stories from Reuters Ltd; and it was produced from the original Reuters Corpus Volume I after necessary corrections [Lewis et al., 2004]. We use the five preprocessed subsets of RCV1-V2 publicly available at LIBSVM website. [3] We will refer to these five subsets as RCV1. In each RCV1 subset, there are 3000 samples for training and 3000 samples for testing in 101 categories. This dataset is highly imbalanced. In each training set, the largest category has around 1400 samples while each of the 15 smallest categories contains no more than five samples.

RCV1 was used in Fan and Lin [2007] and Tang et al. [2009]. The former research studies on thresholding strategies with the OVA approach. The latter work proposes a multi-label method named MetaLabeler. We use the higher results of the former research as baseline; that is, 74.1% micro-F_1 and 34.1% macro-F_1 obtained with the OVA approach, 77.3% micro-F_1 and 49.0% macro-F_1 with the SCutFBR.1 for OVA optimized for micro-F_1, and 76.2% micro-F_1 and 50.6% macro-F_1 with the SCutFBR.1 for OVA optimized for macro-F_1. The 77.3% micro-F_1 and 50.6% macro-F_1 are also the highest results achieved by one method on this dataset.

All samples of RCV1 were already vectorized and normalized. Therefore, we do not apply feature selection to this dataset. We only remove unused features that appear in a testing set but not in a corresponding training set, and then re-normalize the testing vectors. Since there are five subsets, we will report the average performance on them.

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

20 Newsgroups

This corpus, collected by Ken Lang, contains around 20,000 articles of 20 Usenet discussion groups [Joachims, 1997]. Only about 4.5% of the articles were cross-posted among the groups. This corpus is balanced as each category has approximately 1,000 samples. We use the preprocessed data of the corpus in Bekkerman et al. [2003], provided freely online. [4] This corpus will be referred to as 20NG.

The results for 20NG as a multi-labeled dataset were reported in Bekkerman et al. [2003] and Gabrilovich and Markovitch [2004]. [5] Both researches are studies on feature selection using the OVA approach. Our baseline result is 85.6% micro-BEP reported in the former work because the result is for the same preprocessed dataset and with the bag-of-words representation we used. It is also a little better than the result of the latter research. The best result for 20NG is 88.6% micro-BEP, which was also obtained by Bekkerman et al. when using the Information Bottleneck word-cluster representation. There were no macro-average results reported for this dataset.

For 20NG, we use the common bag-of-words representation, weight terms by term frequency (TF), and normalize all document vectors. We do not apply feature selection to this dataset because it was the best strategy reported in the two prior works. As this dataset is already divided into four parts, we will report the average performance of a 4-fold cross-validation schema.

Reuters-21578

This is another corpus of documents gathered from Reuters newswire. [6] We use the preprocessed data of the corpus also in Bekkerman et al. [2003], provided freely online. [7] The preprocessed dataset is in the "ModApte" split with 7063 articles for training and 2742 articles for testing in 114 categories. This dataset is not only highly imbalanced but also considerably noisy. The first and second largest categories of the training set have 2709 and 1488 samples respectively. Meanwhile, there are 49 categories having no more than five articles, and the 19 smallest ones with only one articles. The two categories "Castor-oil" and "Castorseed" contain only one and the same sample.

[4] http://www.cs.technion.ac.il/~ronb/datasets/20NG.zip
[5] There are authors treating 20NG as a single-labeled dataset, for example Joachims [1997].
[6] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[7] http://www.cs.technion.ac.il/~ronb/datasets/Reuters.zip

A number of authors used the Reuters-21578 corpus in two forms: the 10 largest categories or the entire (or almost) dataset. We will indicate the 10 largest categories as Reuters–10–largest, and all categories as Reuters–all. Our baseline results with the OVA approach were reported in Bekkerman et al. [2003] – the source of the preprocessed dataset we use. For Reuters–10–largest, it is 92.3% micro-BEP obtained by the authors while replicating Dumais et al.'s experiment; and for Reuters–all, it is 87.0% micro-BEP achieved by Dumais et al. [1998] themselves. The 92.3% micro-BEP is also the best among previous Reuters–10–largest results [Gabrilovich and Markovitch, 2004, Zhu et al., 2005]. For Reuters–all, the highest results are 87.99% micro-F_1 and 87.99% micro-BEP with BandSVM and 87.96% micro-F_1 and 87.96% micro-BEP with SVM-HF – two two-phase multi-label methods which use an OVA technique as their first phase and were proposed in Godbole and Sarawagi [2004]. [8] Also, the OVA results reported in this paper are 87.15 micro-F_1 and 87.43 micro-BEP, or better than our baseline. There is another result significantly higher than the baseline; that is, 87.8% micro-BEP on the 95 largest categories, achieved by Weiss et al. [1999] by boosting (using 100 decision trees and an adaptive resampling scheme). There were no macro-average results reported for Reuters–10–largest or Reuters–all.

