HIERARCHICAL CLASSIFICATION WITH RARE CATEGORIES AND INCONSISTENCIES

by

Azad Naik
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Computer Science

Committee:

________________________ Dr. Huzefa Rangwala, Dissertation Director

________________________ Dr. Carlotta Domeniconi, Committee Member

________________________ Dr. Jessica Lin, Committee Member

________________________ Dr. Hemant Purohit, Committee Member

________________________ Dr. Sanjeev Setia, Department Chair

________________________ Dr. Kenneth S. Ball, Dean, Volgenau School of Engineering

Date: ________________________ Spring Semester 2017
George Mason University
Fairfax, VA

Hierarchical Classification with Rare Categories and Inconsistencies

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Azad Naik
Master of Science
George Mason University, USA, 2013
Bachelor of Technology
Indian Institute of Technology (ISM) - Dhanbad, India, 2009

Director: Dr. Huzefa Rangwala, Professor
Department of Computer Science

Spring Semester 2017
George Mason University
Fairfax, VA

Dedication

This dissertation is dedicated to my beloved parents, Late Mr. Bhojeshwar Prasad Naik and Mrs. Shyama Naik, my loving wife, Mrs. Richa Parihar, and to my lovely younger sister, Akanksha. Without their love, support and encouragement this work would not have been possible.

Acknowledgments

I would like to thank my dissertation director, Dr. Huzefa Rangwala for the immense help throughout the project, advice given by Dr. Rangwala has been of great help in solving the challenging problem. I am grateful for his countless hours of reading, encouraging and patience throughout the entire process.

I would also like to extend my thanks to the committee member, Dr. Carlotta Domeniconi, Dr. Jessica Lin and Dr. Hemant Purohit for there valuable guidance and precious time in reviewing. I am also thankful to lab members for the fruitful discussion and suggestion during the course of my dissertation work.

Finally, I would like to thank my parents, wife, sister, relatives, friends and supporters who made this happen.

This research was supported by the National Science Foundation (NSF) Grant #1252318.

Table of Contents

List of Tables

List of Figures

List of Abbreviations

μF_1	Micro-F_1
kNN	k Nearest Neighbor
$rewHier$	Rewired Hierarchy
$rSVM$	Rank SVM
$rSVM_{CS}$	Cost Sensitive Rank SVM
$rSVM_{CS}^{HR}$	Hierarchy Regularized Cost Sensitive Rank SVM
$rSVM^{HR}$	Hierarchy Regularized Rank SVM
BLF	Bottom Level Flattened
DMOZ	Directory Mozilla
FC	Flat Classification
FS	Feature Selection
HC	Hierarchical Classification
IN	Internal Node
INF	Inconsistent Node Flattened
IPC	International Patent Classification
LCL	Local Classifier per Level
LCN	Local Classifier per Node
LCPN	Local Classifier per Parent Node
LN	Leaf Node
LR	Logistic Regression
MeSH	Medical Subject Headings
MF_1	Macro-F_1
MLF	Multiple Level Flattened
MTL	Multi-Task Learning
NC	Node Creation
ND	Node Deletion
NG	NewsGroup
PCRewire	Parent-Child Rewiring
SVM	Support Vector Machine
TD	Top-Down
TLF	Top Level Flattened

Abstract

HIERARCHICAL CLASSIFICATION WITH RARE CATEGORIES AND INCONSISTENCIES

Azad Naik, PhD

George Mason University, 2017

Dissertation Director: Dr. Huzefa Rangwala

Advancement in digital technology has generated a massive amount of data. Large amount of information streaming in from various sources such as phones, tablets, computers and internet has made an immense need to provide a structured and organized view of the data. Hierarchy (taxonomy) is one of the most easy and convenient way of data organization. It has been used extensively to store large volumes of data in various application domains ranging from biological datasets (for organizing genes and protein sequences) to image and text datasets (for providing the structured view of billions of images and web pages).

Hierarchical structure representation of the data can be effectively used to eliminate the expensive and tedious task of manual classification. To this end, Hierarchical Classification (HC) deals with the task of automatically classifying the instances (examples) within the topic hierarchy have been developed. Although, HC is popular among the researchers due to its wide application, it faces severe challenges due to the following reasons:

1. Data imbalance problem with several classes having very few positive instances for training (rare categories problem).

2. Incorporating hierarchical structural relationships into the model parameters optimization.

3. Multi-label classification were instances may belong to multiple classes.

4. Feature selection for effective and efficient discrimination between classes.

5. Inconsistent hierarchy due to manual design based on semantics and partial domain knowledge.

6. Non-mandatory leaf node predictions (or orphan nodes detection) were instances may be assigned to internal nodes in the hierarchy, and

7. Scalability due to large number of classes and instances with high-dimensional features.

Several approaches that address these issues individually (or multiple issues together) have been developed over the years.

In this thesis, we develop novel and innovative methods to deal with the rare categories and inconsistencies problems within the hierarchy classification framework. For rare categories, we present a ranking methodology that exploits the hierarchical structure for generalized model learning, whereas for dealing with hierarchical inconsistencies we propose an inconsistent node flattening and rewiring strategy. We have also developed the methods for embedding feature selection into the HC framework which helps to improve the classification accuracy while reducing the computational time during the learning and prediction phases. Additionally, the proposed approach also reduces the memory required to store the large number of model parameters. Finally, we develop a multi-stage integrated pipeline to solve the large-scale HC problem that can also detect orphan classes (*i.e.*, classes with no examples in the training set).

Chapter 1: Introduction

1.1 Motivation

Hierarchical structures are widely popular for organizing large volumes of data and have been successfully used in various large-scale online prediction competitions, such as Large Scale Hierarchical Text Classification (LSHTC[1]) challenge for representing web documents in web-taxonomy, ImageNet[2] for organizing images according to the WordNet hierarchy and BioASQ[3] challenge for biomedical semantic indexing of PubMed journal abstracts organized in the MeSH hierarchy.

Manual annotation of unlabeled instances (examples) into hierarchy of classes is a tedious and cumbersome task. This problem become even more difficult with the exponential data growth rate over time. Although, several traditional binary and multi-class classification techniques have been developed for automated classification, they are not efficient and effective for Hierarchical Classification (HC) because they ignore the implicit inter-class relationships information that are available from the hierarchical structures. To overcome this shortcoming various HC approaches have been proposed in the literature [3–9]. Still there are many research gaps that need be addressed with more theoretical foundations and experimental studies. In the next few paragraphs we will discuss and highlight some of the critical challenges faced by large-scale HC that form the main focus point of this thesis.

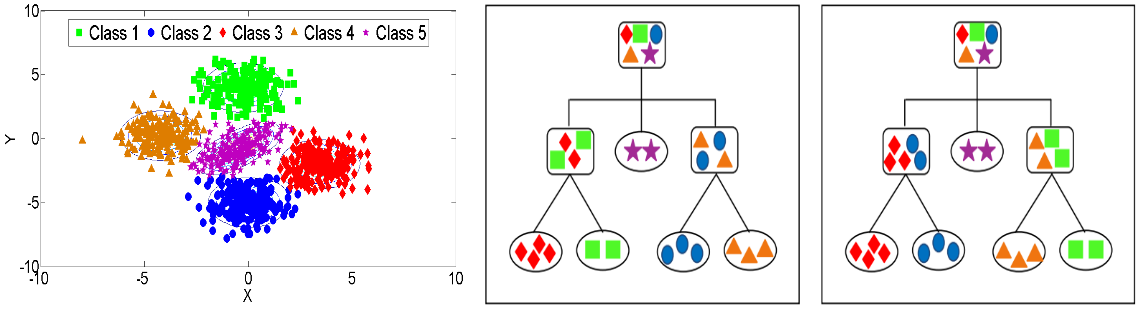
Hierarchical structure inconsistency - Designing a consistent hierarchy is challenging either due to insufficient domain knowledge or several confounding classes (such as *soc.religion.christian* and *talk.religion.misc* classes in Newsgroup dataset[4], both relate

[1] http://lshtc.iit.demokritos.gr/

[2] http://image-net.org/

[3] http://www.bioasq.org/

[4] http://qwone.com/∼jason/20Newsgroups



(a) Synthetic dataset with five class

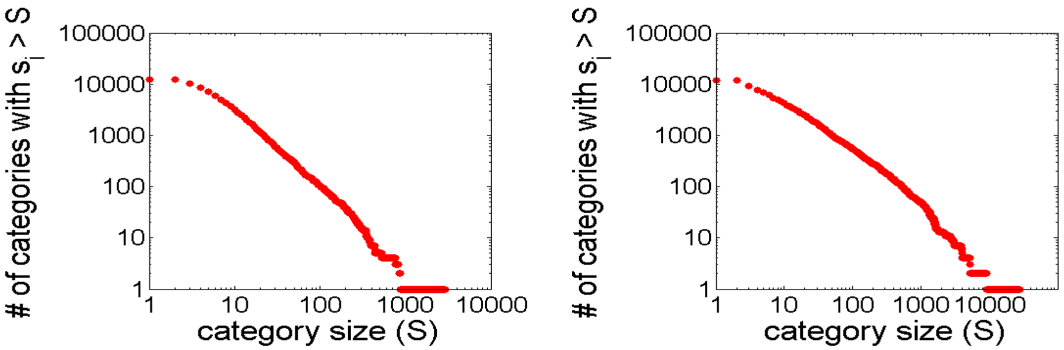
(b) Hierarchy 1

(c) Hierarchy 2

Figure 1.1: (a) Synthetic dataset with five classes (marked with different symbol and color) and two different hierarchical structure shown in (b) and (c).

to religion). Moreover, hierarchy generation based on semantics is susceptible to inconsistencies [2, 10]. This problem is more common for large-scale datasets. To illustrate in detail, consider the example shown in Figure 1.1(a), it consist of 1000 points divided into five classes that are generated using gaussian distribution with different mean and variance. Figure 1.1(b) and Figure 1.1(c) shows two different possible hierarchical structures with these classes. *Hierarchy 1* separates examples into three categories at level 1, namely {(♦, 🟩),(★),(▲, ●)} which are not consistent for classification since categories (♦, 🟩) and (▲, ●) are not easily separable (assuming linear separators) whereas *Hierarchy 2* is more consistent since it groups easily separable classes together. This explains intuitively that inconsistent hierarchical structure can deteriorate the classification performance. To overcome this problem, we will discuss various approaches that we have developed to remove inconsistencies from the hierarchy prior to learning models.

Rare categories - Large-scale HC problem involves thousands of categories with varying distribution of instances per category. In datasets of such a large-scale, skewed class distributions is observed where plenty of classes have few examples for training (for *e.g.* more than 75% of LSHTC datasets belongs to rare categories with less than 10 examples) making it considerably difficult to learn a generalized model. This is known as the rare categories problem [4] and datasets exhibit power-law distribution for examples per class



(a) DMOZ-2010

(b) DMOZ-2012

Figure 1.2: Power law distribution for DMOZ-2010 and DMOZ-2012 datasets.

as shown in eq. (1.1). To improve HC performance, it is necessary to address the issue of rare categories. In this thesis, we explore various rank-based approaches for improving the performance on rare categories.

$$P(s_i > S) \propto S^{-\gamma} \tag{1.1}$$

where s_i denotes the size of class i and γ denotes the power law exponent. Figure 1.2(a) and (b) show the power-law distribution followed in large-scale DMOZ-2010 and DMOZ-2012 datasets, respectively.

Less cohesive or overlapping categories - Within real-world datasets, two or more categories may be so similar that learning a discriminative model between them is difficult [11]. In such cases, it is beneficial for classification to merge these categories into one common category. For example, categories like 'softball' and 'baseball' have several common features such as *pitch*, *ball*, *bat*, *gloves* due to which discrimination between instances of these two classes becomes difficult without having enough training instances (less likely for large-scale datasets). Alternatively, due to lack of domain knowledge instances with different characteristics, classes are put together into a common category making it less cohesive. In such cases, it is beneficial to split a category into multiple sub-categories for effective classification. Typically, classes with more generic representation are less cohesive whereas over simplified classes have overlapping instances. In order to improve performance, it is important to resolve less cohesive and overlapping categories from the hierarchy.

Orphan node detection - Due to change in distinctive characteristics of data over time; we often need to evolve/adapt the hierarchy by creating a new set of classes whose instances have not yet been observed in the training data. This problem referred as orphan node detection (or identification) is an important factor to consider for improving the HC performance [12]. In this thesis, we will discuss about an approach that we have developed for detecting the orphan nodes in the hierarchy. Our approach is easily integrable to exisiting top-down HC prediction approach with minor changes.

Scalability - Large-scale HC are often characterized by data consisting of thousands of categories and millions of instances with high dimensional features. Dealing with data of such a large-scale motivates the development of scalable approaches that can be easily parallelized across multiple compute nodes. To this end, several distributed approaches have been proposed [13,14]. However, there is specific hardware and software requirements which limits the applicability of this approach. To overcome this, we propose to solve this problem using feature selection (FS) techniques that helps to improve performance by removing the redundant and irrelevant features while being computationally tractable due to squeezing of high-dimensional features that reduces the time required between optimization iterations. Further, FS also helps to reduce the memory required for storing model parameters which is of the utmost importance in large-scale settings.

1.2 Contributions

In this thesis, we have addressed some of the issues that are prevalent with large-scale HC. Specifically, the main contributions can be summarized as follows:

1. Developed local and global flattening approaches that are able to better identify the set of inconsistent nodes that exists within the hierarchy, thereby improving the HC performance [2].

2. Developed an efficient filter-based rewiring approach for taxonomy modification which

unlike previous wrapper based approaches does not require multiple, expensive computations. Our approach is scalable and can be applied to the HC problems with high-dimensional features, large number of classes and examples [10].

3. Developed global and adaptive approaches for embedding state-of-the-art feature selection algorithms into HC framework [15].

4. Developed rank-based models to improve the performance on classes with rare categories. In addition, hierarchical information is exploited into the model learning to boost the performance further [16].

5. Developed multi-stage embarrassingly parallel integrated framework to address the multiple issues faced with large-scale HC. Moreover, we also propose an exploratory learning approach for orphan node detection that can be easily incorporated in the integrated framework [17].

1.3 Thesis Outline

Chapter 2 discusses about the symbols and notations used in this thesis and provides a brief overview of various HC methods in the literature. In Chapter 3 and 4, various hierarchical structure modification approaches to resolve inconsistencies within the hierarchy have been proposed. Chapter 5 provides insight into different approaches for integrating information theoretic feature selection methods into the HC framework followed by Chapter 6 that discusses about the various rank-based approaches we have developed to solve the rare categories problem. In Chapter 7, multi-stage integrated pipeline is discussed that is beneficial for improving the overall HC performance by addressing multiple HC issues. Finally, we conclude and discuss about the various future research directions in Chapter 8.

Chapter 2: Background

This chapter presents the background work for research addressed by this thesis. Specifically, this chapter gives a broad overview of hierarchical classification and inconsistencies within the hierarchy definition. Many work has been accomplished by the researchers in this area. However, we will restrict our discussion to those methods that are useful for understanding and grasping the materials presented in this thesis.

2.1 Notations

This section discusses the commonly used notations in this thesis. Summary of notations are described in Table 3.1. For ease of understanding, vectors and matrices are denoted by **bold** letters. Moreover, for representing matrices we have used uppercase letters and vectors are represented using lowercase letters.

Specific to the HC problem, $\mathcal{N}$ denotes the total number of nodes in the hierarchy. Total number of training instances is denoted by symbol N, where N input training pairs are represented using $\mathcal{D} = \{(\mathbf{x}(i), y(i))\}_{i=1}^{N}$ where $\mathbf{x}(i) \in \mathcal{X}$ corresponds to the i-th input vector in the input domain (space) $\mathcal{X}$ and $y(i) \in \mathcal{Y}$ corresponds to the true label in the ouput domain (space) $\mathcal{Y}$. For binary classifiers, learned optimal model weight vectors corresponding to the n-th node in the hierarchy $\mathcal{H}$ is represented using $\mathbf{w}_n$ whereas the group of m weight vectors are represented using the notation $[\mathbf{W}]_{m*d}$, where d corresponds to the dimensionality (number of features) of the input vector. For multi-class classifiers, multiple classifiers are trained at each of the internal node in the hierarchy. Therefore, we use the notation $\mathbf{w}_n^c$ to represent the learned model corresponding to c-th child of node n whereas combined model at node n is represented using $\mathbf{W}_n = [\mathbf{w}_n^c]_{c \in \mathcal{C}(n)}$. $\mathcal{C}(n)$ denotes the set of children of node n

Table 2.1: Notations

Symbol	Description
$\mathcal{H}$	original given hierarchy
$\mathcal{N}$	set of nodes in $\mathcal{H}$
N	total number of training examples
$\mathbb{R}$	set of real numbers
d or $\mathcal{F}$	dimensionality (number of feature) of input vector
$\mathcal{L}$	set of leaf categories (classes or labels)
$\mathbf{x}(i) \in \mathbb{R}^d$	input vector for i-th training example
$y(i) \in \mathcal{L}$	true label for i-th training example
$\mathbf{w}_n$	learned model weight vectors using binary classifiers for n-th node in the hierarchy
$\mathbf{w}_n^c$	learned model weight vectors using multi-class classifiers corresponding to c-th children of n-th node in the hierarchy
$\mathcal{C}(n)$	set of children for n-th node in the hierarchy
$\pi(n)$	parent of n-th node in the hierarchy
$\mathcal{S}(n)$	set of siblings for n-th node in the hierarchy
$\mathcal{A}(n)$	set of n-th node ancestors including the node itself but excluding root
$\hat{\mathbf{x}}(i) \in \mathbb{R}^d$	input vector for i-th test example
$\hat{y}(i) \in \mathcal{L}$	predicted label for i-th test example
$y_n(i) \in \pm 1$	binary label used for i-th training example to learn weight vectors for n-th node in the hierarchy, $y_n(i) = 1$ iff $y(i) = n$, -1 otherwise
$y_n^c(i) \in \pm 1$	binary label used for i-th training example to learn weight vector corresponding to c-th child of n-th node in the hierarchy, $y_n^c(i) = 1$ iff $y(i) = c$, -1 otherwise
Ψ_n	optimal objective function value for n-th node in the hierarchy
Ψ_n^c	optimal objective function value for c-th child of n-th node in the hierarchy

whereas parent and siblings of node n are denoted by symbol $\pi(n)$ and $\mathcal{S}(n)$, respectively. $\mathcal{A}(n)$ denotes the set of n-th node ancestors including the node itself but excluding root.

For training binary classifiers at node n, we use the binary label $y_n(i) = \pm 1$ for i-th training instance where $y_n(i) = 1$ iff $y(i) = n$ and -1, otherwise. Similarly, for multi-class classifiers we use the binary label $y_n^c(i) = \pm 1$ for i-th training instance where $y_n^c(i) = 1$ iff $y(i) = c$ and -1, otherwise. Predicted label for the i-th test instance $\hat{\mathbf{x}}(i)$ is represented using the notation $\hat{y}(i)$.

2.2 Hierarchical Classification

Hierarchical Classification (HC) is one of the most important problem in data mining and machine learning community that has received significant interest among researchers due to its practical importance. In general, HC problem can be formally defined as:

Definition 1 (Problem Definition)**.** *Given, a hierarchy $\mathcal{H}$ defined over the output (label) space $\mathcal{Y}$ and a set of N training examples composed of pairs $\mathcal{D}=\{(\mathbf{x}(i), y(i))\}_{i=1}^{N}$, where $(\mathbf{x}(i), y(i)) \in \mathcal{X} \times \mathcal{Y}$, the goal of the hierarchical classification is to learn a mapping function $f : \mathcal{X} \in \mathbb{R}^d \rightarrow \mathcal{Y}$ that maps the inputs in the input space $\mathcal{X}$ to outputs in the output space $\mathcal{Y}$, such that the function f is accurately able to predict the output y of an input instance x and generalizes well to data that is not observed during the training.*

In literature various method exists to solve the HC problem based on how the hierarchical relationships information is leveraged during the model learning [3]. One of the simplest approach, known as flat classification disregards the hierarchical structure and train classifiers for each of the leaf categories to discriminate from remaining leaf categories. Other approaches involve utilizing the hierarchical relationships information during the learning and/or predicting phase. For example, local and global classifiers. While local classifiers are trained by splitting the hierarchical structure into several smaller structures for utilizing the local relationships, global classifiers are trained by considering the entire class hierarchy at once. In next few sections we will discuss various existing methods for each of these approaches in detail.

2.2.1 Flat Classification Approach

This is one of the simplest and straight forward implementation of the standard classification algorithm into the HC problem. In this method, we ignore the hierarchy and train an independent one-vs-rest binary or multi-class classifiers corresponding to each of the leaf categories that can discriminate it from remaining leaf categories. Label prediction $\hat{y}$ for an unknown test instance $\mathbf{x}$ is done according to the rule shown in eq. (2.1).

$$\hat{y} = \underset{y \, \in \, \mathcal{Y}}{\mathbf{argmax}} \, f(\mathbf{x}, y|\mathbf{w}) \tag{2.1}$$

where the function $f : \mathcal{X} \times \mathcal{Y}$ is parameterized by the model weight vector $\mathbf{w}$.

This approach provides an indirect solution to the HC problem because all the ancestors associated with the predicted leaf category are also assigned to the test instance. Some of the well-known standard formulation of binary [18] and multi-class [19] classifiers that can be used for flat classification is shown in eq. (2.2) and eq. (2.3).

$$\textbf{Binary classifier} \quad \underset{\mathbf{w}}{\textbf{minimize}} : \sum_{i=1}^{N} \xi\Big(\mathbf{w}; \mathbf{x}(i), y(i)\Big) + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{2.2}$$

$$\begin{aligned} \textbf{Multi} - \textbf{class classifier} \quad & \underset{\mathbf{w}_l, \xi_i}{\textbf{minimize}} : \sum_{i=1}^{N} \xi_i + \frac{\lambda}{2} \sum_{l=1}^{|\mathcal{L}|} ||\mathbf{w}_l||_2^2 \\ & \textbf{subject to} : \mathbf{w}_{y(i)}^T \mathbf{x}(i) - \mathbf{w}_l^T \mathbf{x}(i) \geq e_i^l - \xi_i, i = 1, \cdots, N \\ & \textbf{where} : e_i^l = \begin{cases} 0 & \text{if } y(i) = l \\ 1 & otherwise \end{cases} \end{aligned} \tag{2.3}$$

where $\lambda > 0$ is the penalty parameter, $(\xi, \xi_i) \geq 0$ denotes the loss function such as hinge loss or logistic loss and $|| \cdot ||_2^2$ denotes the squared l_2-norm.

Although, flat classification approach is known for its simplicity and has been shown to work well in practice for small and well-balanced datasets [20], it's performance suffers when the number of classes (categories) that needs to be discriminated becomes huge and are not balanced [4], potentially containing lots of rare categories. It also has a major problem with longer training and prediction time because it consider all the examples during the model training and invoke all the models for the leaf categories to make label prediction making it computationally expensive, especially when the number of classes becomes huge.

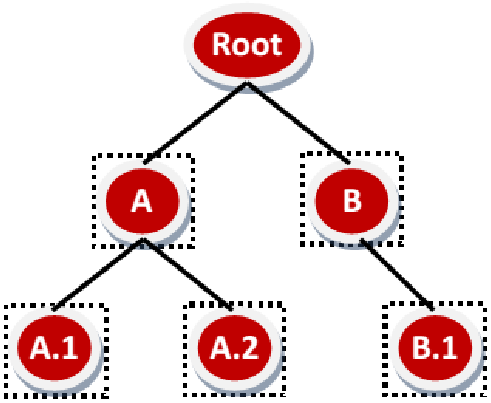


Figure 2.1: Local classifier per node.

2.2.2 Local Classification Approach

This method explores local hierarchical structure relationships information such as parent-child and siblings relationships during the model learning. Based on how the local information is extracted during the model learning, local classification approach can be further categorized into three broad categories (i) Local Classifier per Node approach (LCN) (ii) Local Classifier per Parent Node approach (LCPN) and (iii) Local Classifier per Level approach (LCL).

Local Classifier per Node Approach

In this approach, binary classifier f_n is learned for each node (except root) $n \in \mathcal{N}$ in the hierarchy $\mathcal{H}$ as shown in Figure 2.1. The dashed squares in the figure represents binary classifiers. Goal of this approach is to learn the classifiers that can effectively discriminate between the sibling nodes in the hierarchy. Usually, for training the classifier at a node, we assign examples belonging to the n-th node and its descendants as the positive training examples, and those belonging to the siblings of the n-th node and there descendants as the negative examples. However, in literature different criteria for defining the positive and negative examples has been used [21–23].