For Reuters–all (thus including Reuters–10–largest), we represent articles by the bag-of-words and term frequency schema, and normalize document vectors; that is, in the same manner for the 20NG dataset. Like the authors of the baseline results, we apply feature selection to Reuters–all. We use the T2CS-CHI technique after removing features appearing fewer than twice in every category. Since there is one subset in the "ModApte" split, we will simply report performance on the testing set.

4.4.2 Base Categorizer and Parameter Tuning

We implemented both the OVA and MP methods. For the prebuilt base binary categorizer, we use either LIBLINEAR [Fan et al., 2008] or LIBOCAS [Franc and Sonnenburg, 2009] – two of the current fastest linear Support Vector Machine (SVM) libraries. We employed the former during researching; and ran the latter additionally at last with all the same settings. [9] So unless stated explicitly otherwise, our results are obtained with LIBLINEAR.

[8] The micro-BEP values were computed by us from the micro-precisions and micro-recalls in this paper.

[9] We implemented the OVA and MP methods, as well as other processing modules, in Python programming language (www.python.org). They call either the LIBLINEAR or LIBOCAS linear binary categorizer, which is wrapped as a Python module by Cython language (www.cython.org).

Similarly to the task of categorizing structured web sources (see Section 3.2), we use a linear kernel with a bias of 1 (the software default value). In a dataset, we determine parameters for each category separately through a 10-fold cross-validation procedure on training sets.

For RCV1, there are two parameters: *svm-C* (trade-off between training error and margin) and $\widetilde{s}$ (minimum partition size). Following the studies of the baseline results, we determine these parameters by using accuracy. Within a category, we first tune $\widetilde{s}$ while setting *svm-C* to 1 (the software default value); and then determine *svm-C* with the found $\widetilde{s}$. At this point, we are able to obtain results with the MP (or OVA) method without a thresholding strategy. Finally, we go on to find a new threshold by using the SCutFBR.1 for MP (or OVA) with a predefined fbr value, optimized for either micro-F_1 or macro-F_1.

For 20NG, the tuning procedure is similar to the above procedure for RCV1. The only difference is the specified feasible set of $\widetilde{s}$, for all categories of 20NG are large (around $1,000$ samples).

For Reuters–all, there is one more parameter: fs-K (feature selection level). The other settings (*svm-C*, $\widetilde{s}$, fbr) are the same as those of RCV1. Since the T2CS-CHI technique makes use of category structure, its result is affected by the MP partition schema used. For now, we tune $\widetilde{s}$ and fs-K together in an exhaustive grid-search of pairs ($\widetilde{s}$, fs-K) while setting *svm-C* to 1. We then use the same procedure as for RCV1 to find *svm-C* and thresholds.

Lastly, in our MP implementation, we always included the option of an OVA instance (by setting $\widetilde{s}$ to a very big number) as suggested in Section 4.2.2. During tuning, in case of two parameters $\widetilde{s}$ with the same performance, we chose the bigger one. In this way, for very small categories, the option of an OVA instance could be selected since default choices were returned by the MP (and also OVA) cross-validation procedure most of the time (see Section 4.5.1). Specifically, the feasible set of $\widetilde{s}$ for RCV1, Reuters–all (including Reuters–10–largest) is $[25, 50, 100, 150, 200, 250, 300, 1000000]$; and for 20NG $[1, 1000000]$, where the biggest option 1000000 is equivalent to an OVA instance and the option 1 a partition schema of all categories (that is, each 20NG category has a corresponding partition formed).[10] They are the default parameters used in our experiments, unless stated explicitly otherwise.

[10] For other parameters, the feasible set of fs-K is $[1, 2, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]$ (percents); the feasible set of *svm-C* is $[2^{-2}, 2^{-1}, 2^0, 2^1, 2^2, 2^3]$; and $fbr = 10\%$. The set of *svm-C* and the fbr value were used by Fan and Lin.

4.5 Classification Results

In this section, we present the performance and comparison of the OVA and MP methods with or without SCutFBR.1, together with LIBLINEAR or LIBOCAS, in different experiments. We report the actual runtime in the experiments as well. Similarly to the studies of the baseline results [Fan and Lin, 2007, Bekkerman et al., 2003], we use macro-F_1, micro-F_1 and micro-BEP. We also use micro-P, micro-R in order to analyze the results in more detail. These performance measures are defined in Section 2.4. Briefly, micro-F_1, micro-BEP, micro-P and micro-R indicate micro-average F-measure, break even point, precision and recall respectively; and macro-F_1 indicates macro-average F-measure. In addition, source code to reproduce our classification results is available at http://dbs.cs.uni-duesseldorf.de/research/mp/ .

4.5.1 Main OVA and MP Results

Tables 4.1, 4.2 and 4.3 show the results for the three corpora RCV1-V2 (RCV1), 20 Newsgroups (20NG) and Reuters-21578 (Reuters–all and Reuters–10–largest) respectively, where "baseline" indicates the baseline OVA results reported in the prior works, whose preprocessed datasets we use (see Section 4.4.1); OVA and MP without a prefix "baseline" indicate our implementation of these two methods; OVA–SCut$_{mic}$ the SCutFBR.1 for OVA optimized for micro-F_1; and MP–SCut$_{MAC}$ the SCutFBR.1 for MP optimized for macro-F_1. All results in these tables are obtained with LIBLINEAR, and with the default smallest-first partition schema in the case of the MP method. Before any further discussion, it is observed that comparing to the baseline results, those of our OVA implementation are (a) similar over RCV1 (except for micro-F_1 without SCutFBR.1, ours is better), (b) much higher over 20NG, and (c) similar over Reuters–all and better over Reuters–10–largest. Therefore, from now on, we will only discuss the results within our own implementation.

As can be seen in the Tables 4.1, 4.2 and 4.3, without a thresholding strategy, the MP method outperforms the OVA approach on all the three corpora in micro-F_1, micro-BEP, micro-R and macro-F_1. In contrast, in micro-P, the OVA approach is better; and this reflects the tendency of placing more samples into the more populous (and negative) category, which is observed in the case of imbalanced data for (base) SVM categorizers [see Joachims, 2000]. In addition, for very small categories (for example, 20 smallest categories with no more than six samples in the first RCV1 subset), both the OVA and MP methods performed the same. This result can be explained as follows. The prebuilt

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
Baseline OVA	74.1				34.1
OVA	75.0	77.3	90.5	64.0	34.7
MP	78.5	79.2	86.4	71.9	40.3
Baseline OVA–SCut$_{mic}$	77.3				49.0
OVA–SCut$_{mic}$	77.2	77.4	81.4	73.5	49.7
MP–SCut$_{mic}$	78.9	78.9	81.4	76.5	52.8
Baseline OVA–SCut$_{MAC}$	76.2				50.6
OVA–SCut$_{MAC}$	76.2	76.2	75.5	76.9	51.2
MP–SCut$_{MAC}$	78.2	78.2	78.6	77.8	53.3

Table 4.1: Results for RCV1-V2 subsets (RCV1).

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
Baseline OVA		85.6			
OVA	88.7	89.1	94.8	83.4	88.6
MP	90.3	90.4	91.4	89.3	90.3
OVA–SCut$_{mic}$	89.8	89.8	91.7	87.9	89.7
MP–SCut$_{mic}$	90.4	90.4	92.6	88.3	90.3
OVA–SCut$_{MAC}$	89.8	89.8	91.0	88.6	89.8
MP–SCut$_{MAC}$	90.4	90.4	92.5	88.4	90.3

Table 4.2: Results for 20 Newsgroups (20NG).

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
All categories (Reuters–all)					
Baseline OVA		87.0			
OVA	86.9	87.1	91.9	82.3	38.1
MP	88.2	88.3	90.8	85.9	40.1
OVA–SCut$_{mic}$	85.8	85.8	86.7	84.9	46.6
MP–SCut$_{mic}$	86.9	86.9	86.1	87.6	48.5
OVA–SCut$_{MAC}$	85.4	85.4	85.2	85.6	47.5
MP–SCut$_{MAC}$	86.8	86.8	85.9	87.7	48.6
10 largest categories (Reuters–10–largest)					
Baseline OVA		92.3			
OVA	92.7	92.7	93.9	91.5	85.2
MP	93.3	93.3	92.8	93.7	87.2
OVA–SCut$_{mic}$	92.6	92.6	92.9	92.3	85.7
MP–SCut$_{mic}$	93.2	93.2	92.0	94.4	87.5
OVA–SCut$_{MAC}$	92.6	92.6	92.7	92.6	85.9
MP–SCut$_{MAC}$	93.2	93.2	92.0	94.4	87.5

Table 4.3: Results for Reuters-21578.

base SVM categorizer we used, as well as most of the others, is designed to maximize accuracy. When there are, for example, five positive and more than one thousand negative samples, the naive strategy that labels all samples as negative yields very high accuracy; but an effort to have only one true positive would likely result in several more false positives. Thus, in most of time, both the OVA and MP methods simply labeled all testing samples as negative, or in other words, as not belonging to the categories. So during parameter tuning, the $\widetilde{s}$ option of an OVA instance (that is, the biggest $\widetilde{s}$ value) was selected, and MP instances became OVA instances.

With the SCutFBR.1 thresholding strategy, the MP method also performs significantly better than the OVA approach on all the three corpora in micro-F_1, micro-BEP, micro-R and macro-F_1; but not as much as without SCutFBR.1. However this time, in micro-P, the MP's results are only poorer over Reuters–all when optimized for micro-F_1 and over the Reuters–10–largest; while in the other cases, they are similar or higher.

Adjusting threshold often increases the macro- and micro-average results of the OVA approach, and the macro-average results of the MP method. For RCV1 and Reuters–all – two datasets with small categories, SCutFBR.1 increases significantly macro-F_1 for both the OVA and MP methods. It also increases micro-F_1 over RCV1, but hurts micro-F_1 over Reuters–all for both the methods. For the other two datasets with large categories, SCutFBR.1 improves performance for the OVA approach on 20NG, but not on Reuters–10–largest. Meanwhile, the thresholding strategy does not help the MP method on both the datasets. This indicates that via the decomposition, the MP method makes training data more balanced; and thus adjusting threshold further becomes unnecessary for large categories.