To make the label prediction of an unknown test instance $\mathbf{x}$, the algorithm (shown in

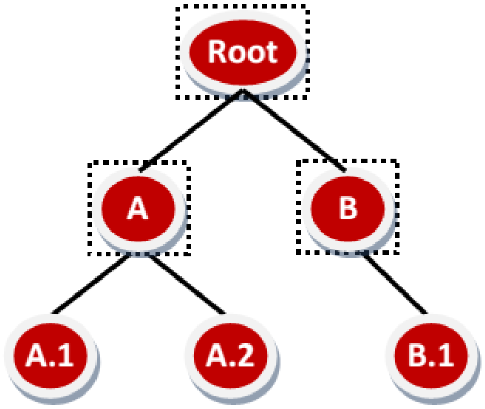


Figure 2.2: Local classifier per parent node.

eq. (2.4)) typically proceeds in the top-down fashion starting at the root and recursively selecting the best children till it reaches a terminal node that belong to the set of leaf categories $\mathcal{L}$, which is the final predicted label.

$$\hat{y} = \left\{ \begin{array}{l} \textbf{initialize} \quad n := root \\ \\ \textbf{while } n \notin \mathcal{L} \\ \\ \quad n := \textbf{arg max}_{q \in \mathcal{C}(n)} \; f_q(\mathbf{x}) \\ \\ \textbf{return } n \end{array} \right\} \tag{2.4}$$

This approach although popular in literature, suffers from training a large number of binary classifiers when the size of hierarchy becomes huge.

Local Classifier per Parent Node Approach

In this approach, multi-class classifier is learnt for each of the parent nodes in the hierarchy $\mathcal{H}$ as shown in Figure 2.2. The dashed square in the figure represents multi-class classifiers. Like local classifier per node approach, goal of this approach is to learn classifiers that can effectively discriminate between the siblings. For training the classifier at each parent node p we use the examples from its descendants where each of the children categories $\mathcal{C}(p)$ of

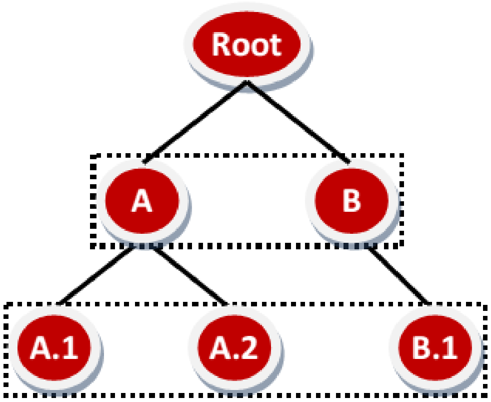


Figure 2.3: Local classifier per level.

parent node p corresponds to different class. Predicting the label for an unknown test instance $\mathbf{x}$ is done in a similar manner as shown in eq. (2.4).

Local Classifier per Level Approach

In this approach, multi-class classifier is learnt for each level in the hierarchy as shown in Figure 2.3. Among local approaches, this is the least popular approach in the literature. For training the classifier at each level, we use the examples from nodes in the level and its descendants, where different nodes in the level corresponds to different class. Prediction for an unknown test instance $\mathbf{x}$ is done by choosing the best node at each level in the hierarchy $\mathcal{H}$. Since classifiers at each level makes independent predictions, there is possibility that this approach may lead to inconsistent prediction. For example, in Figure 2.3 inconsistent prediction occurs if the prediction made by the level 1 classifier is B whereas the level 2 classifier predicts A.2 that corresponds to different branch in the hierarchy. In order to make this approach useful, classification results are complemented with a post-processing step to resolve such inconsistent predictions.

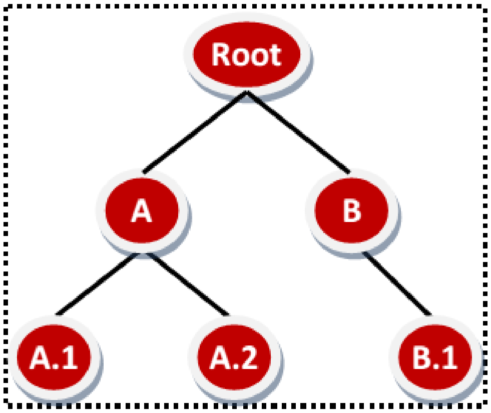


Figure 2.4: Global classifier.

2.2.3 Global Classification Approach

This approach is often referred as the big-bang approach in the literature [24]. Unlike local approaches, global classification approach learns a single complex classification model by taking into account the class hierarchy as a whole as shown in Figure 2.4. For predicting the labels of an unknown test instance $\mathbf{x}$, an approach similar to flat or local methods is followed.

2.2.4 Literature Review

There have been numerous work proposed in the literature to address the problem of HC. In this section we will review some of the most commonly used methods.

Introduced orthogonality between the node and its ancestors, Zhou *et al.* [9]

Classifying sibling classes at lower level becomes difficult as the depth of the hierarchy increases because the classes become more similar to each other. To address this problem, a hierarchical SVM that enforces the learned model parameters of the node to be orthogonal to its ancestor nodes was proposed in the paper. Orthogonality between the i-th node and its ancestor nodes $j \in \mathcal{A}(i)$ is incorporated by introducing the regularization constraints $|\mathbf{w}_i^T \mathbf{w}_j|$ into the minimization objective function. Optimization formulation for there proposed solution can be represented using the equation shown in eq. (2.5).

$$\begin{aligned} \textbf{minimize}: &\quad \frac{1}{2}\sum_{i,j=1}^{m} K_{ij}|\mathbf{w}_i^T\mathbf{w}_j| + \frac{C}{N}\sum_{k=1}^{N}\xi_k \\ \textbf{subject to}: &\quad \mathbf{w}_i^T\mathbf{x}(k) - \mathbf{w}_j^T\mathbf{x}(k) \geq 1-\xi_k, \forall j \in S(i), \forall i \in \mathcal{A}\Big(y(k)\Big), \xi_k \geq 0, \forall k \in \{1,\cdots,N\} \end{aligned} \tag{2.5}$$

where $C > 0$ is a penalty parameter, $\xi_k > 0$ is the loss function and $K_{ij} \geq 0$ captures the hierarchical structure relationships. More precisely, $K_{ij} = 0$, if node i is neither an ancestor nor a descendant of node j, otherwise $K_{ij} > 0$.

Shrinking data sparse leaf node model parameters towards data rich ancestor nodes, McCallum *et al.* [25]

Insufficient examples available for training at the leaf categories is one of the main reason for inferior classification performance. To overcome this, a well established statistical technique known as *shrinkage* is explored in this paper. Learned model parameter estimates of the data sparse leaf node is generalized by enforcing the parameter smoothness with the data rich ancestors. Probabilistically, we can define the marginal probability of generating an input instance $\mathbf{x}$ given the model parameters $\mathbf{w}$ as the sum of total probability over individual components using the equation shown in eq. (2.6).

$$P(\mathbf{x}|\mathbf{w}) = \sum_{y\in\mathcal{Y}} P(y|\mathbf{w})P(\mathbf{x}|\mathbf{w}_y) \tag{2.6}$$

where $P(\mathbf{x}|\mathbf{w}_y)$ can be computed as the product of probability estimates for individual input component as shown in eq. (2.7).

$$P(\mathbf{x}|\mathbf{w}_y) = P(|\mathbf{x}|)\prod_{k=1}^{|\mathbf{x}|} P(\mathbf{x}_k|\mathbf{w}_y) \tag{2.7}$$

Given the initial parameter estimates of the learned model weight vectors corresponding to class y and its ancestor as $\left\{\mathbf{w}_y^i\right\}_{i \in \mathcal{A}(y)}$. Parameter smoothing can be applied by shrinking the learned parameters of class y to its ancestor categories as per the rule shown in eq. (2.8).

$$\widehat{\mathbf{w}}_y = \sum_{i \in \mathcal{A}(y)} \lambda_y^i \mathbf{w}_y^i \tag{2.8}$$

where $\widehat{\mathbf{w}}_y$ denotes the final learned parameter estimates corresponding to the class y.

Two stage classification for large scale taxonomy, Xue *et al.* [26]

As the hierarchy size increases, learning models for each node in the hierarchy becomes difficult. To address this issue Xue *et al.* proposed two stage classification approach. For each test document, in the first stage a set of candidate categories is retrieved based on similarity to the test document. Then the second stage builds a classifier on the hierarchy restricted to the set of categories fetched in first stage and classifies the test document using the restricted hierarchy. Although, pruning reduces the hierarchy to a manageable size, one severe drawback of this approach is having to train a different classifier for each test document.

Parent-child regularization, Gopal *et al.* [6]

Hierarchical dependencies between different classes in the hierarchy is leveraged by regularizing the model parameters of each class with their parent class. They proposed two models, HR-SVM and HR-LR based on the type of loss function used for classifier training.

The proposed formulation of their approach is shown in eq. (2.9) and eq. (2.10).

$$\mathbf{HR-SVM} \quad \underset{\mathbf{W}}{\mathbf{minimize}} \sum_{n \in \mathcal{N}} \frac{1}{2} ||\mathbf{w}_n - \mathbf{w}_{\pi(n)}||^2 + C \sum_{n \in \mathcal{L}} \sum_{i=1}^{N} \Big[1 - y_n(i) \mathbf{w}_n^T \mathbf{x}(i)\Big]_+ \tag{2.9}$$

$$\mathbf{HR-LR} \quad \underset{\mathbf{W}}{\mathbf{minimize}} \sum_{n \in \mathcal{N}} \frac{1}{2} ||\mathbf{w}_n - \mathbf{w}_{\pi(n)}||^2 + C \sum_{n \in \mathcal{L}} \sum_{i=1}^{N} \mathbf{log}\Big(1 + exp\Big(- y_n(i) \mathbf{w}_n^T \mathbf{x}(i)\Big)\Big) \tag{2.10}$$

$$\mathbf{where} : y_n(i) = \begin{cases} 1 & \text{if } \mathbf{x}(i) \text{ belongs to class } n \in \mathcal{L} \\ -1 & \text{otherwise} \end{cases}$$

HR-SVM and HR-LR models gives state-of-the-art performance results on LSHTC datasets and can also be easily parallelized by optimizing the alternate even and odd levels at subsequent iterations. This is possible because there is no dependency between even-even or odd-odd levels.

Refined Experts, Bennett *et al.* [7]

Hierarchical classification problem suffers from two significant challenges - *error propagation* and increasingly complex $non-linear\ decision\ surfaces$ at higher levels in the hierarchy. To overcome this problem, author proposed the method of refined experts, where $refinement$ method is used to eliminate the error propagation by changing the training distribution based on cross-validation results to prevent errors and *expert* extraction of meta-features at lower levels is done to improve the decision boundary at higher levels. Combining both of this step into top-down classification settings is referred by the author as $refined\ experts.$ Empirical evaluation of their proposed approach shows an improvement upto 30% in Macro-F_1 score.

2.3 Inconsistencies within the Hierarchy

Most of the HC approaches often use hierarchy during the learning process to design appropriate loss function for classification. Their performance can be severely affected if the hierarchy used for learning is not well-suited for classification purpose. In majority of the cases, the hierarchy used for training is manually designed by the domain experts that reflects the human view of the domain. This manual process of hierarchy creation suffers from various design issues that makes it unsuitable to achieve high classification accuracy. Major reason behind such issues includes: (i) hierarchy is designed for the sole purpose of easy search and navigation without taking classification into consideration. (ii) a-priori it is not clear to domain experts when to generate new nodes (*i.e.* hierarchy expansion) or merge two or more nodes (*i.e.* link creation) in the hierarchy and it is often left at the discretion of domain experts to decide, which results in certain degree of arbitrariness (iii) large number of classes poses a unique challenge for manual design of good hierarchy.

Hierarchical inconsistencies is a general term that refers to the various types of inconsistency that can exist within the hierarchy, which makes it difficult to learn generalized classifiers. Based on the adapted hierarchy creation process, two types of inconsistency can exist within the hierarchy. (i) *inconsistent node* - obtained due to over simplification process of creating new node to combine two or more specific concepts into one general concept and (ii) *inconsistent link* - obtained due to domain experts personal choice or bias. Figure 2.5 shows the various types of inconsistencies. Similar type of nodes are marked with same color in the figure, red edge (link) in the figure denotes inconsistent link whereas the inconsistent node is marked with red rectangular box.

Various methods to solve hierarchical inconsistencies issue have been proposed in the literature. These methods can be broadly categorized into two different categories. First category of methods ignore the existing hierarchy and generates its own hierarchy (automatic hierarchy generation) based on some methods such as clustering whereas second category of methods aim to modify the provided hierarchy using some elementary operations such as

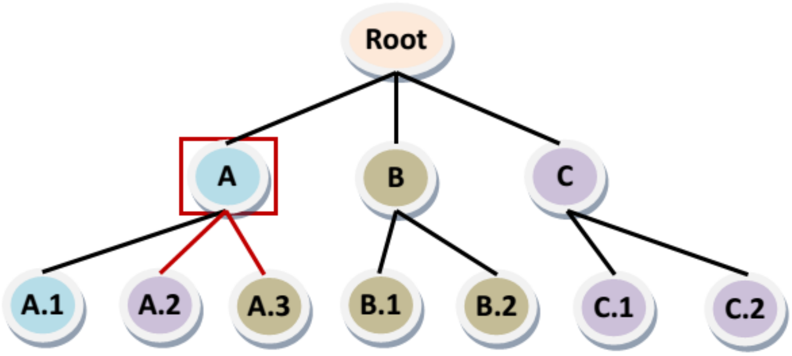


Figure 2.5: Hierarchy with inconsistent node and links (marked in red).

insert and delete, to make it better suited for obtaining higher classification accuracy. Both methods have their own advantages and disadvantages. Automated generation of hierarchy requires pre-processing time for creating hierarchy which may be computationally expensive if the number of classes are huge. It also requires the number of hierarchical levels to be pre-defined as an input to the algorithm which may be unknown and difficult to decide. On positive side, automated generated hierarchy may results in better classification performance possibly because of lesser number of inconsistent nodes and links in the hierarchy. On the contrary, modifying the existing hierarchy is less time consuming but it requires the decision to be made at each step of the hierarchy modification (using elementary operation), which can be very difficult. In the next few paragraphs we will discuss in detail the literature of existing algorithms for both of these methods.

2.3.1 Automated Hierarchy Generation Approach

Linear discriminant projection and clustering approach, Li *et al.* [1]

In this paper, author has proposed the use of linear discriminant projection to transform all instances to lower dimensional space before performing the hierarchical agglomerative clustering, which produces meaningful hierarchy of clusters. Hierarchies generated using this method showed improved classification performance but the main drawback of their approach is that they ignore the original hierarchical structure which may carry some important information. In addition, there is no theoretical guarantee for deciding the number of levels that can be used to achieve the best classification performance. Moreover, this approach is practically not suitable for large scale problems due to the projection step which is not scalable.

2.3.2 Existing Hierarchy Modification Approach

Modification based on promote, demote and merge operations, Tang *et al.* [27]

Empirical study done by author showed varying classification performance with different hierarchical structures. To this end, author proposed the usage of three basic elementary operations - promote, demote and merge, on the existing hierarchy to make it suitable for achieving best classification performance. At each step of hierarchy modification, one of the three operations is applied to obtain new hierarchy which is better than the previous best hierarchy. The process of hierarchy modification is repeated until there is no more possible modifications which can lead to the performance improvement. Experiments on text dataset showed improved performance but there are few caveats of this approach: (i) Deciding which operation to apply first and which part of the hierarchy to explore first is one of the critical factors governing runtime performance. Although there is no justification for these issues, author pointed out that with some intelligent heuristics this problem can be simplified. (ii) After each step of hierarchy modification there is an evaluation phase which validates if the modified hierarchy performed better compared to the last one. This step can be expensive if the hierarchy under consideration has huge number of classes. To some extent evaluation phase computation can be reduced by restricting the analysis to the part of the hierarchy where modification has been done. This modified approach has been explored in [28].

Genetic Algorithm for improving performance, Qi and Davison [29]

Considering only the current best obtained hierarchy for improving the performance may not be an optimal approach, so the author considers using multiple best performing hierarchies at each step for improving the performance using genetic-based algorithm. Author proposed different methods for adapting the genetic operations such as mutations and cross-over operations to the hierarchical settings. Experiments on multiple classification tasks showed that their proposed algorithm can significantly improve classification task. However, the performance is highly dependent on the hierarchies and the operators selected at each step.

Maximum Margin based Approach, Babbar *et al.* [30]

In this paper author identified that level(s) flattening [31] may not be an optimal strategy to reduce error propagation. To overcome this author proposed removing only the problematic (inconsistent) nodes in the hierarchy for achieving better classification performance. Set of inconsistent nodes within the existing hierarchy is determined using the maximum margin value that is obtained at each node using the SVM-based discriminative classifiers. Improved accuracy and runtime performance was observed while comparing their proposed approach with other level flattening approaches such as top-level flattening, bottom-level flattening and multiple-level flattening [31, 32].

2.4 Evaluation Metrics for Hierarchical Classification

2.4.1 Flat Measures

Standard set based measures [33] such as Micro-F_1 (μF_1) and Macro-F_1 (MF_1) are used for evaluating the performance of various methods. To compute μF_1, we sum up the category specific true positives (TP_c), false positives (FP_c) and false negatives (FN_c) for different categories and compute the μF_1 score as:

$$\begin{aligned}
P &= \frac{\sum_{c \in \mathcal{L}} TP_c}{\sum_{c \in \mathcal{L}} (TP_c + FP_c)} \\
R &= \frac{\sum_{c \in \mathcal{L}} TP_c}{\sum_{c \in \mathcal{L}} (TP_c + FN_c)} \\
\mu F_1 &= \frac{2PR}{P + R}
\end{aligned} \tag{2.11}$$

Unlike μF_1, MF_1 gives equal weight to all the categories so that the average score is not skewed in favor of the larger categories. MF_1 is defined as follows:

$$\begin{aligned}
P_c &= \frac{TP_c}{TP_c + FP_c} \\
R_c &= \frac{TP_c}{TP_c + FN_c} \\
MF_1 &= \frac{1}{|\mathcal{L}|} \sum_{c \in \mathcal{L}} \frac{2 P_c R_c}{P_c + R_c}
\end{aligned} \tag{2.12}$$

where, $|\mathcal{L}|$ is the number of categories (classes).

2.4.2 Hierarchical Measures

Different from flat measures that penalizes each of the misclassified examples equally, hierarchical measures take into consideration hierarchical distance between the true label and predicted label for evaluating the classifier performance [34]. In general, misclassifications that are closer to the actual class are less severe than misclassifications that are farther from the true class with respect to the hierarchy (for *e.g.*, an example from *hockey* class misclassified as the *baseball* class is less severe in comparison to the *hockey* misclassified as *cat*). The hierarchy based measures include hierarchical F_1 (hF_1) (harmonic mean of hierarchical precision (hP), hierarchical recall (hR)) and tree-induced error (TE) [35], which are defined as follows:

$$\begin{aligned}
hP &= \frac{\sum_{i=1}^{N} \left|\mathcal{A}\Big(\hat{y}(i)\Big) \cap \mathcal{A}\Big(y(i)\Big)\right|}{\sum_{i=1}^{N} \left|\mathcal{A}\Big(\hat{y}(i)\Big)\right|} \\
hR &= \frac{\sum_{i=1}^{N} \left|\mathcal{A}\Big(\hat{y}(i)\Big) \cap \mathcal{A}\Big(y(i)\Big)\right|}{\sum_{i=1}^{N} \left|\mathcal{A}\Big(y(i)\Big)\right|} \\
hF_1 &= \frac{2 * hP * hR}{hP + hR} \qquad (2.13) \\
TE &= \frac{1}{N}\sum_{i=1}^{N} \delta\Big(\hat{y}(i), y(i)\Big) \qquad (2.14)
\end{aligned}$$

where, $\mathcal{A}\Big(\hat{y}(i)\Big)$ and $\mathcal{A}\Big(y(i)\Big)$ are respectively the sets of ancestors of the predicted and true labels which include the label itself, but do not include the root node. $\hat{y}(i)$ is the predicted label and $y(i)$ is the true label of example i, $\delta(a, b)$ gives the length of the undirected path between categories a and b in the graph.

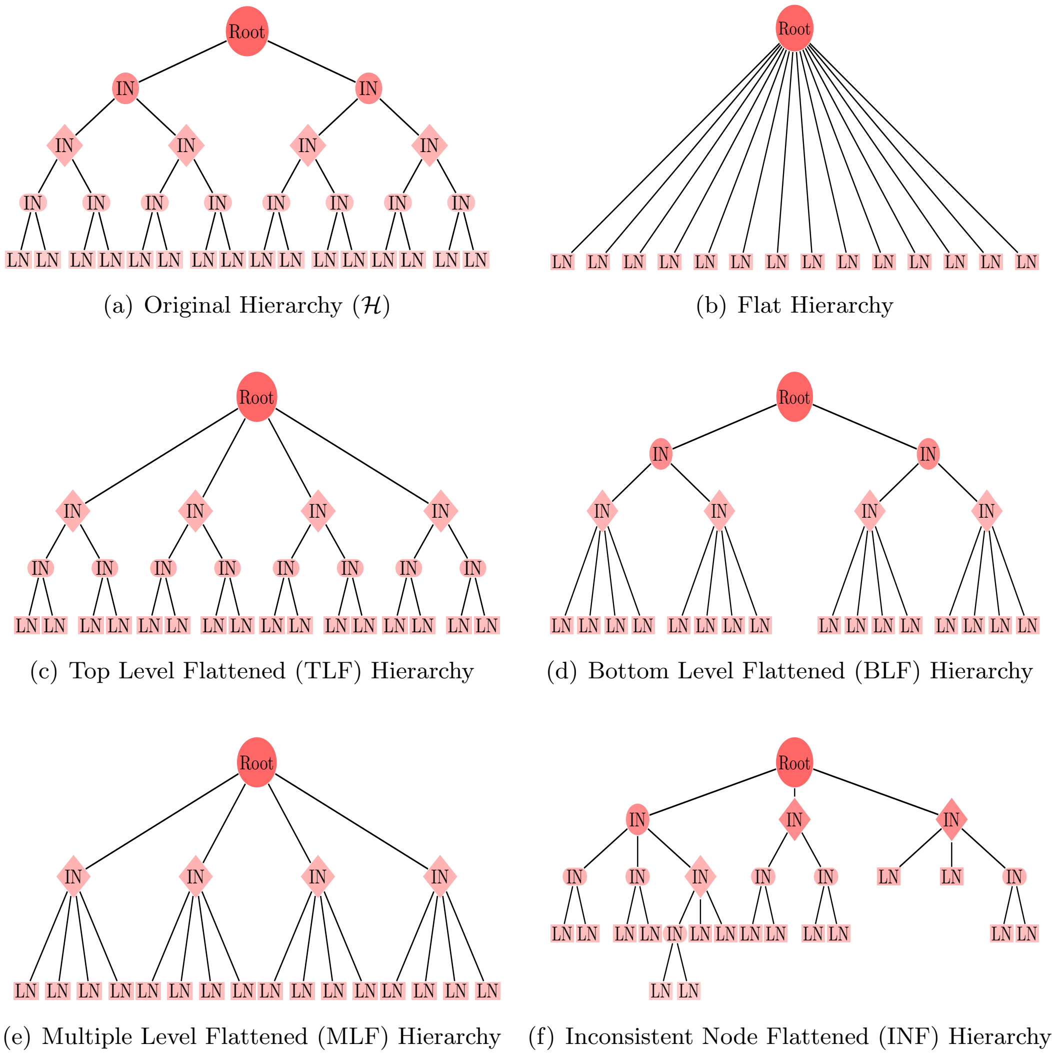
It should be noted that for test dataset with orphan nodes, results are reported by considering only the seed classes *i.e.*, having at least one training instance as reported in other studies such as Dalvi *et al.* [12]. Moreover, for hierarchical inconsistencies we have used the original hierarchy for all methods for consistent evaluation of hierarchical measures unless noted.