Let us now compare the SCutFBR.1 for OVA with the MP method (without a thresholding strategy) through examining the micro-BEP measure. For the OVA approach, SCutFBR.1 increases micro-BEP over 20NG, keeps it similar over RCV1 when optimized for micro-F_1 and over Reuters–10–largest, and decreases it over RCV1 when optimized for macro-F_1 and over Reuters–all. Thus, the increase of micro-F_1 made by the SCutFBR.1 for OVA over RCV1 is only the result of balancing micro-P and micro-R. (Note that the BEP measure is the unweighted mean of precision and recall whereas the F-measure is the harmonic mean.) Meanwhile, the micro-BEP results of the MP method are significantly higher than those of the OVA approach on all datasets. The MP method also makes micro-P and micro-R more balanced. That is why between the MP method and the OVA approach, the difference in micro-F_1 is larger than the differ-

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
RCV1-V2 subsets (RCV1)					
MP	76.7	78.3	89.6	67.0	36.2
MP–SCut$_{mic}$	78.4	78.5	81.3	75.7	51.3
MP–SCut$_{MAC}$	77.6	77.6	77.4	77.8	52.3
20 Newsgroups (20NG)					
MP	90.3	90.3	91.5	89.1	90.3
MP–SCut$_{mic}$	90.3	90.4	92.5	88.2	90.3
MP–SCut$_{MAC}$	90.3	90.4	92.4	88.3	90.3
All categories of Reuters-21578 (Reuters–all)					
MP	88.0	88.1	91.4	84.8	40.8
MP–SCut$_{mic}$	87.0	87.0	86.8	87.2	47.7
MP–SCut$_{MAC}$	86.9	86.9	86.4	87.4	48.4
10 largest categories of Reuters-21578 (Reuters–10–largest)					
MP	93.3	93.3	93.5	93.1	87.0
MP–SCut$_{mic}$	93.3	93.3	92.3	94.3	87.8
MP–SCut$_{MAC}$	93.3	93.3	92.2	94.4	87.8

Table 4.4: Results of Multi-Pair with the biggest-first partition schema.

ence in micro-BEP. In addition, adjusting threshold for the MP method only decreases or keeps micro-BEP unchanged on all datasets.

Finally, the micro-BEP and micro-F_1 results obtained by the MP method alone and the macro-F_1 results with the SCutFBR.1 for MP (optimized for either micro-F_1 or macro-F_1) are higher than the previous results collected in Section 4.4.1 on all datasets.

4.5.2 MP Results with the Biggest-first Partition Schema

Table 4.4 shows the results of the MP method with the biggest-first partition schema (see Section 4.2.2), for RCV1, 20NG, Reuters–all and Reuters–10–largest, and with LIBLINEAR. Other notions used in this table are the same as those used in the previous Table 4.1. In the following paragraphs, we will consider the Table 4.4 together with the Tables 4.1, 4.2 and 4.3.

It can be observed that for 20NG, Reuters–all and Reuters–10–largest, the biggest-first schema performs similarly to the smallest-first schema in micro-F_1, micro-BEP and macro-F_1. For RCV1, before adjusting threshold, the performance of the biggest-first

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
RCV1-V2 subsets (RCV1)					
MP	77.7	78.7	87.9	69.6	37.8
MP–SCut$_{mic}$	78.7	78.7	80.9	76.6	52.1
MP–SCut$_{MAC}$	77.9	77.9	77.9	77.9	52.6
All categories of Reuters-21578 (Reuters–all)					
MP	87.8	87.9	90.8	85.0	41.0
MP–SCut$_{mic}$	86.4	86.4	86.4	86.4	48.3
MP–SCut$_{MAC}$	86.2	86.2	85.5	86.9	48.7
10 largest categories of Reuters-21578 (Reuters–10–largest)					
MP	92.8	92.8	92.9	92.7	86.4
MP–SCut$_{mic}$	92.8	92.8	92.3	93.2	86.4
MP–SCut$_{MAC}$	92.8	92.8	92.2	93.3	86.5

Table 4.5: Results of Multi-Pair with the two-option feasible set of $[150, 1000000]$ for $\widetilde{s}$, where the option 1000000 is equivalent to an OVA instance.

schema is considerably lower than that of the smallest-first one; and after adjusting, the former is still poorer but the gap is narrower.

As with the smallest-first schema, the MP method with the biggest-first schema outperforms the OVA approach on all datasets in both cases with and without SCutFBR.1.

4.5.3 MP Results with a Two-option Feasible Set for $\widetilde{s}$

We conducted an additional experiment whose two-option feasible set for $\widetilde{s}$ is $[s, 1000000]$, where s is a value in the range of $[25, 50, 100, 150, 200, 250, 300]$ and the 1000000 option is equivalent to an OVA instance. Table 4.5 shows the results of the MP method where $s = 150$, for RCV1, Reuters–all and Reuters–10–largest, with the default smallest-first partition schema and LIBLINEAR. Other notions used in this table are the same as those used in the previous Table 4.1. We do not report the results for 20NG in this table since they are the same as those of the Table 4.2 presented before. Also, comparing to the results of $s = 150$, those of $s = 25$ or $s = 50$ are lower; but for the others $s \geq 100$, the results are only a little higher or lower. So it is unnecessary to show them here. [11]

Comparing to the previous MP results of full $\widetilde{s}$ options shown in the Tables 4.1 and 4.3, the results of two-option $s = 150$ are lower. For micro-F_1 and micro-BEP, the differences

[11] The results of others $s \neq 150$ are available online together with the experiments' source code.

are from 0.2% to 0.8% and mostly around 0.5% over RCV1, Reuters–all and Reuters–10–largest. For macro-F_1, the two-option results are around 1.0% poorer over RCV1 and Reuters–10–largest, but similar over Reuters–all.

Comparing to the OVA results already presented in the Tables 4.1 and 4.3, the MP results of two-option $s = 150$ are significantly higher in micro-F_1, micro-BEP and macro-F_1 on all datasets most of the times; except for micro-F_1 and micro-BEP over Reuters–10–largest, the MP results are slightly better.

4.5.4 OVA and MP Results with LIBOCAS

Tables 4.6, 4.7 and 4.8 show the results of the OVA and MP methods obtained with LIBOCAS, together with the default smallest-first partition schema, for RCV1, 20NG, Reuters–all and Reuters–10–largest. Other notions used in the tables are the same as those used in the previous Table 4.1. In the following paragraphs, we will consider the Tables 4.6, 4.7 and 4.8 together with the Tables 4.1, 4.2 and 4.3.

As can be seen in these tables, the results obtained with LIBOCAS are very similar to those obtained with LIBLINEAR. Thus all the discussions on the OVA and MP methods with LIBLINEAR are also applied to LIBOCAS.

Meticulously, with the MP method, for RCV1 and Reuters–all – two datasets with small categories, LIBOCAS usually gives slightly better results (0.1% or so) than LIBLINEAR does, except for macro-F_1 over Reuters–all LIBOCAS around 0.8% better. For 20NG, both the SVM libraries perform almost the same with both the OVA and MP methods. For Reuters–10–largest, sometimes LIBOCAS is insignificantly higher while other times insignificantly poorer.

4.5.5 Computational Efforts

In this section, we report the actual runtime, training and testing altogether, in our experiments so as to give the coarse speed comparison between the OVA and MP methods. All experiments were conducted on a 2.4GHz 1.5GB RAM Pentium 4 operated by Debian GNU/Linux 5.0. Let MP_{full} indicate the MP instance of full $\widetilde{s}$ options, MP_{s150} the MP instance of two-option $s = 150$, and OVA_{ins} the OVA instance in our experimental settings.

We first consider the RCV1 dataset. When using LIBLINEAR, it took OVA_{ins} about 58 minutes, MP_{s150} 1 hour 44 minutes, and MP_{full} 4 hours 19 minutes; or OVA_{ins} was 1.8

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
OVA	75.0	77.3	90.6	64.0	34.7
MP	78.5	79.2	86.6	71.9	40.4
OVA–SCut$_{mic}$	77.4	77.6	81.6	73.6	50.0
MP–SCut$_{mic}$	79.0	79.0	81.4	76.7	53.0
OVA–SCut$_{MAC}$	76.3	76.3	75.7	77.0	51.3
MP–SCut$_{MAC}$	78.2	78.2	78.6	77.9	53.4

Table 4.6: Results for RCV1-V2 subsets (RCV1), with LIBOCAS.

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
OVA	88.7	89.1	94.7	83.5	88.7
MP	90.3	90.4	91.5	89.2	90.3
OVA–SCut$_{mic}$	89.8	89.9	91.7	88.0	89.7
MP–SCut$_{mic}$	90.4	90.4	92.5	88.3	90.3
OVA–SCut$_{MAC}$	89.8	89.8	90.9	88.6	89.8
MP–SCut$_{MAC}$	90.4	90.4	92.4	88.5	90.3

Table 4.7: Results for 20 Newsgroups (20NG), with LIBOCAS.