Chapter 3: Flattened Hierarchy for Removing Inconsistencies

3.1 Introduction

In the past, various methods have been developed to improve the HC performance [3]. One of the simplest method is to learn binary one-versus-rest classifier for each of the leaf categories, ignoring the hierarchical relationships. This method is referred as flat classification. Other methods involve use of the hierarchies (see Section 3.2) during the learning and prediction process. Hierarchies provide useful structural relationships (such as parent-child and siblings) among different classes that can be exploited for learning generalized classification models. Previously, researchers have demonstrated the usefulness of hierarchies for classification and have obtained promising results [6, 8, 25, 26, 36–38]. However, in many situations hierarchies used for learning models are not consistent due to the presence of *inconsistent* nodes (and links) resulting in excessive error propagation. As such, HC approaches are outperformed by the flat classifiers that completely ignore the hierarchy [20, 39].

Flat classifiers, though effective in some cases, suffer from two major issues: (i) During the prediction phase, flat classifiers invoke all the models for leaf categories and are considerably slower than top-down HC approaches, in which only the models in the relevant path are invoked. (ii) For large-scale HC problems, it is challenging to learn effective classification models that can discriminate between large number of classes. This problem is worse for datasets with skewed class distributions where plenty of classes have very few examples for training (rare categories problem) [4]. Large-scale datasets show power-law distribution of examples per category [40]. Considering these issues, the focus of this work is to improve top-down HC approaches, which are computationally feasible for large-scale datasets and



(a) Original Hierarchy ($\mathcal{H}$)

(b) Flat Hierarchy

(c) Top Level Flattened (TLF) Hierarchy

(d) Bottom Level Flattened (BLF) Hierarchy

(e) Multiple Level Flattened (MLF) Hierarchy

(f) Inconsistent Node Flattened (INF) Hierarchy

Figure 3.1: Various hierarchical structures (b)-(f) obtained after flattening some of the nodes (or levels) from the original hierarchy shown in (a). 'IN' denotes the internal node and 'LN' denotes the leaf node.

handle the imbalance problem by utilizing structural relationships.

The main drawback of top-down HC approaches that contributes to their inferior classification performance is error propagation — compounding of errors from misclassifications at higher levels which cannot be rectified at the lower levels. This problem can be alleviated to certain extent by restructuring (modifying) the hierarchy to remove inconsistent nodes that causes performance deterioration. In this work, our main contribution includes development of data-driven approaches for removing inconsistencies in the expert defined (original) hierarchy leading to a hierarchy that achieves higher classsification performance irrespective of the HC approach used for training. We propose a *flattening* approach where inconsistent nodes are selectively removed from the hierarchy. The criterion for flattening a node is based on the optimal regularized risk minimization objective value attained by the model trained for that node on a separate validation set. If the objective value for a node n is above a certain threshold, then we flatten n, *i.e.*, we remove n from the hierarchy and add its children to n's parent node. Based on the strategy adapted for identifying inconsistent nodes, we propose two different approaches for inconsistent nodes flattening (INF) from the hierarchy: (i) Local approach (Level-INF) that computes a level-wise cutoff threshold and (ii) Global approach (Global-INF) that computes a global threshold for the entire hierarchy.

Experimental comparisons of top-down HC approach on our proposed modified hierarchy shows statistically significant performance improvement in comparison to the baseline hierarchy (expert defined or original) and other comparative methods for hierarchy modification [30, 31]. We also performed detailed analysis to show that the reduction in misclassification error at higher levels with our proposed hierarchy modification approach leads to reduced error propagation and hence better classification performance. In comparison to flat classification, our approach is more accurate for classes with fewer training instances (rare categories) and is computationally efficient for large-scale HC problems during the prediction phase [41].

3.2 Literature Review

There has been a large body of research focusing on the HC problem. Besides completely ignoring the hierarchy (flat classifiers), one class of HC methods solve various local subproblems that train individual classifier(s) for each of the nodes (or parent nodes) in the hierarchy or learn classifiers for each of the levels. This methods are referred as local classification because only local structural relationship information are used during training these classifiers. To predict the labels of instances, top-down local hierarchical methods proceed by selecting the most relevant node at the topmost level and then recursively selecting the best node until a leaf category is reached, which is the final predicted label. Local approaches are more popular due to their computational benefits [3]. Contrary to local classification, global classification methods [42] learn a single complex model over all the nodes in the hierarchy and are computationally more expensive than flat and local methods. Therefore, in this work we focus on top-down local classification methods for training models and predicting labels.

Some of the earlier studies focus on exploiting hierarchies among categories for the purpose of classification [8,27,35,43], but the number of categories are limited to a few hundreds. One of the earlier breakthroughs in the field of hierarchical text categorization was by Koller *et al.* [36]. This approach used a divide and conquer paradigm for solving the HC problem which can easily be adopted in large-scale settings. Following this, numerous approaches have been developed to improve HC for larger datasets. Liu *et al.* [44] studied the classification performance using a SVM based method that scales for millions of categories. Gopal *et al.* [6] used a regularization term within the optimization function to capture the parent-child relationships in the hierarchy. This approach referred as HR-LR and HR-SVM shows improved classification performance but the training procedure for large-scale problem requires a distributed implementation and map-reduce supported infrastructure. Xue *et al.* [26] proposed a two stage approach. For each test document, in the first stage a set of candidate categories are retrieved based on similarity to the test document. Then the second stage builds a classifier on the hierarchy restricted to the set of categories fetched in first stage and classifies the test document using the restricted hierarchy. Although, pruning reduces the hierarchy to a manageable size, one severe drawback of this approach is having to train a different classifier for each test document which is expensive. Other works in the field of HC can be found in a detailed survey by Silla *et al.* [3].

3.2.1 Hierarchy Modification

As discussed earlier, most HC approaches rely on hierachical relationships for learning complex models to improve the classification performance. However, the performance can be negatively impacted if the hierarchy used during learning models is inconsistent. Therefore, it is of utmost importance to generate an improved hierarchical representation that is suitable for classification prior to learning models. Inconsistencies in the hierarchy are due to the following reasons:

(i) Hierarchies are designed for efficient search and navigation without considering HC performance.

(ii) Hierarchical groupings of categories is done based on semantics, whereas classification depends on data characteristics such as *term frequency*.

(iii) Multiple hierarchies are possible for the same dataset (such as SCOP and CATH [45] for protein structures). However, there is no consensus regarding which hierarchy is better for classification.

(iv) Consistent hierarchy design for datasets with large number of categories is prone to errors.

Several approaches that restructure the hierarchy have been developed in past. Level flattening [31] is one of the approach used in earlier works of hierarchy modification, where some of the levels are flattened (removed) from the original hierarchy prior to learning models. Based on levels that are flattened various methods of modification exist. Top Level

Flattening (TLF) as shown in Figure 3.1(c) modifies the hierarchy by flattening the top level in the original hierarchy. Model learning and prediction for flattened hierarchy is done in similar fashion as top-down methods. Bottom Level Flattening (BLF) and Multiple Level Flattening (MLF), shown in Figures 3.1(d) and 3.1(e) are similar methods of hierarchy modification where bottom and multiple levels are removed, respectively. As done in Wang *et al.* [31], we removed the first and third levels for evaluating the MLF approach.

Babbar *et al.* [30] proposed a maximum-margin based strategy for hierarchy modification. This method selectively removes some of the inconsistent nodes in the hierarchy based on margins rather than removing complete levels. Hierarchy modification using this approach (shown in Figure 3.1(f)), is beneficial for classification and has been theoretically justified [46]. We followed a similar approach for hierarchy modification. However, our method differs in following two aspects: (i) We developed a more systematic approach for threshold selection to identify the inconsistent nodes for flattening. Our approach is based on deviation from mean that is empirically tuned for each dataset, and (ii) We also considered a global perspective of the hierarchy (Global-INF) for threshold selection to identify the inconsistent set of nodes. This approach is more intuitive and realistic measure for threshold selection because it prevents excessive flattening of the nodes that is just based on local decisions, thereby allowing the benefits of leveraging the hierarchy during the model learning and classification process, especially for rare categories (see Section 3.6 for justification).

Hierarchy modification using a supervised learning approaches are also proposed in the literature [4, 27], where the hierarchy is gradually modified to achieve better hierarchy for improving the classification performance. These methods have an expensive evaluation costs that needs to be performed after each modification, making it computationally infeasible for large-scale settings. Hence, we do not compare our approach to these methods. Other competitive methods that involves restructuring the hierarchy are developed by us and appear in an arXiv publication [10].

Table 3.1: Notation description

Symbol	Description
$C > 0$	mis-classification penalty parameter
f_n^*	optimal objective function value for n-th node obtained using validation dataset. We have dropped the subscript n at some places for ease of description
$\mathcal{H}_L$	modified hierarchy after level-wise inconsistent node removal
$\mathcal{H}_G$	modified hierarchy after global inconsistent node removal
$\mathcal{N}_k$	set of nodes at k-th level in $\mathcal{H}$
$\mathcal{I}_L$	set of inconsistent nodes using level-wise INF method
$\mathcal{I}_G$	set of inconsistent nodes using global-INF method
$\mu(\mathcal{S})$	mean of samples in set $\mathcal{S}$
$\sigma(\mathcal{S})$	standard deviation of samples in set $\mathcal{S}$
$\mathcal{S}_k$	set of f^* values for node at k-th level in $\mathcal{H}$
$\mathcal{S}$	set of f^* values for all nodes (except root) in $\mathcal{H}$
τ_k	threshold limit for identifying inconsistent nodes at k-th level in $\mathcal{H}$
τ	threshold limit for identifying inconsistent nodes in $\mathcal{H}$
$\psi_k \geq 0$	fitness parameter for level-wise threshold selection at k-th level in $\mathcal{H}$
$\psi \geq 0$	fitness parameter for global threshold selection in $\mathcal{H}$

3.3 Methods

Table 3.1 summarizes the common notations used in this work.

3.3.1 Problem Setup

Given, a hierarchy $\mathcal{H}$ we train a binary one-vs-rest classifiers for each of the node $n \in \mathcal{N}$ — to discriminate its positive examples from the examples of other nodes (*i.e.*, negative examples) in the hierarchy. We followed the 'inclusive policy' for training classifiers, where all the descendant categories of node n (including node itself) is considered as positive examples and the remaining categories as negative examples [22]. In this work, we have used logistic regression (LR) [47] as the underlying base model for training. The LR objective uses logistic loss to minimize the empirical risk and $l2$-norm term (denoted by $|| \cdot ||_2^2$) to control the model complexity and prevent from overfitting. The objective function for training a model corresponding to node n is provided in eq. (3.1).

$$\min_{\mathbf{w}_n} \left[C \sum_{i=1}^{N} \log \Big(1 + \exp \Big(- y_n(i) \mathbf{w}_n^T \mathbf{x}(i) \Big) \Big) + \frac{1}{2} \left\| \mathbf{w}_n \right\|_2^2 \right] \tag{3.1}$$

For each node n, we solve eq. (3.1) to obtain the optimal weight vector denoted by $\mathbf{w}_n$. The complete set of parameters for all the nodes $\{\mathbf{w}_n\}_{n \in \mathcal{N}}$ constitutes the learned model for the hierarchical top-down classifier. For LR models the conditional probability for $\hat{y}_n(i) \in \pm 1$ given its feature vector $\mathbf{x}(i)$ and the weight vector $\mathbf{w}_n$ is given by eq. (3.2) and the classification decision function using eq. (3.3).

$$P\Big(\hat{y}_n(i) \mid \mathbf{x}(i), \mathbf{w}_n \Big) = \frac{1}{\Big(1 + \exp \Big(- y_n(i) \mathbf{w}_n^T \mathbf{x}(i) \Big) \Big)} \tag{3.2}$$

$$\hat{y}_n(i) = \left\{ \begin{array}{ll} +1 & f_n(\mathbf{x}(i)) = \mathbf{w}_n^T \mathbf{x}(i) \geq 0 \\ -1 & \text{otherwise} \end{array} \right\} \tag{3.3}$$

For a test example with feature vector $\mathbf{x}(i)$, the top-down classifier predicts the class label $\hat{y}(i) \in \mathcal{L}$ as shown in eq. (3.4), where $\mathcal{C}(p)$ denotes the set of children of node p. Essentially, the algorithm starts at the root node and recursively selects the best child nodes till it reaches a terminal node belonging to the set of leaf categories $\mathcal{L}$.

$$\hat{y}_i = \left\{ \begin{array}{l} \textbf{initialize} \quad p := \textit{root} \\ \textbf{while } p \notin \mathcal{L} \\ \quad p := \textbf{argmax}_{q \in \mathcal{C}(p)} \ f_q(\mathbf{x}(i)) \\ \textbf{return } p \end{array} \right\} \tag{3.4}$$

3.3.2 Inconsistent Node Flattening

Motivation Gao *et al.* [46] showed that for any classifier that correctly classifies m random input-output pairs using a set of $\mathcal{D}$ decision nodes, the generalization error bound with probability estimates greater than 1 - ζ is less than the expression shown in eq. (3.5).

$$\frac{\delta r^2}{m}\left[\sum_{n\in\mathcal{D}}(\frac{1}{\gamma_n^2})\log(4em)\log(4m)+|\mathcal{D}|\log(2m)-\log(\frac{2}{\zeta})\right] \tag{3.5}$$

where γ_n denotes the margin at node $n \in \mathcal{D}$, δ is a constant term and r is the radius of the ball containing the distribution's support.

This provides two significant strategies in designing our approach to reduce the generalization error: (i) Increase the margin γ_n for learned models at node $n \in \mathcal{N}$ in the hierarchy, or (ii) Decrease the number of decision nodes $|\mathcal{D}|$ involved in making the prediction. For achieving the optimum classification performance, we need to balance the trade-off between the margin γ_n and the number of decision nodes $|\mathcal{D}|$. Two of the extreme cases for learning hierarchical classifiers are flat and top-down methods. For flat classifiers, we have to make single decision (*i.e.*, $|\mathcal{D}| = 1$) but margin width γ_n is presumably small due to the large number of leaf categories that needs to be distiguished, which makes it difficult to obtain large margin. For top-down hierarchical classifiers, we have to make a series of decisions from root to leaf nodes (*i.e.*, $|\mathcal{D}| \geq 1$) but margin γ_n is larger due to the fewer number of categories that needs to be distinguished at each of the decision nodes. Motivated by this trade-off, in this work we propose a method that removes some of the inconsistent nodes in the hierarchy $\mathcal{H}$, and thereby, increasing the value of margin γ_n for learned models at node n in the hierarchy, while minimizing the number of decision nodes to classify an unlabeled test instances.

In order to improve the effectiveness of classification we need to identify these inconsistent nodes and flatten them. We mark a node n within the hierarchy as inconsistent if the value of the objective function f_n^* becomes greater than a chosen threshold value. To get a

Algorithm 1: Level-wise Inconsistent Node Removal

```
Data: Original Hierarchy $\mathcal{H}$, input-output $(x_i, y_i)$
Result: Modified Hierarchy $\mathcal{H}_L$
Train $l2$-regularized LR model in a top-down order
/* Set of inconsistent node, initially empty */
$\mathcal{I}_L := \Phi$;
for $k$ := 1 ... end_level do
    /* Set of all nodes $f^*$ values in the level */
    $\mathcal{S}_k := \Phi$;
    for $n \in \mathcal{N}_k$ do
        $\mathcal{S}_k := \mathcal{S}_k \cup \{f_n^*\}$;
    end
    $\tau_k := \mu(\mathcal{S}_k) + \psi_k \sigma(\mathcal{S}_k)$;
    /* Identify inconsistent node in level */
    for $n \in \mathcal{N}_k$ do
        if $(f_n^* > \tau_k$ & $n \notin \mathcal{L})$ then
            $\mathcal{I}_L := \mathcal{I}_L \cup \{n\}$;
        end
    end
end
/* New hierarchy with inconsistent node(s) removed */
$\mathcal{H}_L = \mathcal{H}$ - $\{\mathcal{I}_L\}$;
return $\mathcal{H}_L$
```

more reliable estimate of the f_n^*, we first train the regularized LR models on a training set locally for each node and then compute the objective function on a separated validation set, which is different from the training set. The objective value on validation set for node n is denoted by f_n^*. We develop the following approaches for setting the threshold for flattening.

Level-wise Inconsistent Node Flattening: In this approach, referred as Level-INF, we select a different threshold τ_k locally for each level k in the hierarchy. Algorithm 1 presents the level-wise approach that selects inconsistent nodes at each level in a top-down manner. The threshold τ_k for level k is computed as the sum of mean and ψ_k times the standard deviation of the set of values $\{f_n^*\}_{n \in \mathcal{N}_k}$, where ψ_k is a fitness parameter at level k that is empirically estimated for each dataset based on $\{f_n^*\}_{n \in \mathcal{N}_k}$ values (see Section 3.6.2) and $\mathcal{N}_k$ represents the set of nodes in level k. All nodes $n \in \mathcal{N}_k$ that satisfy $f_n^* > \tau_k$ are

Algorithm 2: Global Inconsistent Node Removal

```
Data: Original Hierarchy $\mathcal{H}$, input-output $(x_i, y_i)$
Result: Modified Hierarchy $\mathcal{H}_G$
Train $l2$-regularized LR model in a top-down order
/* Set of inconsistent node, initially empty */
$\mathcal{I}_G := \Phi;$
/* Set of all nodes (except root) $f^*$ values in $\mathcal{H}$ */
$\mathcal{S} := \Phi;$
for $n \in \mathcal{N}$ do
    $\mathcal{S} := \mathcal{S} \cup \{f_n^*\};$
end
$\tau := \mu(\mathcal{S}) + \psi\sigma(\mathcal{S});$
/* Identify inconsistent node in $\mathcal{H}$ */
for $n \in \mathcal{N}$ do
    if ($f_n^* > \tau$ & $n \notin \mathcal{L}$) then
        $\mathcal{I}_G := \mathcal{I}_G \cup \{n\};$
    end
end
/* New hierarchy with inconsistent node(s) removed */
$\mathcal{H}_G = \mathcal{H}$ - $\{\mathcal{I}_G\};$
return $\mathcal{H}_G$
```

marked as inconsistent and added to the set of inconsistent nodes denoted by $\mathcal{I}_L$. This procedure is repeated for all levels of the hierarchy. Finally, we flatten the nodes in set $\mathcal{I}_L$ — remove $n \in \mathcal{I}_L$ and corresponding edges, and add edges from children of n to n's parent node. The modified hierarchy thus obtained is denoted by $\mathcal{H}_L$. Using the modified hierarchy, we re-train a top-down classifier.

Global Inconsistent Node Flattening: Different from Level-INF approach, which sets different thresholds for each level, the global method shown in Algorithm 2 computes a single threshold value for all levels. The threshold τ is computed as the sum of mean and ψ times the standard deviation of the set of value $\{f_n^*\}_{n \in \mathcal{N}}$, where ψ is a fitness parameter that is empirically estimated for dataset considering all $\mathcal{N}$ nodes f_n^* values. τ is used to identify the set of inconsistent nodes $\mathcal{I}_G$ in the hierarchy (*i.e.*, all nodes n with $f_n^* > \tau$). The hierarchy obtained by flattening the nodes present in $\mathcal{I}_G$ is denoted by $\mathcal{H}_G$. Using the

Table 3.2: Dataset statistics

Name	#Total Node	#Leaf Node	Depth	#Training	#Testing	#Features
CLEF	88	63	4	10,000	1,006	80
DIATOMS	399	311	4	1,940	993	371
IPC	553	451	4	46,324	28,926	1,123,497
DMOZ-SMALL	2,388	1,139	6	6,323	1,858	51,033
DMOZ-2010	17,222	12,294	6	128,710	34,880	381,580
DMOZ-2012	13,963	11,947	6	383,408	103,435	348,548

modified hierarchy, we re-train a top-down classifier. In this work we refer to this approach as Global-INF.

3.4 Experimental Protocol

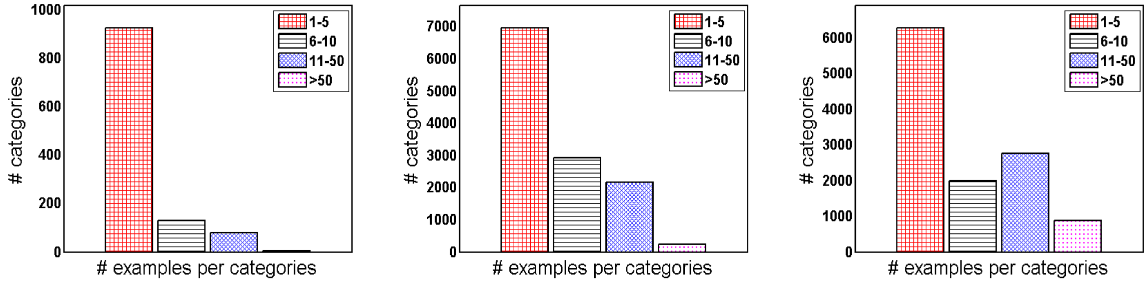
3.4.1 Datasets

We have used text and image datasets for evaluating the performance of our proposed approaches. Various statistics of the datasets used in our experiments are listed in Table 3.2. All these datasets are single-labeled and the examples are mandatorily assigned to the leaf nodes in the hierarchy (although our proposed approaches is trivially extendable to datasets with multi-label and non-mandatory leaf node label assignments). For all text datasets, we have applied the tf-idf transformation with $l2$-norm normalization to the word-frequency feature vector. Description of the used data is as follows:

Image Datasets

CLEF [48] Medical images annotated with medical applications codes. Each image is represented by the 80 features that are extracted using local distribution of edges.

DIATOMS [49] Diatom images that was created as the part of the ADIAC project. Features for each image is created using various feature extraction techniques mentioned in [49]. Further, we have preprocessed the original dataset by removing the examples that



(a) DMOZ-SMALL (b) DMOZ-2010 (c) DMOZ-2012

Figure 3.2: Distribution of DMOZ datasets visualizing majority of the classes with rare categories (marked in red and black).

belongs to the internal nodes.

Text Datasets

IPC[1] Collection of patent documents organized in international patent classification hierarchy.

DMOZ-SMALL, DMOZ-2010 and DMOZ-2012[2] Multiple web documents organized into various classes using the hierarchical structure. It is subset of the web pages from open directory project and has been released as the part of the LSHTC[3] challenge. For evaluating the DMOZ-2010 and DMOZ-2012 datasets we have used the provided test split and prediction scores are obtained using the web-portal interface[4,5] that was used during the competition.

3.4.2 Experimental Details

In all the experiments, we have divided the training dataset into train and small validation dataset in the ratio 90:10. Each experiment was run five times with different sets of train

[1] http://www.wipo.int/classifications/ipc/en/
[2] http://dmoz.org
[3] http://lshtc.iit.demokritos.gr/
[4] http://lshtc.iit.demokritos.gr/node/81
[5] http://lshtc.iit.demokritos.gr/LSHTC3_oracleUpload

and validation split chosen randomly. Testing is done on an independent held-out dataset as provided by these benchmarks. The model is trained by choosing mis-classification penalty parameter (C) in the set [10^{-3}, 10^{-2}, 10^{-1}, 1, 10^{1}, 10^{2}, 10^{3}]. The best parameter is selected using a validation set. The best parameters are used to re-train the models on the entire training set and the performance is measured on a held-out test set. For the INF methods, we compute and save the f_n^* value for each node in the hierarchy using a validation set. Setting the threshold as $\mu + \psi\sigma$ (or $\mu + \psi_k\sigma$ for k-th level in Level-INF approach), we remove the inconsistent nodes where best value of fitness parameter ψ (or ψ_k) is computed empirically for each dataset (see Section 3.6.2). All experiments were conducted using a modified version of liblinear[6] software [50] and were run on ARGO, a research computing cluster provided by the Office of Research Computing (URL: http://orc.gmu.edu), at George Mason University, VA.

3.5 Comparative Approaches

3.5.1 Flat Methods

Logistic Regression (LR) We train binary one-versus-rest regularized LR classifiers for each of the leaf categories, ignoring the hierarchical structure. The prediction decision $\hat{y}$ for unlabeled test instance $\mathbf{x}$ is based on the maximum prediction score achieved when compared across the one-versus-rest classifiers as shown in eq. (3.6).

$$\hat{y} = \underset{n \,\in\, \mathcal{L}}{\mathbf{argmax}}\; \mathbf{w}_n^T \mathbf{x} \tag{3.6}$$

Error Correcting Output Codes (ECOC) [51] This approach combines binary classifiers to exploit correlations and correct errors. Codewords are generated randomly with bits assigned for representing the hierarchical information between the categories. Experiments were done with codeword length varying from 32 to 1024 bits depending on the dataset. For

[6] http://www.csie.ntu.edu.tw/∼cjlin/liblinear/

testing an unlabeled example, the output codeword is compared to the codeword of each categories, and the one with the minimum hamming distance is selected as the class label for that example.