Method	*Micro-F_1*	*Micro-BEP*	*Micro-P*	*Micro-R*	*Macro-F_1*
All categories (Reuters–all)					
OVA	86.9	87.1	92.0	82.3	38.2
MP	88.3	88.4	91.3	85.5	41.0
OVA–SCut$_{mic}$	86.2	86.2	87.3	85.2	46.8
MP–SCut$_{mic}$	87.2	87.2	86.9	87.4	49.3
OVA–SCut$_{MAC}$	85.8	85.8	85.9	85.7	47.6
MP–SCut$_{MAC}$	86.9	86.9	86.3	87.6	49.3
10 largest categories (Reuters–10–largest)					
OVA	92.5	92.5	93.8	91.3	85.0
MP	93.3	93.3	93.1	93.5	87.1
OVA–SCut$_{mic}$	92.8	92.8	92.8	92.7	86.1
MP–SCut$_{mic}$	93.2	93.2	92.3	94.1	87.3
OVA–SCut$_{MAC}$	92.7	92.7	92.6	92.8	86.1
MP–SCut$_{MAC}$	93.2	93.2	92.2	94.2	87.3

Table 4.8: Results for Reuters-21578, with LIBOCAS.

times faster than MP_{s150}, and 4.5 times than MP_{full}. With LIBOCAS, it took OVA_{ins} around 5 hours 19 minutes, and MP_{full} 45 hours 19 minutes; or OVA_{ins} was 8.5 times faster than MP_{full}. Likewise, experiments with LIBOCAS ran 5.5 or 10.5 times longer than those with LIBLINEAR did. In Franc and Sonnenburg [2009], there was another report that LIBOCAS was slower than LIBLINEAR when using a *svm*-$C > 1$ (trade-off between training error and margin).

Let us now take the Reuters–all dataset as another example. When using LIBLINEAR, it took OVA_{ins} approximately 1 hours 40 minutes, MP_{s150} 3 hour 24 minutes, and MP_{full} 14 hours 20 minutes; or OVA_{ins} was 2.0 times quicker than MP_{s150}, and 8.6 times than MP_{full}. With LIBOCAS, it took OVA_{ins} about 2 hours 14 minutes, and MP_{full} 25 hours 50 minutes; or OVA_{ins} was 11.6 times quicker than MP_{full}. Experiments with LIBOCAS ran 1.3 or 1.8 times slower than those with LIBLINEAR did as well. Hence, in this dataset, LIBOCAS was also slower than LIBLINEAR, but not as much as in RCV1.

In brief, the OVA instances were at least 1.8 times faster than the MP instances; and the actual runtime depended on various factors, including datasets, the mode of parameter tuning, feature selection and base categorizers. Indeed, the fact that the MP method is slower than the OVA approach is unavoidable, as we switch from the view of binary (or two-class) classification to the view of multi-class classification (see Section 4.2.3).

4.6 Chapter Summary

In this chapter, we propose a multi-label classification method named Multi-Pair (MP). This new method is developed from the commonly used One-Vs-All (OVA) approach by making use of category structure. The MP method can be described in a common-sense language as follows. Instead of a general question (like A-vs-$\overline{A}$ in the OVA approach), we should ask more specific questions (like A-vs-B_A, A-vs-C_A,...) whenever possible. Also, an alternative question (say A-vs-B_A) should be meaningful (B_A is within the boundary of a category B) and not too specific (B_A is big enough). Furthermore, we incorporate the SCutFBR.1 thresholding strategy into the MP method. In our experimental evaluation, the MP method outperforms the OVA approach in both cases with and without SCutFBR.1.

Chapter 5

Conclusion

This thesis is a study on multi-class classification through two different research problems. The first problem is to organize structured sources on the Web. In this task, along with the classification approach using aggressive feature selection (FS), we propose T2CS-CHI – a new multi-class FS technique. The second task is to develop a multi-label classification algorithm. In this second task, based on the commonly used One-Vs-All (OVA) approach, we introduce a new method named Multi-Pair (MP). Then we go further to combine it with the SCutFBR.1 thresholding strategy. All our proposed methods are accompanied by experimental evaluation, whose results indicate that these methods are effective.

The aspect that makes our works different from prior researches is that we make use of category structure. In the existing FS techniques and OVA approach, a multi-class classification problem is transformed into binary classification problems between a category and its complement, that is, a two-class view. Meanwhile, we take into account the fact that the complement of a category is composed of multiple categories, that is, a multi-class view. We either do not merge the samples of the complement together (as with T2CS-CHI), or divide the complement into smaller partitions which is within a categories or a category itself (as with MP). Then we transform a multi-class problem into binary problems between a category and a partition of its complement. In this way, we aim at making use the existing boundary among categories. Our key observation is that to make use of this category structure effectively, a partition should be large enough, for it is difficult to draw out a pattern from a few data samples.

Like multi-class single-label classification is the extension of binary classification, our multi-class view can be considered as an extension of the two-class view. The OVA approach is the special case of the MP method, where the complement of each category is kept undivided by a MP partition schema. Likewise, the category-specific χ^2 metric

is defined for binary classification. The T2CS-CHI metric is for multi-class single-label classification; and in the case of binary classification, the T2CS-CHI metric is equivalent to the category-specific χ^2 metric.

For future work, we are interested in:

(a) evaluating the effectiveness of the T2CS-CHI technique in other multi-class classification problems,

(b) making the MP method faster,

(c) extending the MP method in order to use the correlation information among categories,

(d) making use of category structure in developing an algorithm, but this time, for multi-class single-label classification.

Bibliography

Own Publications

H. Q. Le and S. Conrad. Classifying structured web sources using aggressive feature selection. In *Proceedings of WebIST'09, 5th International Conference on Web Information Systems and Technologies*, 2009.

H. Q. Le and S. Conrad. Classifying structured web sources using Support Vector Machine and aggressive feature selection. In *Web Information Systems and Technologies (5th International Conference, WEBIST 2009, Revised Selected Papers)*, volume 45 of *Lecture Notes in Business Information Processing*, pages 270–282. Springer, 2010a.

H. Q. Le and S. Conrad. Multi-Pair: A method making use of category structure for multi-label classification. *The Journal of Machine Learning Research (submitted)*, 2010b.

Other References

R. A. Baeza-Yates and B. A. Ribeiro-Neto. *Modern Information Retrieval*. Addison-Wesley, 1999.

L. Barbosa and J. Freire. Searching for hidden-web databases. In *Proceedings of WebDB'05, 8th International Workshop on Web and Databases*, pages 1–6, 2005.

L. Barbosa, J. Freire, and A. Silva. Organizing hidden-web databases by clustering visible web documents. In *Proceedings of ICDE'07, 23rd International Conference on Data Engineering*, pages 326–335, 2007.

R. Bekkerman, R. El-Yaniv, N. Tishby, and Y. Winter. Distributional word clusters vs. words for text categorization. *Journal of Machine Learning Research*, 3:1183–1208, 2003.

K. P. Bennett and C. Campbell. Support vector machines: hype or hallelujah? *ACM SIGKDD Explorations*, 2:1–13, 2000.

M. K. Bergman. White paper - the Deep Web: Surfacing hidden value. Available at http://www.brightplanet.com/resource-library/white-papers/, 2001.

J. P. Callan, M. Connell, and A. Du. Automatic discovery of language models for text databases. *ACM SIGMOD Record*, 28:479–490, 1999.

S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. *ACM SIGMOD Record*, 27:307–318, 1998.

C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

K. C.-C. Chang, B. He, C. Li, M. Patel, and Z. Zhang. Structured databases on the Web: Observations and implications. *ACM SIGMOD Record*, 3:61–70, 2004.

K. C.-C. Chang, B. He, and Z. Zhang. Toward large scale integration: Building a MetaQuerier over databases on the Web. In *Proceedings of CIDR'05, 2nd Conference on Innovative Data Systems Research*, pages 44–55, 2005.

S. Chawathe, H. Garcia-molina, J. Hammer, K. Irel, Y. Papakonstantinou, J. Ullman, and J. Widom. The Tsimmis project: Integration of heterogeneous information sources. In *IPSJ Conference*, 1994.

N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16(1): 321–357, 2002.

C. Cortes and V. Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *The Journal of Machine Learning Research*, 2:265–292, 2001.

T. G. Dietterich and G. Bakiri. Solving multiclass learning problems via error-correcting output codes. *Journal of Artificial Intelligence Research*, 2:263–286, 1995.

S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of CIKM'98, 7th International Conference on Information and Knowledge Management*, pages 148–155, 1998.

R.-E. Fan and C.-J. Lin. A study on threshold selection for multi-label classification. Available at http://www.csie.ntu.edu.tw/∼cjlin/papers.html, 2007.

R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, , and C.-J. Lin. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research*, 9: 1871–1874, 2008. Software available at http://www.csie.ntu.edu.tw/∼cjlin/liblinear/.

G. Forman. An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, 3:1289–1305, 2003.

V. Franc and S. Sonnenburg. Optimized cutting plane algorithm for large-scale risk minimization. *The Journal of Machine Learning Research*, 10:2157–2192, 2009. Software available at http://cmp.felk.cvut.cz/∼xfrancv/ocas/html/.

E. Gabrilovich and S. Markovitch. Text categorization with many redundant features: using aggressive feature selection to make SVMs competitive with C4.5. In *Proceedings of ICML'04, 21st International Conference on Machine Learning*, 2004.

N. Ghamrawi and A. McCallum. Collective multi-label classification. In *Proceedings of CIKM'05, 14th International Conference on Information and Knowledge Management*, pages 195–200, 2005.

S. Godbole and S. Sarawagi. Discriminative methods for multi-labeled classification. In *Advances in Knowledge Discovery and Data Mining*, volume 3056/2004 of *Lecture Notes in Computer Science*, pages 22–30. Springer, 2004.

B. He and K. C.-C. Chang. Statistical schema matching across web query interfaces. In *Proceedings of COMAD'03, 10th International Conference on Management of Data*, pages 217–228, 2003.

B. He, T. Tao, and K. C.-C. Chang. Organizing structured web sources by query schemas: A clustering approach. In *Proceedings of CIKM'04, 13th Conference on Information and Knowledge Management*, pages 22–31, 2004.

H. He, W. Meng, C. Yu, and Z. Wu. WISE-Integrator: A system for extracting and integrating complex web search interfaces of the Deep Web. In *Proceedings of VLDB'05, 31st International Conference on Very Large Data Bases*, pages 1314–1317, 2005.