3.5.2 Top-Down (TD) Hierarchical Methods

For all TD hierarchical baselines we train a binary one-vs-rest classifiers for each of the node (except root) in the hierarchy and predictions are made starting from the root node and recursively selecting the best scoring child nodes until a leaf node is reached (see eq. (3.4)). Depending upon the hierarchy that we use during the training and prediction process, we compare with the following baselines.

Top-Down Logistic Regression (TD-LR) Original hierarchy provided by the domain experts is used for classifiers training and label prediction.

Level flattening Modified hierarchy obtained by flattening different level(s) is used instead of original hierarchy. Depending on level(s) flattened we have Top Level Flattening (TLF), Bottom Level Flattening (BLF), Multiple Level Flattening (MLF) hierarchy as discussed in Section 3.2.1.

Maximum-margin based Taxonomy Adaptation (MTA) Original hierarchy is modified using the margin value computed at each node in the hierarchy as described in Babbar *et al.* [30].

3.6 Results

3.6.1 Comparison to Top-Down Hierarchical Baselines

Performance based on Flat Metrics: Table 3.3 presents the μF_1 and MF_1 performance comparison of our proposed hierarchical modification approaches with TD-LR (involves no

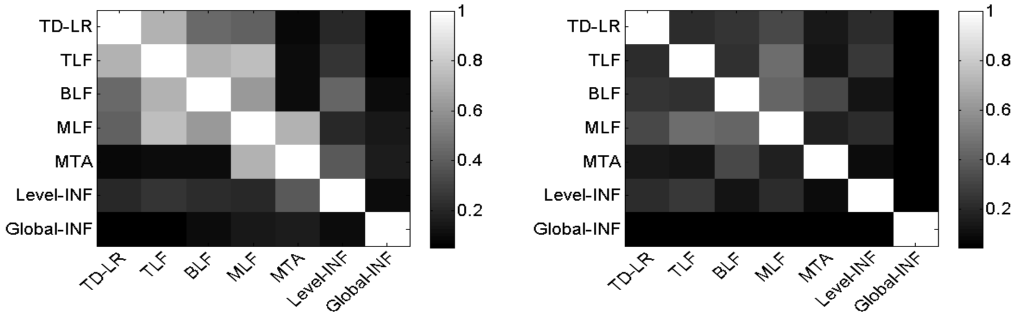
Table 3.3: μF_1 and MF_1 performance comparison of various TD hierarchical baselines against our proposed approaches

Dataset	Metric	Hierarchical Baselines					Proposed Approaches	
		TD-LR	TLF	BLF	MLF	MTA	Level-INF	Global-INF
CLEF	$\mu F_1(\uparrow)$	72.74 (0.43)	75.84 (0.32)	73.76 (0.32)	X	74.48 (0.42)	75.25 (0.55)	**77.14**△ **(0.01)**
	$MF_1(\uparrow)$	35.92 (0.01)	38.45 (0.65)	40.93 (0.19)	X	39.53 (0.74)	39.89 (0.24)	**46.54▲ (0.01)**
DIATOMS	$\mu F_1(\uparrow)$	53.27 (0.32)	56.93 (0.28)	53.27 (0.24)	X	58.36 (0.64)	58.32 (0.64)	**61.31▲ (0.53)**
	$MF_1(\uparrow)$	44.46 (0.24)	45.17 (0.62)	44.30 (0.64)	X	45.21 (0.65)	48.77 (0.12)	**51.85▲ (0.23)**
IPC	$\mu F_1(\uparrow)$	49.32 (0.32)	51.28 (0.61)	50.36 (0.64)	X	51.36 (0.32)	50.40 (0.32)	**52.30**△ **(0.12)**
	$MF_1(\uparrow)$	42.51 (0.94)	44.99 (0.43)	43.74 (0.81)	X	42.80 (0.94)	43.26 (0.43)	**45.65**△ **(0.11)**
DMOZ-SMALL	$\mu F_1(\uparrow)$	45.10 (0.23)	45.48 (0.19)	44.34 (0.32)	45.80 (0.64)	46.01 (0.74)	45.43 (0.21)	**46.61**△ **(0.28)**
	$MF_1(\uparrow)$	30.65 (0.43)	30.60 (0.54)	30.94 (0.53)	30.62 (0.32)	30.82 (0.63)	30.34 (0.12)	**31.86▲ (0.64)**
DMOZ-2010	$\mu F_1(\uparrow)$	40.22 (0.55)	41.32 (0.32)	40.34 (0.24)	41.77 (0.56)	41.82 (0.42)	40.71 (0.83)	**42.37 (0.27)**
	$MF_1(\uparrow)$	28.37 (0.46)	29.05 (0.84)	28.41 (0.57)	29.11 (0.13)	29.18 (0.54)	28.66 (0.53)	**30.41 (0.64)**
DMOZ-2012	$\mu F_1(\uparrow)$	50.13 (0.28)	50.32 (0.42)	50.11 (0.32)	48.05 (0.39)	50.31 (0.48)	49.90 (0.92)	**50.64 (0.22)**
	$MF_1(\uparrow)$	29.89 (0.23)	29.89 (0.23)	29.73 (0.14)	27.65 (0.48)	30.04 (0.57)	30.52 (0.74)	**30.58 (0.28)**

Table shows mean and (standard deviation) in bracket across five runs. 'X' denotes MLF not possible. The significance-test results are denoted as △ for a p-value less than 0.05 and ▲ for p-value less than 0.01. We have used sign-test and wilcoxon rank test for statistical evaluation of μF_1 and MF_1 scores, respectively. Tests are between our best proposed approach, Global-INF and best baseline approach, MTA for single run. These statistical tests are not performed on DMOZ-2010 and DMOZ-2012 datasets because we do not have access to true labels from the online evaluation system.

hierarchy modification) and comparative TD hierarchy modification approaches as baselines. We see that our proposed approach Global-INF consistently outperforms all other approaches for the different datasets across all metrics. For the image datasets we see a relative performance improvement upto 7% in MF_1 on comparing Global-INF with the best TD modification baseline *i.e.*, MTA. To validate the performance improvement we conducted pairwise statistical significance tests between our best approach, Global-INF and best TD baseline for all datasets except DMOZ-2010 and DMOZ-2012, where true test labels (and class-wise performance) are not available from the online evaluation. Specifically, we compute sign-test for μF_1 [52] and non-parametric wilcoxon rank test for MF_1 scores (it should be noted that significance tests are between two approaches for single run). In Figure 3.3 we present the pairwise statistical comparisons for different approaches studied here on the DMOZ-SMALL dataset. The Global-INF consistently outperforms other baseline approaches studied here.

Overall, results of statistical tests shows that Global-INF approach significantly outperforms the best baseline (and hence other baselines) for all the datasets (see Table 3.3).



(a) μF_1

(b) MF_1

Figure 3.3: P-values comparison of different approaches for DMOZ-SMALL dataset. Significant improvements are denoted by dark black color.

On comparing our two proposed approaches – Global-INF has better performance over Level-INF. This is because the Level-INF approach strictly enforces some of the nodes to be flattened from each levels although there f_n^* value may be much lower than the other nodes at different levels in the hierarchy and vice-versa. In contrast, Global-INF approach takes all nodes f_n^* values into consideration while making a decision and hence it determines a better set of inconsistent nodes. MTA approach has poor performance due to the similar issues as with Level-INF approach. Performance of level flattening approaches, viz., TLF, BLF and MLF, suffers because these methods remove the entire level(s) in the hierarchy and do not take into consideration whether any node in that level is important for HC. TD-LR approach has the worst performance because of the inconsistent nodes present in the original hierarchy that are negatively impacting the generalization capabilities of learned models at the higher levels (see Section 3.6.3), which results in error propogation.

Performance based on Hierarchical Metrics: Hierarchical evaluation metrics hF_1 and TE compute errors for misclassified examples based on the definition of a defined hierarchy. As such, Table 3.4 presents the hF_1 and TE score for all TD approaches evaluated over the original hierarchy and the modified hierarchy (obtained by flattening). We can see that our proposed approach, Global-INF outperforms other approaches because it is able to identify a better set of inconsistent nodes. On comparing the classification performance

Table 3.4: Hierarchical performance comparison of various TD hierarchical baselines against our proposed approaches over original (experts defined) and modified hierarchy

Dataset	**Hier.**	**Hierarchical Baselines**					**Proposed Approaches**	
		TD-LR	**TLF**	**BLF**	**MLF**	**MTA**	**Level-INF**	**Global-INF**
		hF_1 **score** (↑)						
CLEF	Original	74.52 (0.01)	78.24 (0.75)	75.13 (0.46)	X	76.01 (0.74)	76.81 (0.59)	**79.06 (0.01)**
	Modified	-	77.78 (0.65)	78.08 (0.13)	X	77.50 (0.23)	78.28 (0.24)	**80.87 (0.13)**
DIATOMS	Original	56.15 (0.21)	62.53 (0.43)	56.14 (0.17)	X	59.60 (0.28)	60.03 (0.24)	**62.80 (0.04)**
	Modified	-	63.38 (0.24)	57.02 (0.62)	X	59.70 (0.14)	59.98 (0.28)	**63.88 (0.13)**
IPC	Original	62.57 (0.32)	64.39 (0.38)	63.00 (0.10)	X	63.42 (0.54)	63.26 (0.34)	**64.73 (0.12)**
	Modified	-	65.48 (0.32)	63.24 (0.41)	X	63.14 (0.54)	62.52 (0.38)	**66.29 (0.28)**
DMOZ-SMALL	Original	63.14 (0.54)	63.17 (0.43)	63.26 (0.52)	63.32 (0.64)	63.20 (0.54)	61.98 (0.56)	**63.37 (0.44)**
	Modified	-	64.32 (0.50)	63.94 (0.38)	63.39 (0.19)	63.82 (0.42)	58.02 (0.14)	**64.97 (0.75)**
DMOZ-2012	Original	73.04 (0.21)	72.70 (0.17)	73.04 (0.28)	70.49 (0.03)	73.03 (0.11)	71.41 (0.38)	**73.19 (0.02)**
		TE **score** (↓)						
CLEF	Original	1.26 (0.01)	1.08 (0.08)	1.23 (0.03)	X	1.13 (0.09)	1.15 (0.05)	**1.04 (0.03)**
	Modified	-	0.89 (0.07)	0.88 (0.04)	X	0.90 (0.04)	0.94 (0.01)	**0.71 (0.09)**
DIATOMS	Original	1.76 (0.01)	1.49 (0.01)	1.76 (0.03)	X	1.60 (0.03)	1.60 (0.06)	**1.49 (0.02)**
	Modified	-	1.28 (0.02)	1.32 (0.02)	X	1.14 (0.06)	1.16 (0.02)	**1.08 (0.08)**
IPC	Original	2.23 (0.02)	2.12 (0.04)	2.20 (0.01)	X	2.22 (0.06)	2.19 (0.01)	**2.10 (0.02)**
	Modified	-	1.64 (0.01)	1.58 (0.04)	X	1.80 (0.06)	1.83 (0.03)	**1.38 (0.02)**
DMOZ-SMALL	Original	3.55 (0.04)	3.55 (0.02)	3.53 (0.03)	3.53 (0.06)	3.51 (0.08)	3.65 (0.06)	**3.50 (0.02)**
	Modified	-	2.96 (0.05)	2.90 (0.01)	2.62 (0.03)	2.68 (0.03)	2.82 (0.02)	**2.37 (0.03)**
DMOZ-2010	Original	3.69 (0.03)	3.58 (0.01)	3.68 (0.10)	3.56 (0.08)	3.61 (0.02)	3.74 (0.04)	**3.53 (0.01)**

Table shows mean and (standard deviation) in bracket across five runs. 'X' denotes MLF not possible. Evaluations for DMOZ-2010 and DMOZ-2012 datasets cannot be performed on new hierarchy as it is not supported by the web-portal. Further, hF_1 for DMOZ-2010 and TE score for DMOZ-2012 dataset is not available from the online evaluation system.

over the original hierarchy and the modified hierarchy, we can see that for most of the approaches classification on modified hierarchy shows an improved performance. This is because flattening of hierarchies results in the reduction of hierarchical path length for mis-classified examples contributing to performance improvement.

3.6.2 Empirical Study for Threshold (τ) Selection

Figure 3.4 shows the MF_1 performance comparison of flat LR approach against our best TD approach, Global-INF with varying selection of threshold (τ) in the interval [μ, $\mu + 3\sigma$] (performance deteriorates after $\mu + 3\sigma$) with step-size 0.1σ for CLEF and DMOZ-SMALL datasets. We choose these datasets for evaluation because they have different data characteristics. The CLEF dataset is well balanced and does not suffer from the rare categories issue, whereas DMOZ-SMALL dataset is highly imbalanced and majority of the classes belong to rare categories (*i.e.*, having $\leq$ 10 examples) as shown in Figure 3.2. In order

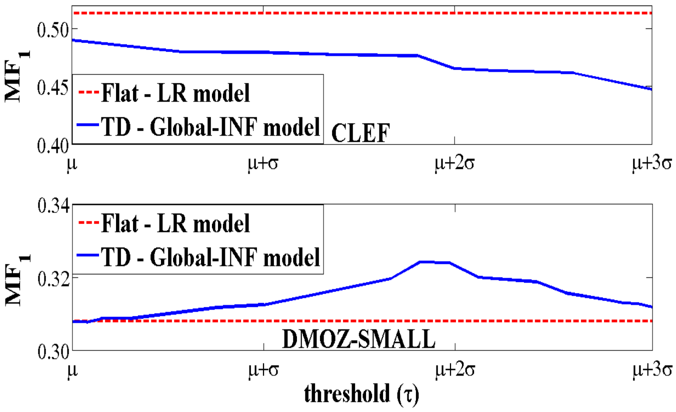
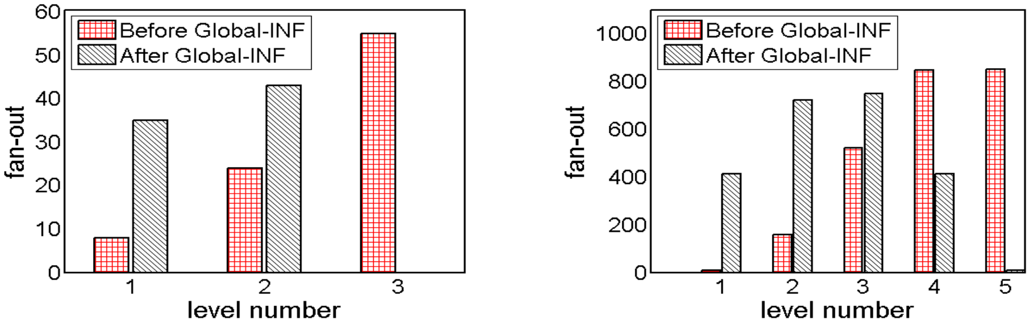


Figure 3.4: MF_1 performance comparison of flat LR approach (marked in dotted red) against best TD approach, Global-INF (marked in solid blue) with different selection of threshold (τ) for CLEF and DMOZ-SMALL datasets. Validation data is used for plotting the graph.

to identify the set of inconsistent nodes in the hierarchy, we compare the computed f_n^* value of each internal node with the chosen threshold (τ) and mark the node as inconsistent iff $f_n^* > \tau$. It can be seen from the figure that for CLEF dataset, performance improves as the threshold (τ) decreases giving intuition that τ should be kept smaller *i.e.*, removing more internal nodes from the hierarchy (enforcing flat structure) is better and hence reducing the threshold value τ can possibly lead to better results. However, for the DMOZ-SMALL dataset, performance first increases and than decreases with maximum performance achieved at $\tau = \mu + 1.8\sigma$. This behavior suggests that for imbalanced data distribution with potentially large number of rare categories, we should generally keep the threshold higher. It helps to leverage the hierarchical information while reducing error propagation by removing inconsistent nodes. The best threshold for a specific dataset can be chosen empirically using a small validation set as done in this study.

To further understand the behavior of modified hierarchy using Global-INF approach, we analyzed the datasets in terms of level-wise fan-out (# children) in the hierarchy, before and after removing inconsistent nodes. We can see from the Figure 3.5 that with both datasets maximum flattening take place at higher levels in the hierarchy, which results in increased fan-out value. Reason for inconsistencies at higher levels is the presence of many



(a) CLEF

(b) DMOZ-SMALL

Figure 3.5: Total fan-out (# children) at each level for CLEF and DMOZ-SMALL datasets before and after flattening inconsistent nodes using Global-INF approach.

Table 3.5: Level-wise error for TD-LR and Global-INF approach

Dataset	Level	TD-LR		Global-INF	
	no.	error (↓)	# ME	error (↓)	# ME
CLEF	L-1	21.27 (0.63)	214	20.18 (0.26)	203
	L-2	07.71 (0.42)	240	07.34 (0.29)	224
	L-3	11.30 (0.16)	274	05.66 (0.13)	227
DMOZ-SMALL	L-1	42.47 (0.32)	789	39.83 (0.17)	740
	L-2	14.45 (0.62)	921	12.91 (0.21)	855
	L-3	15.14 (0.34)	972	17.99 (0.14)	968
	L-4	12.32 (0.02)	1001	07.57 (0.10)	991
	L-5	15.66 (0.05)	1020	33.33 (0.04)	992

Table shows mean and (standard deviation) of error rate across five runs. # ME denotes the average number of misclassified examples upto that level.

dissimilar classes beneath each node, which makes it comparatively difficult to learn generalized classifiers resulting in higher f_n^* values (i.e., inconsistent nodes marked for removal).

3.6.3 Level-wise Misclassification Error

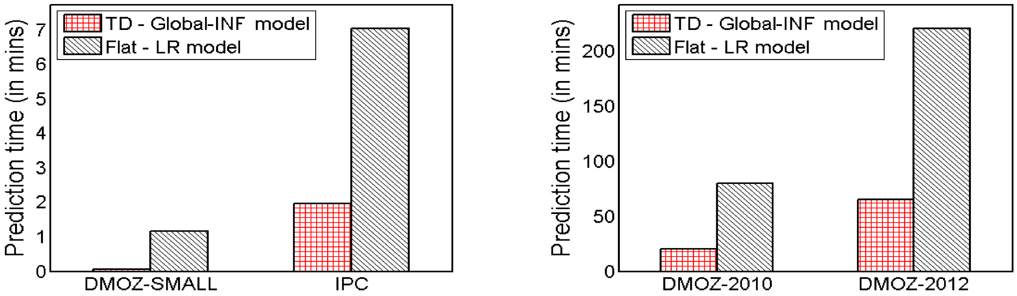
Table 3.5 shows the level-wise error analysis that is obtained for TD-LR and our best approach, Global-INF for CLEF and DMOZ-SMALL datasets. We can see that at higher levels, the Global-INF approach misclassifies fewer examples (shown in #ME column) that results in less error propagation down the levels, and hence better overall performance.

Table 3.6: MF_1 and hF_1 performance comparison between Global-INF and flat baselines with varying distribution of training examples per class.

Dataset	**# Train**	**Best Proposed**		**Flat Baselines**			
	example	**Global-INF**		**LR**		**ECOC**	
	per class	MF_1	hF_1	MF_1	hF_1	MF_1	hF_1
CLEF	6-10	**36.12**	**66.45**	35.14	62.34	36.16	63.45
	11-50	45.52	76.13	**45.92**	**77.04**	45.36	76.40
	>50	52.24	84.12	**55.92**	**87.24**	53.24	85.90
	avg.	46.99	79.00	**51.31▲**	**80.58**	50.02	79.34
DIATOMS	6-10	**41.08**	**48.92**	38.92	46.32	39.14	47.44
	11-50	42.82	**62.72**	**44.18**	62.45	43.82	59.26
	>50	53.17	64.10	**57.24**	**68.10**	52.21	63.28
	avg.	51.85	62.80	**54.17▲**	**63.50**	48.82	61.92
IPC	11-50	43.64	**62.45**	**43.94**	60.92	42.98	60.01
	>50	47.28	68.70	**49.44**	**68.95**	47.21	67.96
	avg.	45.65	**65.73**	**45.74**	64.00	44.65	61.84
DMOZ-SMALL	≤5	**28.77**	**51.86**	27.02	46.81	27.12	47.35
	6-10	**55.55**	**67.47**	54.76	65.40	54.18	63.28
	11-50	72.26	78.74	**72.60**	**80.12**	72.02	78.53
	>50	69.43	86.70	**71.44**	**88.95**	69.80	85.89
	avg.	**31.86△**	**63.37**	30.80	60.87	30.10	60.50
DMOZ-2010	≤5	**18.23**	**53.59**	14.35	48.13	10.46	47.47
	6-10	**23.03**	**55.76**	22.62	51.84	20.74	49.63
	11-50	42.56	**62.39**	**43.26**	61.85	41.92	60.44
	>50	70.74	77.51	**73.20**	**81.51**	68.92	77.18
	avg.	**28.41△**	**56.17**	27.06	53.94	26.12	49.24
DMOZ-2012	≤5	**10.28**	**50.56**	8.78	48.01	7.41	47.14
	6-10	**20.37**	**50.71**	18.84	48.82	18.32	48.26
	11-50	37.19	73.16	**37.98**	**73.24**	35.72	72.73
	>50	53.20	79.73	**55.72**	**84.92**	50.23	78.10
	avg.	**29.14▲**	**68.24**	27.04	66.45	26.64	65.10

CLEF, DIATOMS and IPC datasets does not have any categories with ≤5 examples (≤10 for IPC). The significance-test results are denoted as △ for a p-value less than 0.05 and ▲ for p-value less than 0.01. Wilcoxon rank test is used for statistical evaluation of MF_1 scores. Tests are between Global-INF and best flat, LR approach.

This experiment supports our hypothesis that Global-INF approach identifies better set of inconsistent nodes that helps in minimizing the error propagation. Results for other TD baselines are not shown in the work for brevity.



(a) Small Datasets

(b) Large Datasets

Figure 3.6: Prediction runtime comparison (in mins) between Global-INF and LR approach. Image datasets have small difference hence omitted.

3.6.4 Comparison to Flat Baselines

Table 3.6 shows the MF_1 and hF_1 performance comparison of our best approach, Global-INF against flat baselines – LR and ECOC approach. For easier analysis, we have showed the results for datasets separated by varying distribution of training size (for evaluating DMOZ-2010 and DMOZ-2012 datasets we have used a separate held out dataset because we don't know the actual labels of test dataset from the online evaluation). We show the results for MF_1 because it gives equal importance to all the classes while evaluation and hence provides better essence of the results for datasets with skewed distribution. For computing hF_1, we have used the original hierarchy for consistent evaluation. As we can see from the table, the LR approach outperforms Global-INF approach for CLEF, DIATOMS and IPC datasets because these datasets are well balanced and have smaller number of categories. However, for the DMOZ datasets, our approach Global-INF has better performance because hierarchical structure provides useful information for categorizing classes with rare categories. Within the DMOZ datasets, rare categories make up more than 75% of the classes as shown in Figure 3.2. The ECOC approach has the worst performance because the codewords used in our experiments are chosen randomly and merging of categories may require nonlinear discriminants instead of the linear classifiers used in this work.

Table 3.7: Total training runtime comparison (in mins) between Global-INF and LR approach

	CLEF	**DIATOMS**	**IPC**	**DMOZ-SMALL**	**DMOZ-2010**	**DMOZ-2012**
Global-INF	3	10	830	68	25,462	63,000
LR	1	3	658	46	15,248	46,124

3.6.5 Computational Run Time

Although, the flat LR approach outperforms Global-INF approach for some datasets in terms of classification performance, their prediction runtime is significantly higher and it can be untenable for large-scale problems [6, 44]. The prediction runtime comparison of Global-INF and LR approach is shown in Figure 3.6. As expected, Global-INF approach has comparatively lower prediction runtime (upto 4x improvement). The difference is significant for large-scale datasets (DMOZ-2010 and DMOZ-2012). For completeness, we also report the total training runtime in Table 3.7. The Global-INF approach has higher training runtime due to the overhead involved with classifiers re-training after hierarchy modification and also involves training one-vs-rest binary classifiers for internal nodes in addition to leaf categories. Nevertheless, both flat and TD approaches are trivially parallelizable due to decoupling (*i.e.*, no interactions) between the classifiers learnt at different nodes n in the hierarchy. For reporting training runtime, we trained classifiers in parallel across multiple compute nodes in the cluster and sum up the time taken at each node. In our experiments, we choose expensive one-vs-rest binary classifiers over comparably cheaper one-vs-sibling binary classifiers because our preliminary experiments showed better results with one-vs-rest approach. It should also be noted that there is no significant difference between the prediction and training runtime of different TD approaches, and hence we do not report them here.