C.-W. Hsu, C.-C. Chang, and C.-J. Lin. *A practical guide to support vector classification*, 2009. Available at http://www.csie.ntu.edu.tw/∼cjlin/libsvm.

P. G. Ipeirotis, L. Gravano, and M. Sahami. Probe, count, and classify: categorizing hidden web databases. In *Proceedings of COMAD'01, 8th International Conference on Management of Data*, pages 67–78, 2001.

T. Joachims. Estimating the generalization performance of a SVM efficiently. In *Proceedings of ICML'00, 17th International Conference on Machine Learning*, 2000.

T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. In *Proceedings of ICML'97, 14th International Conference on Machine Learning*, 1997.

T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Machine Learning: ECML-98*, volume 1398/1998 of *Lecture Notes in Computer Science*, pages 137–142. Springer, 1998.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999. Software available from http://svmlight.joachims.org/.

A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of VLDB'96, 22nd International Conference on Very Large Data Bases*, pages 251–262, 1996.

D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. RCV1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

Y. Lu, H. He, Q. Peng, W. Meng, and C. Yu. Clustering e-commerce search engines based on their search interface pages using wise-cluster. *Data & Knowledge Engineering Journal*, 59:231–246, 2006.

D. Mladenic. Feature subset selection in text learning. In *Machine Learning: ECML-98*, volume 1398/1998 of *Lecture Notes in Computer Science*, pages 95–100. Springer, 1998.

J. C. Platt, N. Cristianini, and J. Shawe-taylor. Large margin DAGs for multiclass classification. In *Advances in Neural Information Processing Systems*, pages 547–553. MIT Press, 2000.

M. F. Porter. An algorithm for suffix stripping. In *Readings in information retrieval*, Morgan Kaufmann Multimedia Information And Systems, pages 313–316. Morgan Kaufmann, 1997. Software available at http://tartarus.org/∼martin/PorterStemmer/.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

R. Rifkin and A. Klautau. In defense of One-Vs-All classification. *Journal of Machine Learning Research*, 5:101–141, 2004.

M. Rogati and Y. Yang. High-performing feature selection for text classification. In *Proceedings of CIKM'02, 11th International Conference on Information and Knowledge Management*, 2002.

R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

F. Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47, 2002.

P. Soucy and G. W. Mineau. A simple feature selection method for text classification. In *Proceedings of 17th International Conference on Artificial Intelligence*, pages 897–902, 2001.

P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

L. Tang, S. Rajan, and V. K. Narayanan. Large scale multi-label classification via MetaLabeler. In *Proceedings of WWW2009, 18th International Conference on World Wide Web*, pages 211–220, 2009.

I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun. Support vector learning for interdependent and structured output spaces. In *Proceedings of ICML'04, 21st International Conference on Machine Learning*, pages 412–420, 2004. Software available from http://svmlight.joachims.org/.

G. Tsoumakas and I. Katakis. Multi-label classification: An overview. *Journal of Data Warehousing and Mining*, 2007:1–13, 2007.

G. Tsoumakas and I. Vlahavas. Random k-labelsets: An ensemble method for multilabel classification. In *Machine Learning: ECML 2007*, volume 4701/2007 of *Lecture Notes in Computer Science*, pages 406–417. Springer, 2007.

UIUC. The UIUC Web integration repository. Computer Science Dept., Uni. of Illinois at Urbana-Champaign. http://metaquerier.cs.uiuc.edu/repository, 2003.

S. M. Weiss, F. J. Damerau, D. E. Johnson, F. J. Oles, and T. Goetz. Maximizing text-mining performance. *IEEE Intelligent Systems*, 14:63–69, 1999.

W. Wu, C. Yu, A. Doan, and W. Meng. An interactive clustering-based approach to integrating source query interfaces on the Deep Web. In *Proceedings of COMAD'04, 11th International Conference on Management of Data*, 2004.

Y. Yang. A study of thresholding strategies for text categorization. In *Proceedings of SIGIR'01, 24th International Conference on Research and Development in Information Retrieval*, pages 137–145, 2001.

Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of ICML'97, 14th International Conference on Machine Learning*, pages 412–420, 1997.

O. Zamir and O. Etzioni. Web document clustering: a feasibility demonstration. In *Proceedings of ACM SIGIR'98, 21st International Conference on Research and Development in Information Retrieval*, pages 46–54, 1998.

S. Zhu, X. Ji, W. Xu, and Y. Gong. Multi-labelled classification using maximum entropy method. In *Proceedings of SIGIR'05, 28th International Conference on Research and Development in Information Retrieval*, pages 274–281, 2005.

Index

S

T

V

Die hier vorgelegte Dissertation habe ich eigenständig und ohne unerlaubte Hilfe angefertigt. Die Dissertation wurde in der vorgelegten oder in ähnlicher Form noch bei keiner anderen Institution eingereicht. Ich habe bisher keine erfolglosen Promotionsversuche unternommen.

Düsseldorf, den 01.03.2010

Hieu Quang Le