3.7 Summary

In this work we proposed two different approaches for hierarchy modification that restructures the hierarchy by flattening most prominent set of inconsistent nodes, thereby improving the hierarchy representation which is more suited for HC. Performance evaluation on wide range of datasets over the proposed modified hierarchy shows improved classification results because fewer examples are misclassified at higher levels, resulting in less error propagation. Comparison of our proposed approach with the competitive hierarchy modification approaches in the literature showed significant performance improvement supporting the hypothesis that our approach identifies the better set of inconsistent nodes. We also performed experiments to compare our approach with the flat approach with varying distribution of training examples per categories. Results demonstrated the usefulness of leveraging hierarchical information for classifying classes with fewer training examples.

Chapter 4: Improving Large-scale Hierarchical Classification by Rewiring

4.1 Introduction

Top-down HC methods that leverage the hierarchy during the learning and prediction process are effective approaches to deal with large-scale problems [36]. Classification decision for top-down methods involves invoking only the models in the relevant path within the hierarchy. Though computationally efficient, these methods have higher number of misclassifications due to error propagation [2]. For several benchmarks, the HC approaches are outperformed by *flat* classifiers that ignore the hierarchy [20, 39]. In majority of the cases, the hierarchy available for training classifiers is manually designed by experts based on domain knowledge and is not consistent for classification. In order to improve performance, we need to *restructure* the hierarchy to make it more favorable and useful for classification. Motivated by this idea, our main focus in this work is on generating an improved representation from the expert-defined hierarchy. To summarize, our contributions are as follows:

- We propose an efficient data-driven filter based rewiring approach for hierarchy modification which unlike previous wrapper based approaches [27, 29] does not require multiple, expensive computations. Our approach is scalable and can be applied to the HC problems with high-dimensional features, large number of classes and examples.
- We perform extensive empirical evaluations and case studies to show the strengths of our approach in comparison to other hierarchy modification approaches such as clustering and flattening.
- The modified hierarchy can be used with any hierarchical classification approaches like top-down HC or state-of-the-art approaches incorporating hierarchical relationships

[53]. The modified hierarchy in conjunction with a scalable Top-Down HC approach outperforms the flat classifiers on ~65% of the rare categories (i.e., classes with less than 10 training examples) across the DMOZ datasets (See Section 4.5.5).

4.2 Related Work

Our work is closely related to the rewiring approach developed in Tang *et al.* [27], where the expert-defined hierarchy is gradually modified. Iteratively, a subset of the hierarchy is modified and evaluated for classification performance improvement using the HC learning algorithm. Modified changes are retained if the performance results improve; otherwise the changes are discarded and the process is repeated. This repeated procedure of hierarchy modification continues until the optimal hierarchy is reached. Expensive evaluation at each step makes this approach intractable for large-scale datasets. Another drawback of this approach is deciding which branch of the hierarchy to explore first (for modification) and which elementary operation (promote, demote, merge) to apply at each step. Other work in similar direction can be found in [28, 29].

Earlier studies focused on flattening based approaches where some level or nodes are selectively flattened (removed) based on certain criterion [30–32]. In other work, learning based approach have been proposed [4], where nodes to flatten are decided based on classification performance improvement on a validation set. This approach although useful for smaller datasets, is not scalable due to the expensive evaluation process after each node removal. Recently, Naik *et al.* [2] proposed a taxonomy adaptation where some nodes are intelligently flattened based on empirically defined cut-off threshold and objective function values computed at each node. Hierarchy modification using this approach is scalable and beneficial for classification and has been theoretically justified [46].

Other approaches towards hierarchy modification involves generating hierarchy from scratch, ignoring the expert-defined hierarchy. These approaches exploit hierarchical clustering algorithms for generating the hierarchy [1, 54–56]. Constructing hierarchy using clustering approaches is not popular due to its sensitivity to predefined parameters such as number of levels.

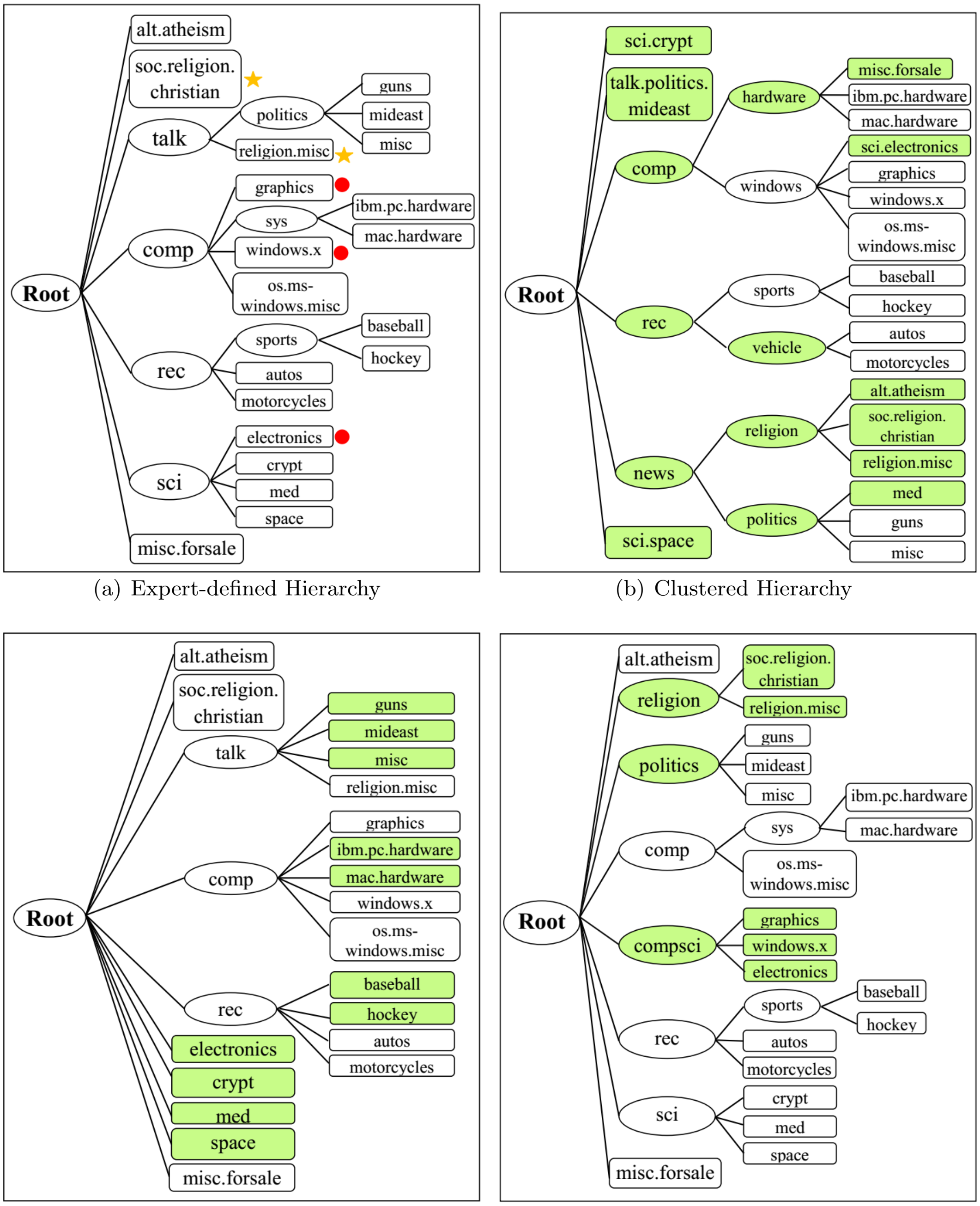
4.3 Methods

4.3.1 Motivation

The manual process of hierarchy creation suffers from various issues. Specifically, (i) Hierarchies are generated by grouping semantically similar categories under a common parent category. However, many different semantically sound hierarchies may exist for same set of classes. For example, in categorizing products, the experts may generate a hierarchy by first separating products based on the company name (*e.g.*, Apple, Microsoft) and then the product type (*e.g.*, phone, tablet) or vice-versa. Both hierarchies are equally good from the perspective of an expert. However, these different hierarchies may lead to different classification results. (ii) Apriori it is not clear to domain experts when to generate new nodes (hierarchy expansion) or merge two or more nodes (link creation) while creating hierarchies, resulting in a certain degree of arbitrariness. (iii) A large number of categories pose a challenge for the manual design of a consistent hierarchy. (iv) Dynamic changes may require hierarchical restructuring.

To remove inconsistencies, various approaches for hierarchy modification have been proposed. These approaches can be broadly categorized into two categories: (i) Flattening approaches [2,4,30–32] where some of the identified inconsistent nodes (based on error rate, classification margins) are flattened (removed) and (ii) Rewiring approaches [27–29] where parent-child relationships within the hierarchy are modified to improve the classification performance. Clustering based methods have also been adapted in some of these studies [1, 54] where consistent hierarchy is generated from scratch using agglomerative or divisive clustering algorithms. A summary of the various existing methods and their characteristics is shown in Table 4.1.

To understand the qualitative difference between hierarchy generated using various



(a) Expert-defined Hierarchy

(b) Clustered Hierarchy

(c) Flattened Hierachy

(d) Rewired Hierachy

Figure 4.1: (a) Expert-defined hierarchy (classes with high degree of similarities are marked with symbols ★, ●) modified using various methods: (b) Agglomerative clustering with cluster cohesion to restrict the height to original height [1] (c) Global-INF flattening method [2] (d) Proposed rewiring method. Modified structure changes are shown in green color.

Table 4.1: The summary review of existing taxonomy modification methods and their characteristics.

Modification Method	**Approach**	**Type**	**Scalable**
Margin-based modification [30]	Flattening	Filter	✓
Level flattening [31]	Flattening	Filter	✓
Inconsistent node flattening [2]	Flattening	Filter	✓
Learning based algorithm [4]	Flattening	Wrapper	×
Agglomerative clustering [1]	Clustering	Wrapper	×
Divisive clustering [54]	Clustering	Wrapper	×
Optimal hierarchy search [27]	Rewiring	Wrapper	×
Genetic based algorithm [29]	Rewiring	Wrapper	×
Our proposed approach:			
Similarity based modification	Rewiring	Filter	✓

approaches, we performed experiments on the smaller newsgroup[1] dataset containing 20 classes. Figure 4.1(b)-(d) shows the hierarchy structure obtained using clustering, flattening and rewiring based approaches, respectively. Hierarchy generated using clustering completely ignores the expert-defined hierarchy information, which contains valuable prior knowledge for classification [27]. Flattening approaches cannot group together the classes from different hierarchical branches (for *e.g*, *soc.religion.christian* and *religion.misc*). On the contrary, the rewiring approaches provide the flexibility of grouping classes from different sub-branches. More details about Figure 4.1 are discussed later in a case study (Section 4.5.1).

4.3.2 Proposed Rewiring Approach

Wrapper based approaches [27–29] modify the hierarchy by making one or few changes, which are then evaluated for classification performance improvement using the HC learning algorithm. Modified changes are retained if the performance results improve; otherwise the changes are discarded and the process is repeated. This repeated procedure of hierarchy modification continues until the optimal hierarchy that satisfies certain criteria is reached. As such, wrapper approaches are not scalable for large datasets.

[1] http://qwone.com/sim jason/20Newsgroups/

Algorithm 3: *rewHier* Algorithm

```
Data: Original Hierarchy $\mathcal{H}$, input-output $(\mathbf{x}_i, y_i)$
Result: Modified Hierarchy $\mathcal{H}_M$
/* Initialization */
$H_M = \mathcal{H}$;
/* Ist step: Grouping Similar Classes Pair */
Compute cosine similarity between all possible class pairs.
/* similar class grouping */
Identify the most similar class pairs with similarity scores value greater than empirically defined threshold
parameter $\tau$. Let $|c|$ denotes the number of such pairs represented by the set $\mathbf{S} = \{s_1, s_2, \ldots, s_{|c|}\}$, where $i$-th
pair $s_i$ is represented using $(s_i^{(1)}, s_i^{(2)})$.
/* IInd step: Inconsistency Identification and Correction */
for $i = 1 \ to \ |c|$ do
    $rewire[1] = 1$; /* check if rewiring is needed for $s_i^{(1)}$ */
    $rewire[2] = 1$; /* check if rewiring is needed for $s_i^{(2)}$ */
    /* Inconsistent pair check */
    if $\pi(s_i^{(1)}) \neq \pi(s_i^{(2)})$ then
        /* check similarity to all siblings */
        foreach $j \in \zeta(s_i^{(1)})$ do
            if $\big((j, s_i^{(2)}) \ or \ (s_i^{(2)}, j)\big) \notin \boldsymbol{S}$ then
                $rewire[2] = 0$;
                break;
            end
        end
        foreach $j \in \zeta(s_i^{(2)})$ do
            if $\big((j, s_i^{(1)}) \ or \ (s_i^{(1)}, j)\big) \notin \boldsymbol{S}$ then
                $rewire[1] = 0$;
                break;
            end
        end
        if $(rewire[1] == 0) \ and \ (rewire[2] == 0)$ then
            /* perform node creation */
            $N_{new} = \phi$ /* create new node */
            $[\mathcal{H}_M] = \mathbf{NC}(N_{new} \rightarrow lca(s_i), s_i \rightarrow N_{new}, \mathcal{H}_M)$;
            /* lca denotes lowest common ancestor */
        else
            if $(rewire[1] == 1)$ then
                $[\mathcal{H}_M] = \mathbf{PCRewire}(s_i^{(1)} \rightarrow \pi(s_i^{(2)}), \mathcal{H}_M)$;
            else
                $[\mathcal{H}_M] = \mathbf{PCRewire}(s_i^{(2)} \rightarrow \pi(s_i^{(1)}), \mathcal{H}_M)$;
            end
        end
    end
end
/* perform node deletion */
$[\mathcal{H}_M] = \mathbf{ND}(\mathcal{H}_M)$;
return $\mathcal{H}_M$
```

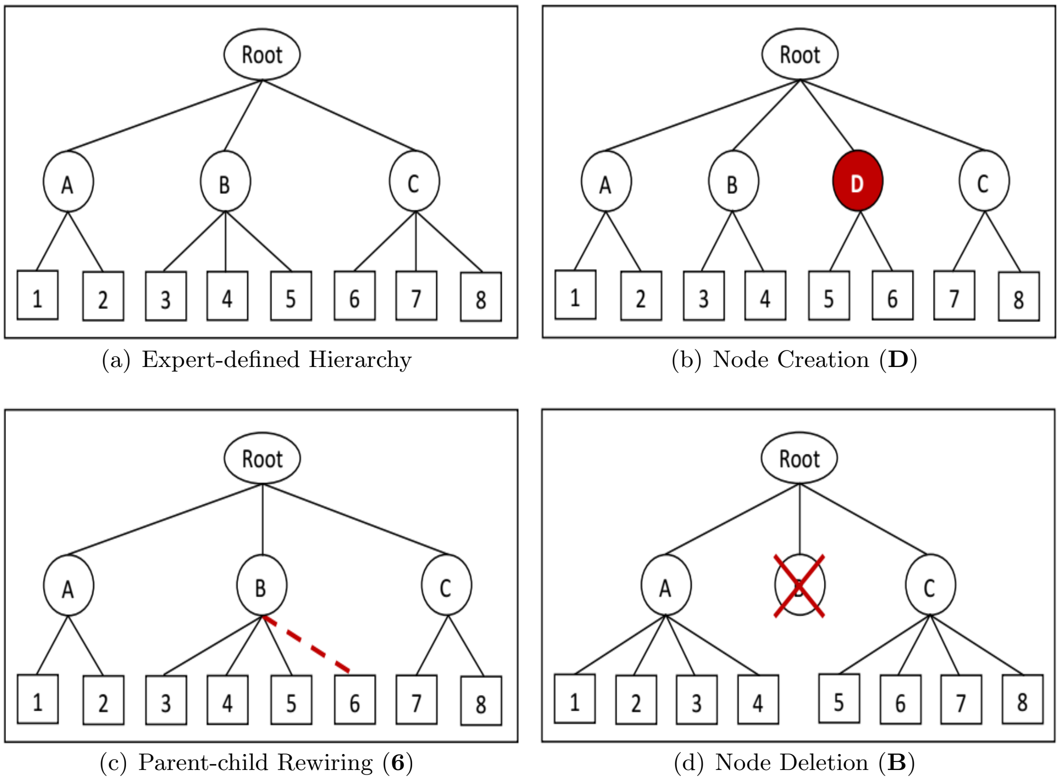
We propose an efficient data-driven filter based rewiring approach where the hierarchy is modified based on certain relevance criterion (pairwise sibling similarity) between the different classes within the hierarchy. Our approach is single step and do not require experimental evaluation for multiple iterations. We refer to our proposed rewiring approach as *rewHier*. Algorithm 3 illustrates our approach for hierarchy modification. Specifically, it consists of two steps:

(i) Grouping Similar Classes Pairs - To ensure classes with high degree of similarity are grouped together under the same parent node in the modified taxonomy, this step identifies the similar classes pairs that exist within the expert-defined hierarchy. Pairwise cosine similarity is used as the similarity measure in our experiments because it is less prone to the curse of dimensionality [57]. Once the similarity scores are computed, we determine the set $\mathbf{S}$ of most similar pairs of classes using an empirically defined cut-off threshold τ for a dataset (detailed analysis regarding τ selection is discussed in Section 4.5.4). For example, in Figure 4.1(a) this step will group together the class pairs with high similarity scores such as $\mathbf{S} = \big[($*religion.misc*, *soc.religion.christian*$)$, $($*electronics*, *windows.x*$)$, $($*electronics*, *graphics*$)$, ...$\big]$.

Pairwise similarity computation between different classes is one of the major bottlenecks of this step. To make it scalable, we distribute the similarity computation across multiple compute nodes.

(ii) Inconsistency Identification and Correction - To obtain the consistent hierarchy, we group together each of the similar class pairs to a common parent node. Iteratively, starting from the most similar class pairs we check for potential inconsistencies *i.e.*, if the pairs of classes are in different branches (sub-trees). In order to resolve the identified inconsistencies we take corrective measures using three basic elementary operations: (i) node creation, (ii) parent-child rewiring and (iii) node deletion. Figure 4.2(b)-(d) illustrates the various hierarchical structures that are obtained after the execution of these elementary operations on the expert-defined hierarchy in Figure 4.2 (a).

Node Creation (NC) - This operation groups together the identified similar class pairs in different branches (sub-trees) of the hierarchy using a new node, with parent as the lowest common ancestors of similar classes. Figure 4.2(b) illustrates this operation where



(a) Expert-defined Hierarchy

(b) Node Creation (**D**)

(c) Parent-child Rewiring (**6**)

(d) Node Deletion (**B**)

Figure 4.2: Modified hierarchical structures (b)-(d) obtained after applying elementary operations to expert-defined hierarchy ($\mathcal{H}$). Leaf nodes are marked with 'rectangle' and structural changes are shown by red color.

the similar class pairs **5** and **6** are grouped together by the newly created node **D**. This operation is used only when a *proper subset* of the leaf nodes from different branches are similar (*i.e.*, not similar to all leaf nodes in the branch; otherwise the parent-child rewiring operation is used).

Parent-child Rewiring (PCRewire) - As shown in Figure 4.2(c), this operation simply assigns (rewires) the leaf node from one parent to another parent node in the hierarchy. It is useful when the leaf node is identified to be similar to all the sibling leaf nodes within the given hierarchy branch. For example, in Figure 4.2(c), if the computed similarity score determines the leaf node **6** to be more similar to nodes **3**, **4** and **5** in comparison to its current siblings **7** and **8**, than it is more desirable from a classification perspective to assign

6 as node **B** child rather than **C**.

Node Deletion (ND) - This refers to deletion of nodes in the hierarchy that are deemed useless for classification. In Figure 4.2(d), node **B** is deleted because there are no leaf nodes that can be classified by node **B**. This operation is used as a post-processing step in our algorithm to refine the hierarchy.

The *rewHier* algorithm determines (outer *for loop*) the best corrective measures (*node creation* or *parent-child rewiring*) that need to be taken. Once all the inconsistencies have been addressed, *rewHier* calls the *node deletion* procedure as a final modification step where unnecessary nodes are deleted. The modified hierarchy obtained after inconsistencies removal can be used to train any classifier.

4.3.3 Top-Down Hierarchical Classification

We propose to use the Top-Down HC approach with our modified hierarchies because it scales well during training and prediction. Specifically, we train binary one-vs-rest classifiers for each of the nodes $n \in \mathcal{N}$ — to discriminate its positive examples from the examples of other nodes (*i.e.* negative examples) in the hierarchy. In this work, we use logistic regression (LR) as the underlying base model for training [6]. The LR objective uses logistic loss to minimize the empirical risk and squared $l2$-norm term (denoted by $|| \cdot ||_2^2$) to control model complexity and prevent overfitting. The objective function f_n for training a model corresponding to node n is provided in eq. (4.1).

$$f_n = \min_{\mathbf{w}_n} \left[C \sum_{i=1}^{N} \log \left(1 + \exp\left(-y_n(i)\mathbf{w}_n^T \mathbf{x}(i)\right)\right) + \frac{1}{2} \|\mathbf{w}_n\|_2^2 \right] \tag{4.1}$$

For each node n in the hierarchy, we solve eq. (4.1) to obtain the optimal weight vector denoted by $\mathbf{w}_n$. The complete set of parameters for all the nodes $[\mathbf{w}_n]_{n \in \mathcal{N}}$ constitutes the learned model for top-down classifier.

For a test example with feature vector $\mathbf{x}(i)$, the top-down classifier predicts the class label

Table 4.2: Dataset statistics.

Dataset	Total Nodes	Leaf Nodes	Levels	Train	Test	Features
CLEF	88	63	3	10000	1006	80
DIATOMS	399	311	3	1940	993	371
IPC	553	451	3	46324	28926	1123497
DMOZ-SMALL	2388	1139	5	6323	1858	51033
DMOZ-2010	17222	12294	5	128710	34880	381580
DMOZ-2012	13963	11947	5	383408	103435	348548

$\hat{y}(i) \in \mathcal{L}$ as shown in eq. (4.2). Essentially, the algorithm starts at the root and recursively selects the best child node until it reaches a terminal node which is the predicted label.

$$\hat{y}(i) = \left\{ \begin{array}{l} \textbf{initialize} \quad p := root \\ \\ \textbf{while } p \notin \mathcal{L} \\ \\ \quad p := \textbf{argmax}_{q \in \mathcal{C}(p)} \; f_q(\mathbf{x}(i)) \\ \\ \textbf{return } p \end{array} \right\} \tag{4.2}$$

4.4 Experimental Protocol

4.4.1 Datasets

We have used an extensive set of datasets for evaluating the performance of our proposed rewiring approach. Various statistics of the datasets used are listed in Table 4.2. CLEF [48] and DIATOMS [49] are image datasets and the rest are text datasets. IPC[2] is a collection of patent documents and the DMOZ datasets are an archive of web-pages available from LSHTC[3] challenge website. For evaluating the DMOZ-2010 and DMOZ-2012 datasets we use the provided test split. The results reported for these two benchmarks are blind prediction (i.e., we do not know the ground truth labels for the test set) obtained from

[2] http://www.wipo.int/classifications/ipc/en/

[3] http://lshtc.iit.demokritos.gr/

the web-portal interface[4,5]. For all text datasets we apply the tf-idf transformation with $l2$-norm normalization on the word-frequency feature vector.

4.4.2 Methods for Comparison

Hierarchical Methods

Based on the hierarchy used during the training process, we use the following methods for comparison.

Top-Down Logistic Regression (TD-LR): Expert-defined hierarchy provided by domain experts is used for training the classifiers.

Clustering Approach: Hierarchy generated using agglomerative clustering is used. For evaluation, we have restricted the height of clustered hierarchy to the original height by flattening using cluster cohesion [1].

Global Inconsistent Node Flattening (Global-INF) [2]: Hierarchy is modified by flattening (removing) the inconsistent nodes based on optimal optimization objective value obtained at each node (eq. (4.1)) and empirically defined global cut-off threshold.

Optimal Hierarchy Search [27]: Optimal hierarchy is identified in the hierarchical space by gradually modifying the expert-defined hierarchy using elementary operations – promote, demote and merge. For reducing the number of operations (and hence hierarchy evaluations), we have restricted the modification to the hierarchy branches where we encountered the maximum classification errors. This modified approach is referred as T-Easy. In the original paper [27], the largest evaluated dataset has 244 classes and 15795 instances.

Flat Method

The hierarchy is ignored and binary one-versus-rest $l2$-regularized LR classifiers are trained for each of the leaf categories. The prediction decision for unlabeled test instances is based on the maximum prediction score achieved across the several leaf categories classifiers.

[4] http://lshtc.iit.demokritos.gr/node/81

[5] http://lshtc.iit.demokritos.gr/LSHTC3_oracleUpload

State-of-the-art Cost-sensitive Learning [53]

Similar to flat method but with cost value associated with each instance in the loss function as shown in eq. (4.3). This approach is referred as HierCost and for evaluations we have used the best cost function "exponential tree distance (ExTrD)" proposed in the paper.

$$f_n = \min_{\mathbf{w}_n} \left[C \sum_{i=1}^{N} \sigma_i \log\left(1 + \exp\left(-y_n(i)\mathbf{w}_n^T \mathbf{x}(i)\right)\right) + \frac{1}{2}\left\|\mathbf{w}_n\right\|_2^2 \right] \tag{4.3}$$

where σ_i is the cost value assigned to example i.

4.4.3 Experimental Details

To make the experimental results comparable to previously published results we use the same train-test split as provided by the public benchmarks. In all the experiments we divide the training dataset into train and a small validation dataset in the ratio 90:10. The final reported testing performance is done on an independent held-out dataset as provided by these benchmarks. The model is trained by choosing the misclassification penalty parameter C in the set $[0.001, 0.01, 0.1, 1, 10, 100, 1000]$. The best parameter selected using a validation set is used to retrain the models on the entire training set. For our proposed rewiring approach, we compute the pairwise similarities between classes using the entire training dataset. Additionally, we use the liblinear solver[6] for optimization in all the experiments. The source code is made available at our website: http://cs.gmu.edu/∼mlbio/TaxMod

[6] http://www.csie.ntu.edu.tw/∼cjlin/liblinear/

Table 4.3: μF_1 and MF_1 performance comparison using different hierarchy modification approaches on newsgroup dataset. Table shows mean and (standard deviation) in bracket across five runs.

Metric	**TD-LR**	**Clustering [1]** **Agglomerative**	**Flattening [2]** **Global-INF**	**Proposed** **rewHier**
	[Figure 4.1(a)]	[Figure 4.1(b)]	[Figure 4.1(c)]	[Figure 4.1(d)]
$\mu F_1(\uparrow)$	77.04 (0.18)	78.00 (0.09)	79.42 (0.12)	**81.24 (0.08)**
$MF_1(\uparrow)$	77.94 (0.04)	78.20 (0.01)	79.82 (0.07)	**81.94 (0.04)**

4.5 Discussion and Results

4.5.1 Case Study

To understand the quality of different hierarchical structures (expert-defined, clustered, flattened and rewired) for the newsgroup dataset shown in Figure 4.1, we perform top-down HC using each of the hierarchy, separately. The dataset has 11269 training instances, 7505 test instances and 20 classes. We evaluate each of the hierarchy by randomly selecting five different sets of training and test split in the same ratio as original dataset.

The results of classification performance is shown in Table 4.3. We can see that using these modified hierarchies substantially improves the classification performance in comparison to the baseline expert-defined hierarchy. On comparing the clustered, flattened and proposed rewired hierarchies, the classification performance obtained from using the rewired hierarchy is found to be significantly better than the flattened and clustered hierarchy. This is because rewired hierarchy can resolve inconsistencies by grouping together the classes from different hierarchical branches.

4.5.2 Evaluating Rewiring Approaches

Performance based on Flat Metrics

Table 4.4 shows the μF_1 and MF_1 performance comparison of rewiring approaches against expert-defined, clustered and flattened hierarchy baselines. The rewiring approaches consistently outperform other baselines for all the datasets across all metrics. For image datasets,

Table 4.4: μF_1 and MF_1 performance comparison using different hierarchy modification approaches.

Dataset	**Evaluation Metrics**	**TD-LR**	**Agglomerative Clustering [1]**	**Flattening Global-INF [2]**	**Rewiring Methods**	
					T-Easy [27]	**rewHier**
CLEF	$\mu F_1(\uparrow)$	72.74	73.24	77.14	**78.12**	78.00
	$MF_1(\uparrow)$	35.92	38.27	46.54	**48.83▲**	47.10▲
DIATOMS	$\mu F_1(\uparrow)$	53.27	56.08	61.31	**62.34▲**	62.05▲
	$MF_1(\uparrow)$	44.46	44.78	51.85	**53.81▲**	52.14▲
IPC	$\mu F_1(\uparrow)$	49.32	49.83	52.30	53.94△	**54.28△**
	$MF_1(\uparrow)$	42.51	44.50	45.65	**46.10△**	46.04△
DMOZ-SMALL	$\mu F_1(\uparrow)$	45.10	45.94	46.61	NS	**48.25△**
	$MF_1(\uparrow)$	30.65	30.75	31.86	NS	**32.92▲**
DMOZ-2010	$\mu F_1(\uparrow)$	40.22	NS	42.37	NS	**43.10**
	$MF_1(\uparrow)$	28.37	NS	30.41	NS	**31.21**
DMOZ-2012	$\mu F_1(\uparrow)$	50.13	NS	50.64	NS	**51.82**
	$MF_1(\uparrow)$	29.89	NS	30.58	NS	**31.24**

▲ (△) indicates that improvements are statistically significant with 0.01 (0.05) significance level. We have used sign-test and non- parameteric wilcoxon rank test for statistical evaluation of μF_1 and MF_1 scores, respectively. Test are performed between rewiring methods and the best baseline, Global-INF. These statistical tests are not performed on DMOZ-2010 and DMOZ-2012 datasets because we do not have access to true labels from the online evaluation system. 'NS' denotes Not Scalable.

the relative performance improvement is larger with performance improvement up to ~11% using MF_1 scores in comparison to the baseline TD-LR method.

In Table 4.4 results with p-values < 0.01 and < 0.05 are denoted by ▲ and △, respectively. We compute the sign-test for μF_1 [52] and non-parametric wilcoxon rank test for MF_1 comparing the F_1 scores obtained per class for the rewiring methods against the best baseline *i.e.* Global-INF. Both, the rewiring approaches significantly outperform the Global-INF method across the different datasets.

The proposed *rewHier* approach shows competitive classification performance in comparison to the T-Easy approach. For smaller datasets, the T-Easy approach has better performance because it searches for the optimal hierarchy in the hierarchical space. However, the main drawback of the T-Easy approach is that it requires computationally expensive learning-based evaluations for reaching the optimal hierarchy making it intractable for large, real-world classification benchmarks such as DMOZ (See detailed discussion in Runtime Comparison).

Table 4.5: hF_1 performance comparison over expert-defined and new modified hierarchy. For DMOZ-2010 dataset hF_1 score is not available from the online evaluation system and for DMOZ-2012 dataset modified hierarchy is not supported.

Dataset	**Hierarchy used**	**Flattening**	**Rewiring Methods**	
		Global-INF	**T-Easy [27]**	**rewHier**
CLEF	Original	79.06	81.43	80.14
	Modified	80.87	81.82	81.28
DIATOMS	Original	62.80	64.28	63.24
	Modified	63.88	66.35	64.27
IPC	Original	64.73	67.23	68.34
	Modified	66.29	68.10	68.36
DMOZ-SMALL	Original	63.37	NS	66.18
	Modified	64.97	NS	66.30
DMOZ-2012	Original	73.19	NS	74.21

Performance based on Hierarchical Metrics

Hierarchical evaluation metrics such as hF_1 computes errors for misclassified examples based on the definition of a defined hierarchy. Table 4.5 shows the hF_1 score for the best baseline method, Global-INF and the rewiring methods evaluated over the original and the modified hierarchy. The rewiring methods shows the best performance for all the datasets because it is able to restructure the hierarchy based on the dataset that is better suited for classification.

Runtime Comparison

In Table 4.6 we compare the training times of the different models. For training, we learn the models in parallel for different classes using multiple compute nodes which are then combined to obtain the final runtime. For our proposed rewiring approach we also compute the similarity between different classes in parallel. We can see from Table 4.6 that TD-LR takes the the least time as there is no overhead associated with modifying the hierarchy; followed by the Global-INF model which requires retraining of models after hierarchy flattening. Rewiring approaches are most expensive because of the compute intensive task of either performing similarity computation in our proposed approach or multiple hierarchy evaluations using the T-Easy approach. The T-Easy method takes the longest time due

Table 4.6: Total training time (in mins).

Dataset	Baseline TD-LR	Flattening Global-INF	Rewiring Methods T-Easy [27]	 rewHier
CLEF	2.5	3.5	59	7.5
DIATOMS	8.5	10	268	24
IPC	607	830	26432	1284
DMOZ-SMALL	52	65	NS	168
DMOZ-2010	20190	25600	NS	42000
DMOZ-2012	50040	63000	NS	94800

Table 4.7: Number of elementary operation executed for rewiring approaches.

# Executed elementary operation for hierarchy modification	Dataset CLEF	 DIATOMS	 IPC
T-Easy [27] (promote, demote, merge)	52	156	412
proposed rewHier method (NC, PCRewire, ND)	25	34	42

to large number of expensive hierarchy evaluations after each elementary operations until the optimal hierarchy is reached. Table 4.7 shows the number of elementary operations executed using the T-Easy and the *rewHier* approach. We can see that T-Easy approach performs large number of operations even for smaller datasets (for *e.g.*, 412 operations for IPC datasets in comparison to 42 for the *rewHier*).

4.5.3 Effect of varying the Training Size

Figure 4.3 shows the MF_1 comparison of rewiring approaches with Global-INF approach on CLEF and DMOZ-SMALL datasets with varying percentage of training size. For both datasets we can see that rewiring approaches outperform the flattening approaches. For the CLEF dataset with smaller training percentage, the *rewHier* approach has better performance. The reason for this behavior might be the over-fitting of the optimal hierarchy with the training data in case of T-Easy approach, which results in poor performance on unseen examples. For training dataset with enough examples as expected, T-Easy method

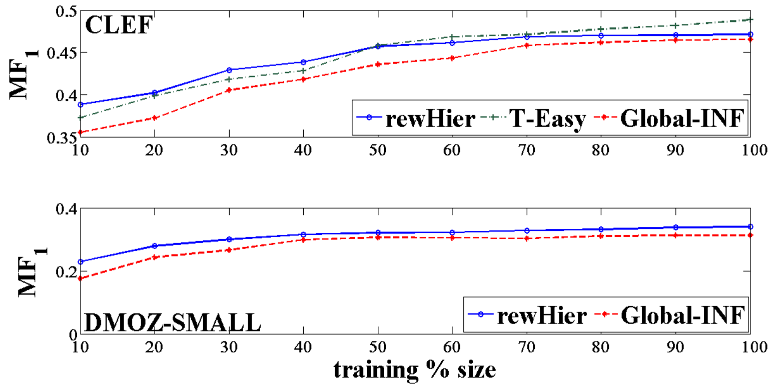


Figure 4.3: MF_1 performance comparison of rewiring approaches with best method, Global-INF, with varying % of training size. T-Easy approach is not scalable for DMOZ-SMALL dataset.

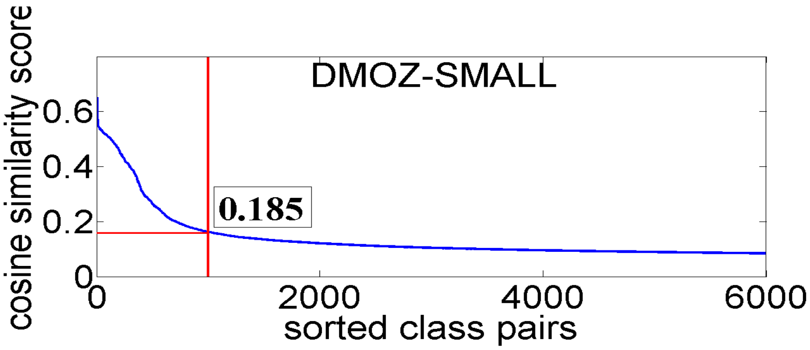


Figure 4.4: Sorted cosine similarity scores for DMOZ-SMALL dataset.

gives the best performance but at the cost of expensive runtime. We cannot run T-Easy on the larger DMOZ datasets.

4.5.4 Threshold ($\mathcal{T}$) Selection to Group Similar Classes Pairs

Figure 4.4 shows the sorted (descending order) class pairs cosine similarity scores for DMOZ-SMALL dataset. We can see that similarity scores become nearly constant after 1000 pairs (and drops further after 6000, not shown in the Figure) that does not provide any interesting similar classes grouping information for taxonomy modification. As such, for this dataset choosing threshold as the similarity score of the 1000-th class pair is a reasonable choice. A

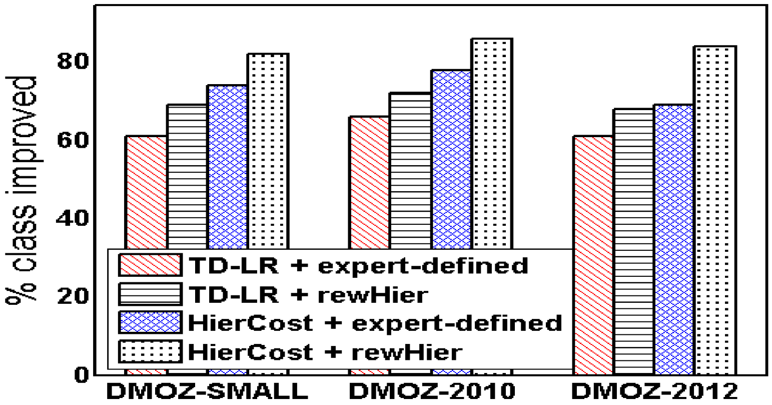


Figure 4.5: Percentage of rare categories (≤10 examples per class) classes improved over flat method.

similar approach to determine the threshold is applied for other datasets as well.

4.5.5 Improvement over Flat and State-of-the-art Approaches

Figure 4.5 presents the percentage of classes improved for TD-LR and HierCost HC approaches in comparison to the flat approach on DMOZ datasets containing rare categories *i.e.*, less than 10 training examples. For the DMOZ-2010 and DMOZ-2012 benchmarks we use a separate held out test dataset since, we do not have the true labels for the provided test set used for the online competition. From Figure 4.5 we observe that both the HC approaches outperforms the flat approach irrespective of the hierarchy being used. Rare categories benefit from the utilization of hierarchical relationships, and using the hierarchy improves the accuracy of HC. Moreover, use of *rewHier* to train the TD-LR and HierCost approaches improves the classification performance in comparison to using the expert-defined hierarchy. Further, the HierCost approach consistently outperforms the TD-LR approach because HierCost penalizes the misclassified instances based on the assignment within the hierarchy. (Interested researcher can find more comprehensive results in the supplementary material).

In terms of prediction runtime, the TD approaches outperform the flat and HierCost approaches. The flat and HierCost models invoke all the classifiers trained for the leaf nodes to make a prediction decision. For the DMOZ-2012 dataset, the flat and HierCost approaches take ~220 minutes for predicting the labels of test instances, whereas the TD-LR model is 3.5 times faster on the same hardware configuration.

4.6 Summary

We propose a data-driven filter based rewired approach for hierarchy modification that is more suited for HC. Our method is robust and can be adapted to work in conjunction with any state-of-the-art HC approaches in the literature that utilize hierarchical relationships. Irrespective of the classifiers being trained, our modified hierarchy consistently gives better performance over use of clustering or flattening to modify the original hierarchy. In comparison to previous rewiring approaches, our method gives competitive results with much better runtime performance that allow HC approaches to scale to significantly large datasets (e.g., DMOZ). Further, experiments on datasets with skewed distribution shows the effectiveness of our proposed method in comparison to flat and state-of-the-art methods, especially for classes with rare categories.

Chapter 5: Large-scale Hierarchical Classification with Feature Selection

5.1 Introduction

Many large-scale HC approaches have been developed in past to deal with the various "big data" challenges by: (i) Training faster models, (ii) Quickly predicting class-labels and (iii) Minimizing memory usage. For example, Gopal *et al.* [13] proposed the log-concavity bound that allows parallel training of model weight vectors across multiple computing units. This achieves significant speed-up along with added flexibility of storing model weight vectors at different units. However, the memory requirements is still large (∼26 GB for DMOZ-2010 dataset) which requires complex distributed hardware for storage and implementation. Alternatively, Map-Reduce based formulation of learning model is introduced [6, 16] which is scalable but have software/hardware dependencies that limits the applicability of this approach.

To minimize the memory requirements, one of the popular strategy is to incorporate the feature selection in conjunction with model training [9, 58]. The main intuition behind these approaches is to squeeze the high-dimensional features into lower dimensions. This allows the model to be trained on low-dimensional features only; significantly reducing the memory usage while retaining (or improving) the classification accuracy. This is possible because only subset of features are beneficial to discriminate between classes at each node in the hierarchy. For example, to distinguish between sub-class 'Chemistry' and 'Physics' that belongs to class 'Science' features like *chemical*, *reactions* and *acceleration* are important whereas features like *coach*, *memory* and *processor* are irrelevant. HC methods that leverage the structural relationship shows improved classification performance but are computationally expensive [6, 8, 44].

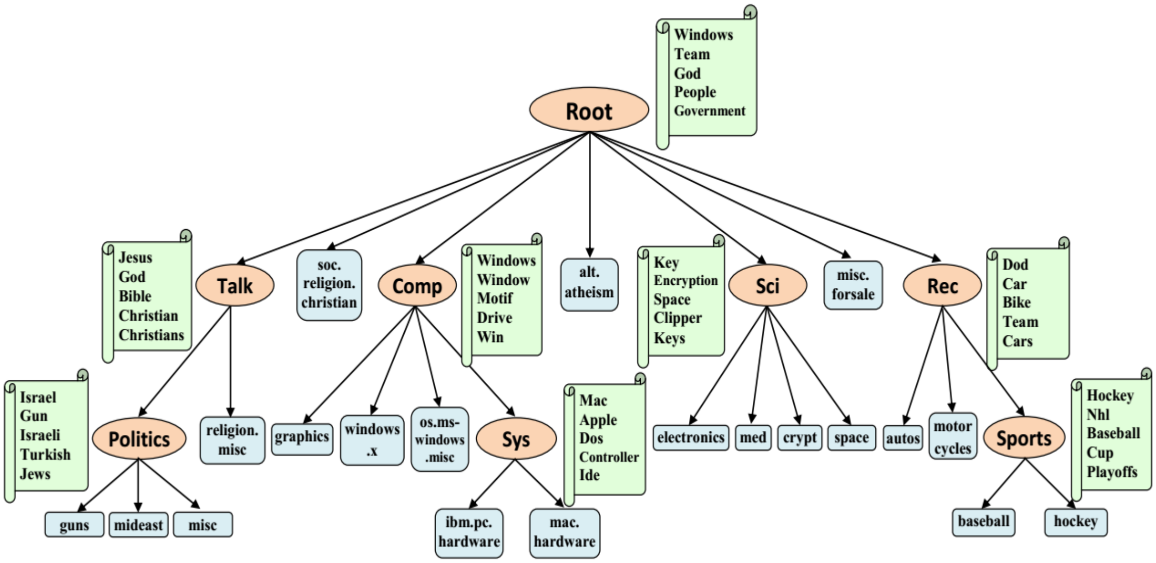


Figure 5.1: Figure demonstrating the importance of feature selection for HC. Green color (sticky note) represents the top five best features selected using gini-index feature selection method at each internal node. Internal nodes are represented by orange color (elliptical shape) and leaf nodes are represented by blue color (rectangular shape).

In this work, we study different filter-based feature selection methods for solving large-scale HC problem. Feature selection serves as the preprocessing step in our learning framework prior to training models. Any methods developed for solving HC problem can be integrated with the selected features, providing flexibility in choosing the HC algorithm of our choice along with computational efficiency and storage benefits. Based on procedure followed for selecting relevant number of features at each node, we propose two different formulations: (i) Global feature selection (Global-FS) and (ii) Adaptive feature selection (Adaptive-FS). Experimental analysis on various real world datasets across different domains demonstrates the utility of the feature selection over full set of high-dimensional features. Further, we also investigate the effect of feature selection in classification performance when the number of labeled instances per class is low.

5.2 Related Work

There have been several studies focused on feature selection methods for the flat classification problem [59–64]. However, very few work emphasize on feature selection for HC problem that are limited to small number of categories [65, 66]. Figure 5.1 demonstrates the importance of feature selection for hierarchical settings where only the relevant features are chosen at each of the decision (internal) nodes. More details about the figure will be discussed in Section 5.5 (Case Study).

Feature selection aims to find a subset of highly discriminant features that minimizes the error rate and improve the classifier performance. Based on the approach adapted for selecting features two broad categories of feature selection exist, namely, wrapper and filter-based methods. Wrapper approaches evaluate the fitness of the selected features using the intended classifier. Although many different wrapper-based approaches have been proposed, these methods are not suitable for large-scale problems due to the expensive evaluation needed to select the subset of features [59]. On the contrary, filter approaches select the subset of features based on the certain measures or statistical properties that does not require the expensive evaluations. This makes the filter-based approaches a natural choice for large-scale problem. Hence, in this work we have focused on various filter-based approaches for solving HC problem (discussed in Section 5.3.2). In literature, third category referred as embedded approaches have also been proposed which are a hybrid of the wrapper and filter methods. However, these approaches have not been shown to be efficient for large-scale classification [59] and hence, we do not focus on hybrid methods.

To the best of our knowledge this is the first work that performs a broad study of filter-based feature selection methods for HC problem.

5.3 Methods

5.3.1 Model Learning

Given a hierarchy $\mathcal{H}$, we train multi-class classifiers for each of the internal nodes $n \in \mathcal{N}$ in the hierarchy— to discriminate between its children nodes $\mathcal{C}(n)$. In this work, we have used Logistic Regression (LR) as the underlying base model for training [6]. The LR objective uses logistic loss to minimize the empirical risk and l_1-norm (denoted by $\left|\left|\cdot\right|\right|_1$) or squared l_2-norm term (denoted by $\left|\left|\cdot\right|\right|_2^2$) to control model complexity and prevent overfitting. Usually, l_1-norm encourages sparse solution by randomly choosing single parameter amongst highly correlated parameters whereas l_2-norm jointly shrinks the correlated parameters. The objective function Ψ_n^c for training a model corresponding to c-th child of node n is provided in eq. (5.1).

$$\Psi_n^c = \min_{\mathbf{w}_n^c} \left[\lambda \sum_{i=1}^{T(n)} \log\left(1 + \exp\left(-y_n^c(i)\big(\mathbf{w}_n^c\big)^T \mathbf{x}(i)\right)\right) + \mathcal{R}\big(\mathbf{w}_n^c\big) \right] \tag{5.1}$$

where $\lambda > 0$ is a mis-classification penalty parameter, $T(n)$ denotes the set of training instances considered at node n and $\mathcal{R}(.)$ denotes the regularization term given by eq. (5.2).

$$\mathcal{R}\big(\mathbf{w}_n^c\big) = \left\{ \begin{matrix} \left|\left|\mathbf{w}_n^c\right|\right|_1^1, \quad l_1 - norm \\ \\ OR \\ \\ \left|\left|\mathbf{w}_n^c\right|\right|_2^2, \quad l_2 - norm \end{matrix} \right\} \tag{5.2}$$

For each child c of node n within the hierarchy, we solve eq. (5.1) to obtain the optimal weight vector denoted by $\mathbf{w}_n^c$. The complete set of parameters for all the children nodes $\mathbf{W}_n$ = $[\mathbf{w}_n^c]_{c\in\mathcal{C}(n)}$ constitutes the learned multi-class classifiers at node n whereas total parameters for all internal nodes $\mathbf{W} = [\mathbf{W}_n]_{n\in\mathcal{N}}$ constitutes the learned model for Top-Down (TD) classifier.

For a test instance $\hat{\mathbf{x}}(i)$, the TD classifier predicts the class label $\hat{y}(i) \in \mathcal{L}$ as shown in eq. (5.3). Essentially, the algorithm starts at the root node and recursively selects the best child nodes until it reaches a terminal node belonging to the set of leaf nodes $\mathcal{L}$.

$$\hat{y}(i) = \left\{ \begin{array}{l} \textbf{initialize} \quad p := root \\ \\ \textbf{while } p \notin \mathcal{L} \\ \\ \quad p := \textbf{argmax}_{q \in \mathcal{C}(p)} \ \left(\mathbf{w}_p^q\right)^T \hat{\mathbf{x}}(i) \\ \\ \textbf{return } p \end{array} \right\} \tag{5.3}$$

5.3.2 Feature Selection

The focus of our study in this work is on filter-based feature selection methods which are scalable for large-scale datasets. In this section, we present four feature selection approaches that are used for evaluation purposes.

Gini-Index - It is one of the most widely used method to compute the best split (ordered feature) in the decision tree induction algorithm [67]. Realizing its importance, it was extended for the multi-class classification problem [68]. In our case, it measure the feature's ability to distinguish between different leaf categories (classes). Gini-Index of i^{th} feature f_i with $\mathcal{L}$ classes can be computed as shown in eq. (5.4).

$$\mathbf{Gini-Index}(f_i) = 1 - \sum_{k=1}^{\mathcal{L}} \Big(p(k|f_i)\Big)^2 \tag{5.4}$$

where $p(k|f_i)$ is the conditional probability of class k given feature f_i.

Smaller the value of Gini-Index, more relevant and useful is the feature for classification. For HC problem, we compute the Gini-Index corresponding to all feature's independently at each internal node and select the best subset of features ($S_{\mathcal{F}}$) using a held-out validation dataset.

Minimal Redundancy Maximal Relevance (MRMR) - This method incorporates the following two conditions for feature subset selection that are beneficial for classification.

(i) Identify features that are mutually maximally dissimilar to capture better representation of entire dataset and

(ii) Select features to maximize the discrimination between different classes.

The first criterion referred as "minimal redundancy" selects features that carry distinct information by eliminating the redundant features. The main intuition behind this criterion is that selecting two similar features contains no new information that can assist in better classification. Redundancy information of feature set $\mathcal{F}$ can be computed using eq. (5.5).

$$\mathfrak{R}_D = \left[\frac{1}{|S_{\mathcal{F}}|^2} \sum_{f_i, f_j \in S_{\mathcal{F}}} I(f_i, f_j) \right] \tag{5.5}$$

where $S_{\mathcal{F}} \subset \mathcal{F}$ denotes the subset of selected features and $I(f_i, f_j)$ is the mutual information that measure the level of similarity between features f_i and f_j [69].

The second criterion referred as "maximum relevance" enforces the selected features to have maximum discriminatory power for classification between different classes. Relevance of feature set $\mathcal{F}$ can be formulated using eq. (5.6).

$$\mathfrak{R}_L = \left[\frac{1}{|S_{\mathcal{F}}|} \sum_{f_i \in S_{\mathcal{F}}} I(f_i, \mathcal{L}) \right] \tag{5.6}$$

where $I(f_i, \mathcal{L})$ is the mutual information between the feature f_i and leaf categories $\mathcal{L}$ that captures how well the feature f_i can discriminate between different classes [61].

The combined optimization of eq. (5.5) and eq. (5.6) leads to a feature set with maximum discriminatory power and minimum correlations among features. Depending on strategy adapted for optimization of these two objectives different flavors exist. The first one referred as "mutual information difference (MRMR-D)" formulates the optimization problem as the difference between two objectives as shown in eq. (5.7). The second one referred as "mutual information quotient (MRMR-Q)" formulates the problem as the ratio between two objectives and can be computed using eq. (5.8).

$$MRMR\text{-}D = \max_{S_{\mathcal{F}} \subseteq \mathcal{F}} (\Re_L - \Re_D) \tag{5.7}$$

$$MRMR\text{-}Q = \max_{S_{\mathcal{F}} \subseteq \mathcal{F}} (\Re_L / \Re_D) \tag{5.8}$$

For HC problem again we select the best top $S_{\mathcal{F}}$ features (using a validation dataset) for evaluating these methods.

Kruskal-Wallis - This is a non-parametric statistical test that ranks the importance of each feature. As a first step this method ranks all instances across all leaf categories $\mathcal{L}$ and computes the feature importance metric as shown in eq. (5.9):

$$KW = (N-1) \frac{\sum_{i=1}^{\mathcal{L}} n_i (\bar{r_i} - \bar{r})^2}{\sum_{i=1}^{\mathcal{L}} \sum_{j=1}^{n_i} n_i (r_{ij} - \bar{r})^2} \tag{5.9}$$

where n_i is the number of instances in i^{th} category, r_{ij} is the ranking of j^{th} instances in the i^{th} category and $\bar{r}$ denotes the average rank across all instances.

It should be noted that using different feature results in different ranking and hence feature importance. Lower the value of computed score KW, more relevant is the feature for classification.

5.3.3 Proposed Framework

Algorithm 4 presents our proposed method for embedding feature selection into the HC framework. It consist of two independent main subroutines: (i) a feature selection algorithm

Algorithm 4: Feature Selection (FS) based Model Learning for Hierarchical Classification (HC)

```
Data: Hierarchy $\mathcal{H}$, input-output pairs $\Big(\mathbf{x}(i), y(i)\Big)$
Result: Learned model weight vectors:
        $\mathbf{W} = [\mathbf{W}_1,\, \mathbf{W}_2,\, \cdots,\, \mathbf{W}_n]$, $n \in \mathcal{N}$
$\mathbf{W} = \phi$;
/* 1st subroutine: Feature Selection */
for $f_i \in \mathcal{F}$ do
    Compute score (relevance) corresponding to feature $f_i$ using feature selection
    algorithm mentioned in Section 5.3.2;
end
Select top $k$ features based on score (and correlations) amongst features where best
value of $k$ is tuned using a validation dataset
/* 2nd subroutine: Model Learning using Reduced Feature Set */
for $n \in \mathcal{N}$ do
    /* learn models for discriminating child at node $n$ */
    Train optimal multi-class classifiers $\mathbf{W}_n$ at node $n$ using reduced feature set as
    shown in eq. (5.1);
    /* update model weight vectors */
    $\mathbf{W} = [\mathbf{W},\, \mathbf{W}_n]$;
end
return W
```

(discussed in Section 5.3.2) for deciding the appropriate set of features at each decision (internal) node and (ii) a supervised learning algorithm (discussed in Section 5.3.1) for constructing a TD hierarchical classifier using reduced feature set. Feature selection serves as the preprocessing step in our framework which provides flexibility in choosing any HC algorithm.

We propose two different approaches for choosing relevant number of features at each internal node $n \in \mathcal{N}$. The first approach which we refer as "global feature selection (Global FS)" selects the same number of features for all internal nodes in the hierarchy where the number of features are determined based on the entire validation dataset performance. The second approach, referred as "adaptive feature selection (Adaptive FS)" selects different number of features at each internal node to maximize the performance at that node. It should be noted that adaptive method only uses the validation dataset that exclusively

Table 5.1: Dataset statistics

Name	**Data Type**	**Leaf Node**	**Internal Node**	**Depth**	**Train**	**Test**	**Features**	**#children** (avgerage per internal node)
NG	Text	20	8	4	11,269	7,505	61,188	3.38
CLEF	Image	63	34	4	10,000	1,006	80	2.56
IPC	Text	451	102	4	46,324	28,926	1,123,497	5.41
DMOZ-SMALL	Text	1,139	1,249	6	6,323	1,858	51,033	1.91
DMOZ-2010	Text	12,294	4,928	6	128,710	34,880	381,580	3.49
DMOZ-2012	Text	11,947	2,016	6	383,408	103,435	348,548	6.93

belongs to the internal node n (*i.e.*, descendant categories of node n). Computationally, both approaches are almost identical because model tuning and optimization requires similar runtime which accounts for the major fraction of computation.

5.4 Experimental Protocol

5.4.1 Datasets

We have performed an extensive evaluation of various feature selection methods on a wide range of hierarchical text and image datasets. Key characteristics about the datasets that we have used in our experiments are shown in Table 5.1. All these datasets are single-labeled and the instances are assigned to the leaf nodes in the hierarchy. For text datasets, we have used the word-frequency representation and perform the tf-idf transformation with l_2-norm to the word-frequency feature vector.

Text Datasets

NEWSGROUP (NG)[1] - It is a collection of approximately 20,000 news documents partitioned (nearly) evenly across twenty different topics such as 'baseball', 'electronics' and 'graphics' (refer to Figure 5.1).

IPC[2] - Collection of patent documents organized in International Patent Classification

[1] http://qwone.com/~jason/20Newsgroups/

[2] http://www.wipo.int/classifications/ipc/en/

(IPC) hierarchy.

DMOZ-SMALL, DMOZ-2010 and DMOZ-2012[3] - Collection of multiple web documents organized in various classes using the hierarchical structure. Dataset has been released as the part of the LSHTC[4] challenge in the year 2010 and 2012. For evaluating the DMOZ-2010 and DMOZ-2012 datasets we have used the provided test split and the results reported for this benchmark is blind prediction obtained from web-portal interface[5].

Image Datasets

CLEF [48] - Dataset contains medical images annotated with Information Retrieval in Medical Applications (IRMA) codes. Each image is represented by the 80 features that are extracted using local distribution of edges method.

5.4.2 Experimental Details

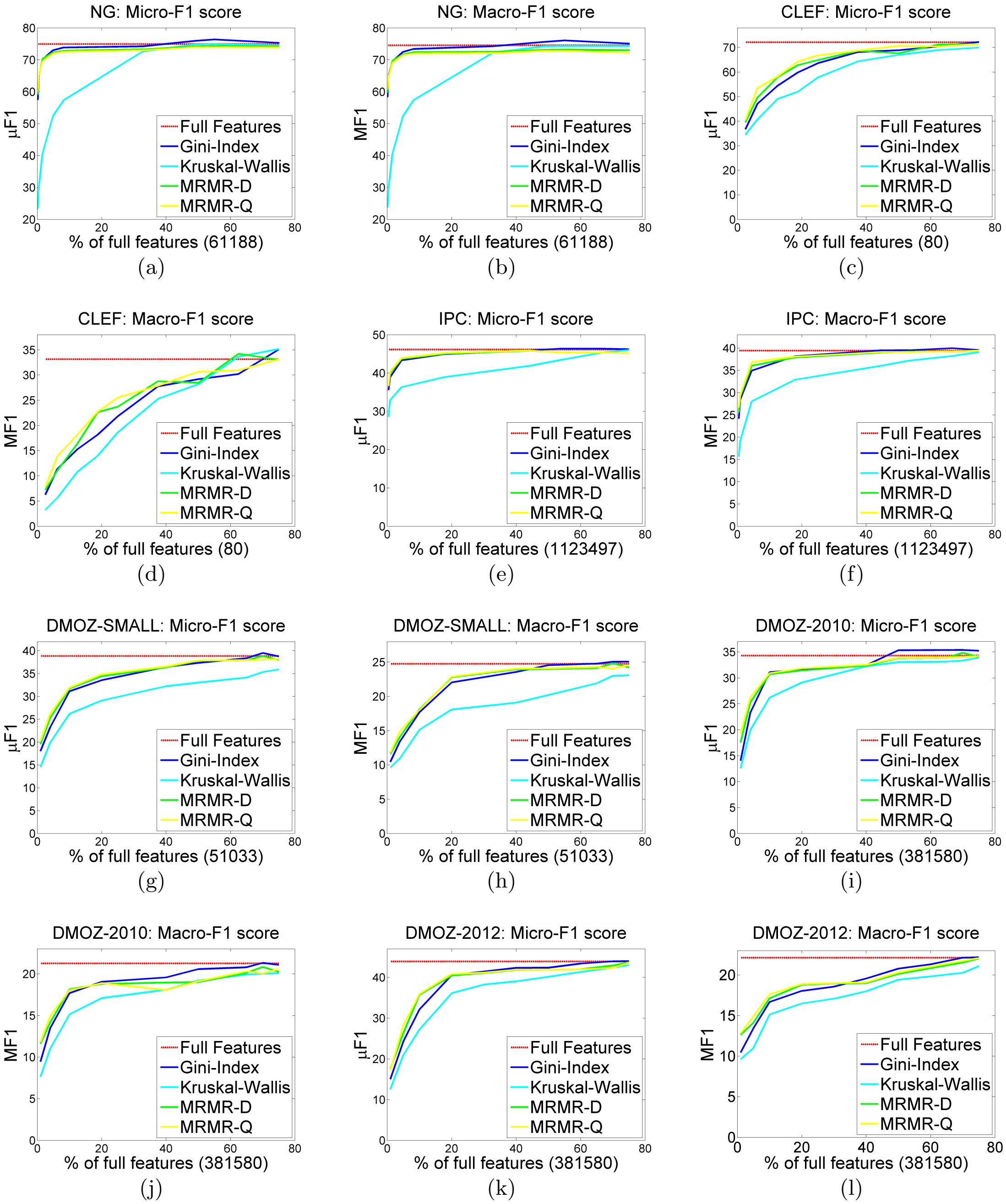
For all the experiments, we divide the training dataset into train and small validation dataset in the ratio 90:10. The train dataset is used to train TD classifiers whereas the validation dataset is used to tune the parameter. The model is trained for a range of mis-classification penalty parameter (λ) values in the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ with best value selected using a validation dataset. Adopting the best parameter, we retrain the models on the entire training dataset and measure the performance on a separate held-out test dataset. For feature selection, we choose the best set of features using the validation dataset by varying the number of features between 1% and 75% of all the features. Our preliminary experiments showed no significant improvement after 75% hence we bound the upper limit to this value. We performed all the experiments on ARGO cluster (http://orc.gmu.edu) with dual Intel Xeon E5-2670 8 core CPUs and 64 GB memory. Source code implementation of the proposed algorithm discussed in this work is made available at our website[6] for repeatability and future use.

[3] http://dmoz.org

[4] http://lshtc.iit.demokritos.gr/

[5] http://lshtc.iit.demokritos.gr/node/81

[6] https://cs.gmu.edu/∼mlbio/featureselection



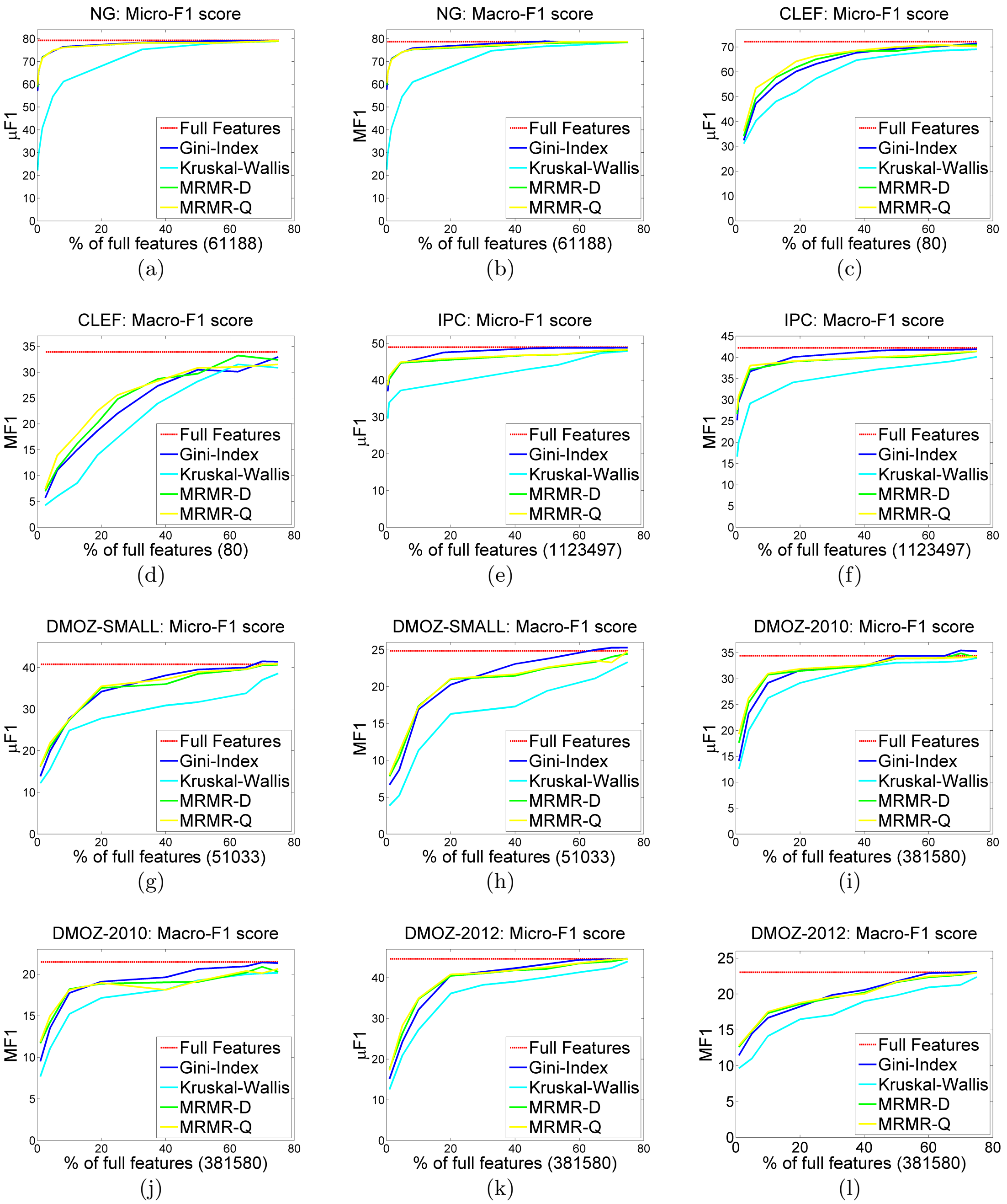
(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 5.2: Performance comparison of LR + l_1-norm models with varying percentage (%) of features selected using different feature selection (Global) methods.



(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

Figure 5.3: Performance comparison of LR + l_2-norm models with varying percentage (%) of features selected using different feature selection (Global) methods.

5.5 Results

5.5.1 Case Study

To understand the quality of features selected at different internal nodes in the hierarchy we perform case study on NG dataset. We choose this dataset because we have full access to feature information. Figure 5.1 demonstrates the results of top five features that is selected using best feature selection method *i.e.*, Gini-Index (refer to Figure 5.2 and 5.3). We can see from the figure that selected features corresponds to the distinctive attributes which helps in better discrimination at particular node. For example, the features like *Dod (Day of defeat or Department of defense)*, *Car*, *Bike* and *Team* are important at node 'Rec' to distinguish between the sub-class 'autos', 'motorcycles' and 'Sports' whereas other features like *Windows*, *God* and *Encryption* are irrelevant. This analysis illustrates the importance of feature selection for TD HC problem.

One important observation that we made in our study is that some of the features like *Windows*, *God* and *Team* are useful for discrimination at multiple nodes in the hierarchy (associated with parent-child relationships). This observation conflicts with the assumption made in the work by Xiao *et al.* [9], which attempts to optimize the objective function by necessitating the child node features to be different from the features selected at the parent node.

5.5.2 Classification Performance Comparison

Global FS - Figures 5.2 and 5.3 shows the μF_1 and $\text{M}F_1$ comparison of LR models with l_1-norm and l_2-norm regularization combined with various feature selection methods discussed in Section 5.3.2 respectively. We can see that all feature selection method (except Kruskal-Wallis) show competitive performance results in comparison to the full set of features for all the datasets. Overall, Gini-Index feature selection method has slightly better performance over other methods. MRMR methods have a tendency to remove some of the

Table 5.2: Performance comparison of adaptive and global approach for feature selection based on Gini-Index with all features. LR + l_1-norm model is used for evaluation.

Dataset	**Metric**	**Adaptive FS**	**Global FS**	**All Features**
NG	μF_1	76.16△	**76.39**△	74.94
	MF_1	**76.10**△	76.07△	74.56
CLEF	μF_1	**72.66**	72.27	72.17
	MF_1	**36.73▲**	35.07△	33.14
IPC	μF_1	**48.23▲**	46.35	46.14
	MF_1	**41.54▲**	39.52	39.43
DMOZ-SMALL	μF_1	**40.32**△	39.52	38.86
	MF_1	**26.12▲**	25.07	24.77
DMOZ-2010	μF_1	**35.94**	35.40	34.32
	MF_1	**23.01**	21.32	21.26
DMOZ-2012	μF_1	**44.12**	43.94	43.92
	MF_1	**23.65**	22.18	22.13

▲ (and △) indicates that improvements are statistically significant with 0.05 (and 0.1) significance level.

important features as redundant based on the minimization objective obtained from data-sparse leaf categories which may not be optimal and negatively influences the performance. The Kruskal-Wallis method shows poor performance because of the statistical properties that is obtained from data-sparse nodes [70].

On comparing the l_1-norm and l_2-norm regularized models of best feature selection method (Gini-Index) with all features, we can see that l_1-norm models have more performance improvement (especially for MF_1 scores) for all datasets whereas for l_2-norm models performance is almost similar without any significant loss. This is because l_1-norm assigns higher weight to the important predictor variables which results in more performance gain.

Since, feature selection based on Gini-Index gives the best performance, in the rest of the experiments we have used the Gini-Index as the baseline for comparison purpose. Also, we consider l_1-norm model only due to space constraint.

Adaptive FS - Table 5.2 shows the LR + l_1-norm models performance comparison of adaptive and global approaches for feature selection with all features. We can see from the table that adaptive approach based feature selection gives the best performance for all the

Table 5.3: Comparison of memory requirements for LR + l_1-norm model

Dataset	Adaptive FS		Global FS		All Features	
	#parameters	size	#parameters	size	#parameters	size
NG	982,805	4.97MB	**908,820**	**3.64MB**	1,652,076	6.61MB
CLEF	**4,715**	**18.86KB**	5,220	20.89KB	6,960	27.84KB
IPC	**306,628,256**	**1.23GB**	331,200,000	1.32GB	620,170,344	2.48GB
DMOZ-SMALL	**74,582,625**	**0.30GB**	85,270,801	0.34GB	121,815,771	0.49GB
DMOZ-2010	**4,035,382,592**	**16.14GB**	4,271,272,967	17.08GB	6,571,189,180	26.28GB
DMOZ-2012	**3,453,646,353**	**13.81GB**	3,649,820,382	14.60GB	4,866,427,176	19.47GB

datasets (except μF_1 score of NG dataset which has very few categories). For evaluating the performance improvement of models we perform statistical significance test. Specifically, we perform sign-test for μF_1 [52] and non-parametric wilcoxon rank test for MF_1. Results with 0.05 (0.1) significance level is denoted by ▲ (△). Tests are between models obtained using feature selection methods and all set of features. We cannot perform test on DMOZ-2010 and DMOZ-2012 datasets because true predictions and class-wise performance score are not available from online web-portal.

Statistical evaluation shows that although global approach is slightly better in comparison to full set of features they are not statistically significant. On contrary, adaptive approach is much better with an improvement of ~2% in μF_1 and MF_1 scores which are statistically significant.

5.5.3 Memory Requirements

Table 5.3 shows the information about memory requirements for various models with full set of features and best set of features that are selected using global and adaptive feature selection. Upto 45% reduction in memory size is observed for all datasets to store the learned models. This is a huge margin in terms of memory requirements considering the models for large-scale datasets (such as DMOZ-2010 and DMOZ-2012) are difficult to fit in memory.

It should be noted that optimal set of features is different for global and adaptive

Table 5.4: Feature selection preprocessing time (in mins)

Dataset	**Feature Selection Method**			
	Gini-Index	**MRMR-D**	**MRMR-Q**	**Kruskal-Wallis**
NG	**2.10**	5.33	5.35	5.42
CLEF	**0.02**	0.46	0.54	0.70
IPC	**15.24**	27.42	27.00	23.24
DMOZ-SMALL	**23.65**	45.24	45.42	34.65
DMOZ-2010	**614**	1524	1535	1314
DMOZ-2012	**818**	1824	1848	1268

methods for feature selection hence they have different memory requirements. Overall, adaptive FS is slightly better because it selects small set of features that are relevant for distinguishing data-sparse nodes present in CLEF, IPC and the DMOZ datasets. Also, we would like to point out that Table 5.3 represents the memory required to store the learned model parameters only. In practice, 2-4 times more memory is required for temporarily storing the gradient values of model paramaters that is obtained during the optimization process.

5.5.4 Runtime Comparison

Preprocessing Time - Table 5.4 shows the preprocessing time needed to compute the feature importance using the different feature selection methods. The Gini-index method takes the least amount of time since it does not require the interactions between different features to rank the features. The MRMR methods are computationally expensive due to the large number of pairwise comparisons between all the features to identify the redundancy information. On other hand, the Kruskal-Wallis method has overhead associated with determining ranking of each features with different classes.

Model Training - Table 5.5 shows the total training time needed for learning models. As expected, feature selection requires less training time due to the less number of features that needs to be considered during learning. For smaller datasets such as NG and CLEF improvement is not noticeable. However, for larger datasets with high-dimensionality such

Table 5.5: Total training time (in mins)

Dataset	Model	Feature Selection (Gini-Index)	All Features
NG	LR + l_1	0.75	0.94
	LR + l_2	0.44	0.69
CLEF	LR + l_1	0.50	0.74
	LR + l_2	0.10	0.28
IPC	LR + l_1	24.38	74.10
	LR + l_2	20.92	68.58
DMOZ-SMALL	LR + l_1	3.25	4.60
	LR + l_2	2.46	3.17
DMOZ-2010	LR + l_1	2258	6524
	LR + l_2	2132	6418
DMOZ-2012	LR + l_1	8024	19374
	LR + l_2	7908	19193

as IPC, DMOZ-2010 and DMOZ-2012 improvement is much higher (upto 3x order speed-up). For example, DMOZ-2010 dataset training time reduces from 6524 minutes to mere 2258 minutes.

Prediction Time - For the dataset with largest number of test instances, DMOZ-2012 it takes 37 minutes to make predictions with feature selection as opposed to 48.24 minutes with all features using the TD HC approach.

In Figure 5.4 we show the training and prediction time comparison of large datasets (DMOZ-2010 and DMOZ-2012) between flat LR and the TD HC approach with (and without) feature selection. The flat method is comparatively more expensive than the TD approach (∼6.5 times for training and ∼5 times for prediction).

5.5.5 Effect of Varying Training Size

Table 5.6 shows the classification performance on NG dataset with varying training dataset distribution. We have tested the models by varying the training size (instances) per class (t_c) between 5 and 250. Each experiment is repeated five times by randomly choosing t_c instances per class. Moreover, adaptive method with Gini-Index feature selection is used for

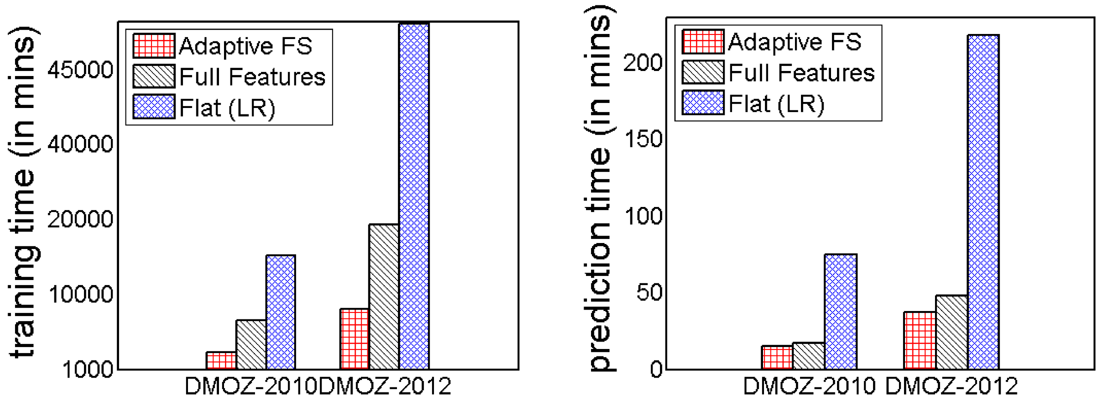


Figure 5.4: Training and prediction runtime comparison of LR + l_1-norm model (in mins)

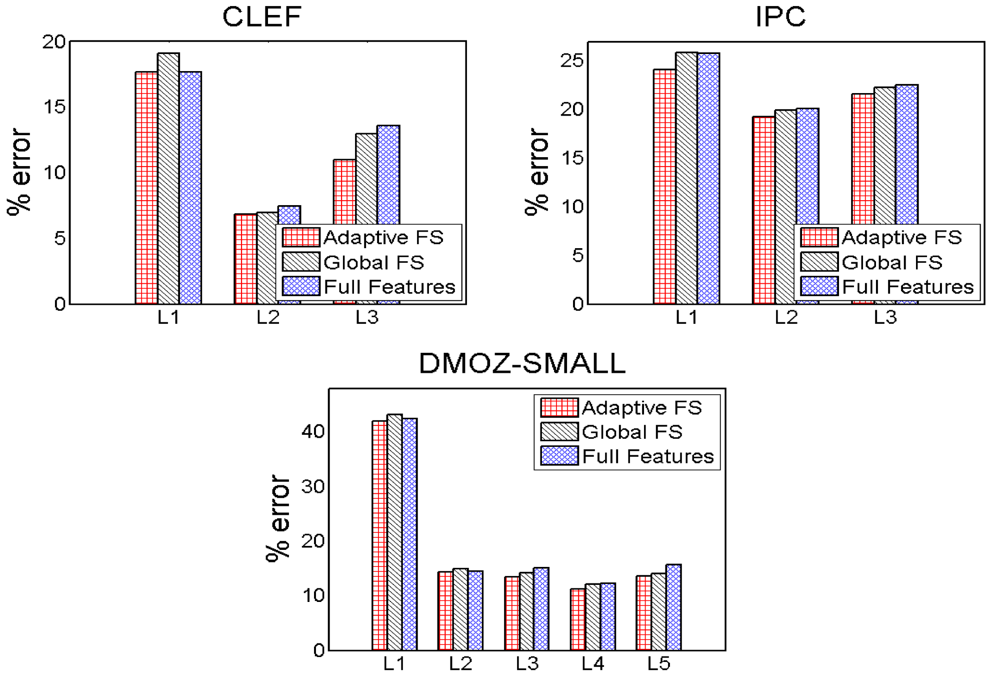


Figure 5.5: Level-wise error analysis of LR + l_1-norm model for CLEF, IPC and DMOZ-SMALL datasets.

Table 5.6: Performance comparison of LR + l_1-norm model with varying training size (number of instances) per class on NG dataset.

Dataset Distribution	**Train size (per class)**	**Feature Selection (Gini-Index)**		**All Features**	
		μF_1	$\mathbf{M}F_1$	μF_1	$\mathbf{M}F_1$
Low Distribution	5	**27.44** ▲ (0.4723)	**26.45** ▲ (0.4415)	25.74 (0.5811)	24.33 (0.6868)
	10	**37.69** △ (0.2124)	**37.51** ▲ (0.2772)	36.59 (0.5661)	35.86 (0.3471)
	15	**43.14** △ (0.3274)	**43.80** △ (0.3301)	42.49 (0.1517)	42.99 (0.7196)
	25	**52.12** ▲ (0.3962)	**52.04** ▲ (0.3011)	50.33 (0.4486)	50.56 (0.5766)
High Distribution	50	**59.55** (0.4649)	59.46 (0.1953)	59.52 (0.3391)	**59.59** (0.1641)
	100	66.53 (0.0346)	66.42 (0.0566)	**66.69** (0.7321)	**66.60** (0.8412)
	200	70.60 (0.6068)	70.53 (0.5164)	**70.83** (0.7123)	**70.70** (0.6330)
	250	72.37 (0.4285)	72.24 (0.4293)	**73.06** △ (0.4732)	**72.86** (0.4898)

Table shows mean and (standard deviation) in bracket across five runs. ▲ (and △) indicates that improvements are statistically significant with 0.05 (and 0.1) significance level.

experiments. For evaluating the performance improvement of models we perform statistical significance test (sign-test for μF_1 and wilcoxon rank test for MF_1). Results with 0.05 (0.1) significance level is denoted by ▲ (△).

We can see from Table 5.6 that for low distribution datasets, the feature selection method performs well and shows improvements of upto 2% (statistically significant) over the baseline method. The reason behind this improvement is that with low data distribution, feature selection methods prevents the models from overfitting by selectively choosing the important features that helps in discriminating between the models of various classes. For datasets with high distribution, no significant performance gain is observed due to sufficient number of available training instances for learning models which prevents overfitting when using all the features.

5.5.6 Levelwise Analysis

Figure 5.5 shows the level-wise error analysis for CLEF, IPC and DMOZ-SMALL datasets with or without feature selection. We can see that at topmost level more error is committed compared to the lower level. This is because at higher levels each of the children nodes that needs to be discriminated is the combination of multiple leaf categories which cannot be modeled accurately using the linear classifiers. Another observation is that adaptive feature selection gives best results at all levels for all datasets which demonstrates its ability to extract relevant number of features at each internal node (that belongs to different levels) in the hierarchy.

5.6 Summary

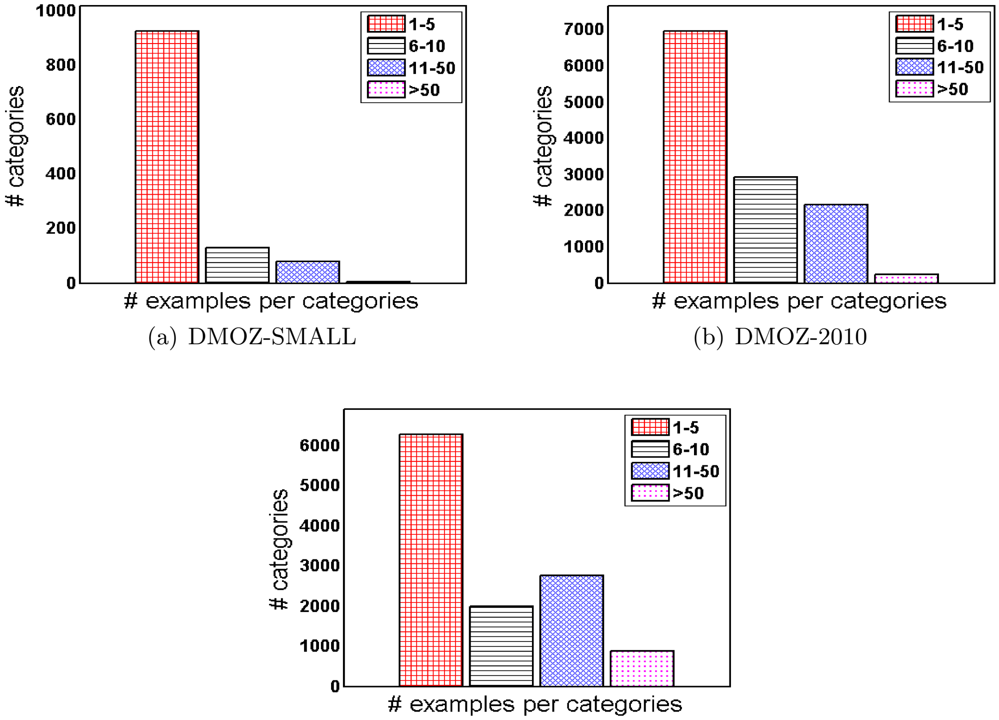
In this work we proposed two different approaches for embedding feature selection into HC framework. For evaluation, we have used four different easily parallelizable feature selection methods in the literature. Experimental evaluation shows that with feature selection we are able to achieve significant improvement in terms of runtime performance (training and prediction) without affecting the accuracy of learned classification models. We also showed that feature selection can be beneficial, especially for the larger datasets in terms of memory requirements. This is the first work presenting the systematic study of various information theoretic feature selection methods for large-scale HC.

Chapter 6: Ranking Models

6.1 Introduction

With the growth of information organized in hierarchical databases, it is essential to develop automated approaches for classifying test instances (e.g., documents, proteins and images) into hierarchies. Several classification approaches have been developed that exploit the hierarchical structure prevalent within these underlying databases. One commonly used approach is to train local one-versus-rest classifiers for each of the nodes within the hierarchy and then make a prediction using a combination of these several trained classifiers. However, the major problem with this approach is that it doesn't perform well for classes with rare categories. In this chapter, we will discuss the various methods that we have developed for addressing the HC issue related to rare categories problem.

Sparsity of instances (examples) distribution on leaf categories poses a severe challenge for generalized classifier learning on rare categories (*i.e.*, having ≤ 10 examples). The problem becomes more evident when the number of categories becomes huge due to the power law distribution followed in the evolution of large-scale taxonomies [71]. Figure 6.1 shows the category-wise data distribution for DMOZ datasets. We can see from the figure that for DMOZ datasets most of the categories (*i.e.* $\geq 75\%$) have fewer (≤ 10) training examples assigned to them and hence suffer from rare categories problem. Hierarchy provides useful information for training these rare categories. We have developed various rank-based approaches for dealing with the rare categories issue.



(a) DMOZ-SMALL

(b) DMOZ-2010

(c) DMOZ-2012

Figure 6.1: Distribution of DMOZ datasets visualizing majority of the classes with rare categories (marked in red and black).

6.2 Related Work

Ranking methods have been used extensively in the web search engines for identifying the relevant documents corresponding to the user search query [72]. It has also been studied extensively and found to be successful in variety of other application domains that includes recommender systems [73], drug discovery [74] and bioinformatics [75]. The objective of ranking methods is to learn a ranking function that given an input query, scores relevant results higher than irrelevant ones or produces an ordinal list of outputs. Based on the choice of ranking function and problem formulation, these methods can be grouped into three broad categories [76]. (i) Pointwise approaches solve a regression problem [77] to estimate the ranking score per instance, McRank [78] and Prank [79] are examples of this approach. (ii)

Pairwise approaches, like RankBoost [80] and RankSVM [81], perform pairwise comparison between training examples to assign higher ranking scores to relevant examples compared to irrelevant examples. These methods are known for their good generalization performance but involve high computational costs due to $O(N^2)$ possible comparisons between N training examples [82]. (iii) Listwise approaches like ListNet [83] and AdaRank [84], minimize a ranking loss based on the relative ordering of all examples within a given set of lists.

Ranking models have been shown that it can be used to construct highly generalizable classification models [85]. However, they have not been studied from the hierarchical classification perspective. In this work, we specifically study the use of pairwise ranking based loss functions for training leaf nodes local classifiers to leverage the hierarchical structure. We also study the effect of using cost-sensitive loss functions along with incorporation of regularization penalties that enforce parent and children within the hierarchy to have similar learned models within our framework.

Ranking methods for hierarchical classification: In order to formulate the hierarchical classification problem with a ranking approach, we provide a ranking score to each of the training instances. This score is assigned to the training examples based on their relative position in the hierarchy. Given a specific leaf-node within the hierarchy, for which we are training the ranking based classifier, examples that belong to this node (positive instance) are assigned the highest ordinal score compared to all other examples (rest/negative instance) in the hierarchy. For the rest of the training examples, the ranking score is based on the hierarchical distances from the leaf node for which we are learning the specific model. If two training instances have the same hierarchical distance, then we assign the same ranking score to each of them. The intuition behind our approach is that sorting the examples allows for information of the hierarchy to be captured better than the one-versus-rest approach of training the leaf node classifiers.

Figure 6.2 shows the ranking assignment for training examples that belong to different leaf nodes within the hierarchy, from the perspective of training the individual leaf node classifiers. In Figure 6.2, each leaf node has some set of positive instances denoted by $\mathbf{x}(i)$.

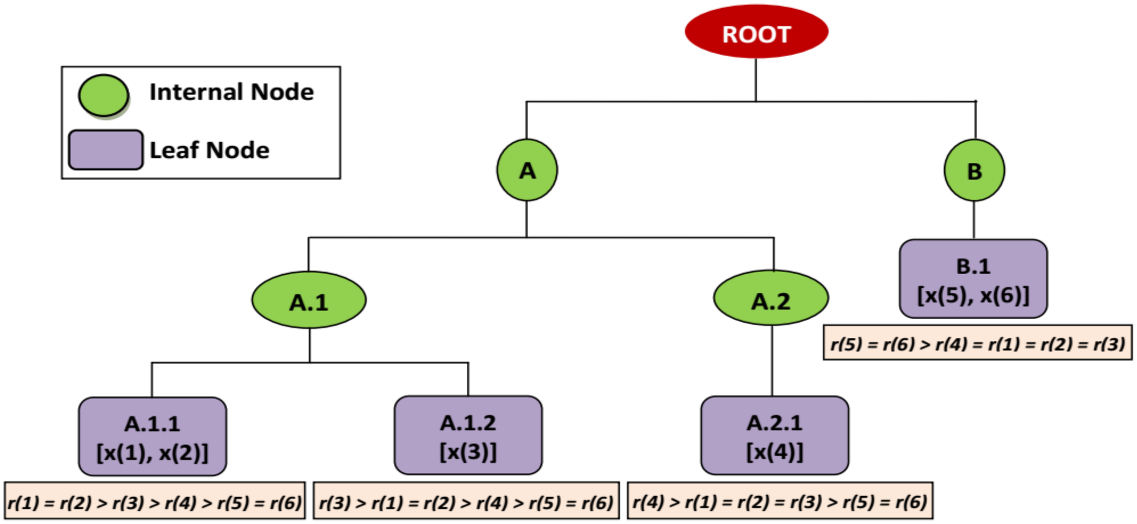

Internal Node
Leaf Node
ROOT
A
B
A.1
A.2
B.1
[x(5), x(6)]
r(5) = r(6) > r(4) = r(1) = r(2) = r(3)
A.1.1
[x(1), x(2)]
A.1.2
[x(3)]
A.2.1
[x(4)]
r(1) = r(2) > r(3) > r(4) > r(5) = r(6)
r(3) > r(1) = r(2) > r(4) > r(5) = r(6)
r(4) > r(1) = r(2) = r(3) > r(5) = r(6)

Figure 6.2: Ranking assignment $r(i)$'s for examples $\mathbf{x}(i)$'s at different leaf nodes in the hierarchy.

For ranking models, we assign rank to each of the $\mathbf{x}(i)$ denoted by $r(i)$. Ranking assignment is done based on the hierarchical distance from the leaf node for which we are learning the model. For *e.g.*, in order to learn the ranking model for leaf node A.1.1 we assign all examples belonging to this category the highest rank, followed by examples belonging to category A.1.2 which is the nearest leaf node category to A.1.1, followed by examples belonging to leaf node A.2.1 and then B.1 the furthest leaf node. In ranking models, for the leaf node all pairs of examples are considered which captures how well the example pairs are related to the leaf node. In contrast, one-vs-rest classifiers only rely on positive and negative examples which does not take into account the degree of relevance for the negative examples.

Assume N rank ordered training examples represented by pairs $(\mathbf{x}(i), r(i))$, the objective of a ranking model is to learn a linear function $f(\mathbf{x}) = \mathbf{w}^{\mathbf{T}}\mathbf{x}$, such that higher ranked pairs are assigned higher scores *i.e.*, $f\big(\mathbf{x}(i)\big) > f\big(\mathbf{x}(j)\big)$ iff $r(i) > r(j)$. Function $f(\mathbf{x})$ provides a ranking score for new and unseen examples. Pairwise ranking models learn the weight vector $\mathbf{w}_l$ corresponding to each leaf node l by minimizing a rank-based loss function across all the training example pairs, while restricting the model to have low complexity using a regularization penalty. We can express general optimization for pairwise ranking models as:

$$\min_{\mathbf{w}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \underbrace{L(\mathbf{w}^T\mathbf{x}(i), \mathbf{w}^T\mathbf{x}(j), z_{ij})}_{Ranking\ Loss} + \lambda \underbrace{\Omega(\mathbf{w})}_{Regularization} \tag{6.1}$$

where $L(\cdot)$ represents the ranking loss function, $\Omega(\cdot)$ represents the regularization (e.g., l_2-norm) and $\lambda \geq 0$ balances the trade-off between the loss and regularization terms. The regularization term safeguards against model over-fitting and allows the model to generalize to the examples not encountered in the training set. In case of the well known ranking SVM model [81], the loss function is a hinge loss and the regularization penalty is a l_2-norm. The optimization problem is expressed as:

$$\min_{\mathbf{w}} \sum_{i=1}^{N-1} \sum_{\substack{j=i+1 \\ r(i)\neq r(j)}}^{N} \Big[1 - z_{ij}\mathbf{w}^T\Big(\mathbf{x}(i) - \mathbf{x}(j)\Big)\Big]_+ + \frac{\lambda}{2}||\mathbf{w}||_2^2 \tag{6.2}$$

where $[a]_+$ denotes the hinge loss given by $\max(0, a)$, and z_{ij} denotes the ranking preference given by:

$$z_{ij} = \begin{cases} 1 & \text{if } r(i) > r(j) \\ 0 & \text{if } r(i) = r(j) \\ -1 & otherwise \end{cases} \tag{6.3}$$

The ranking preference z_{ij} will introduce a penalty for misordering any of the training example pairs no matter how far these pairs are ranked in reality (or in our hierarchy).

6.3 Methods

We modify the rank-based loss function and regularization term described in eq. (6.2) to solve the hierarchical classification problem. Various rank-based approach that we proposed are as follows:

6.3.1 Rank SVM ($rSVM$)

We modify the ranking SVM formulation (eq. (6.2)) for hierarchical classification by introducing a contribution term C_{ij} for each of the input vector pairs $\mathbf{x}(i)$ and $\mathbf{x}(j)$ to ensure that pair of examples that are close to the node for which we are learning the model have more importance than example pairs that are far away in the hierarchy. Minimization problem for this model corresponding to a leaf node l can be formulated as:

$$\min_{\mathbf{w}_l} \sum_{i=1}^{N-1} \sum_{\substack{j=i+1 \\ r(i)\neq r(j)}}^{N} C_{ij} \Big[1 - z_{ij}\mathbf{w}_l^T \Big(\mathbf{x}(i) - \mathbf{x}(j)\Big)\Big]_+ + \frac{\lambda}{2}||\mathbf{w}_l||_2^2 \tag{6.4}$$

where C_{ij} captures the contribution of training pair $\mathbf{x}(i)$ and $\mathbf{x}(j)$ in model learning and is given by,

$$C_{ij} = \begin{cases} 1 & \text{if } \mathbf{x}(i) \in l \; or \; \mathbf{x}(j) \in l \\ \frac{1}{min(h_{li}, h_{lj})} & otherwise \end{cases} \tag{6.5}$$

where h_{li} denotes the hierarchical distance (minimum number of edges) between the leaf node l for which we are learning the model and the training instance $\mathbf{x}(i)$. If one of the training instance is a member of the leaf node l, then $C_{ij} = 1$ (i.e. maximum contribution), otherwise C_{ij} is higher for training pairs that are nearby in the hierarchy for which we are learning the model (and thus making them more important for training the model, $\mathbf{w}_l$) compared to the ones far away. z_{ij} is as expressed in eq. (6.3).

6.3.2 Hierarchy Regularized Rank SVM ($rSVM^{HR}$)

We also include additional regularization term that enforce the models learned for parent and children within the hierarchy are similar to each other [6]. In order to learn the weight vectors for leaf node l we first learn the weight vectors for parent node $\pi(l)$ given by eq. (6.6) and the minimization problem for leaf node is given by eq. (6.7).

$$\min_{\mathbf{w}_{\pi(l)}} \sum_{i=1}^{N-1} \sum_{\substack{j=i+1 \\ r(i)\neq r(j)}}^{N} C_{ij}\left[1 - z_{ij}\mathbf{w}_{\pi(l)}^{T}\Big(\mathbf{x}(i) - \mathbf{x}(j)\Big)\right]_{+} + \frac{\lambda}{2}||\mathbf{w}_{\pi(l)}||_2^2 \tag{6.6}$$

where $\mathbf{w}_{\pi(l)}$ denotes the weight vector for the parent of leaf node l.

$$\min_{\mathbf{w}_l} \sum_{i=1}^{N-1} \sum_{\substack{j=i+1 \\ r(i)\neq r(j)}}^{N} C_{ij}\left[1 - z_{ij}\mathbf{w}_{l}^{T}\Big(\mathbf{x}(i) - \mathbf{x}(j)\Big)\right]_{+} + \frac{\lambda_1}{2}||\mathbf{w}_l||_2^2 + \frac{\lambda_2}{2}||\mathbf{w}_l - \mathbf{w}_{\pi(l)}||_2^2 \tag{6.7}$$

where the regularization term $||\mathbf{w}_l - \mathbf{w}_{\pi(l)}||_2^2$ force the weight vector of leaf node l to be similar to its parent weight vector $\pi(l)$, $\lambda_1 \geq 0$ and $\lambda_2 \geq 0$ denotes the regularization parameters.

6.3.3 Cost Sensitive Rank SVM ($rSVM_{CS}$)

This model takes into account the hierarchical distance between the training pairs while learning the model and assigns a different mispenalty cost to the misclassified training instances that are further away within the hierarchy compared to the ones nearby. Objective function is given by:

$$\min_{\mathbf{w}_l} \sum_{i=1}^{N-1} \sum_{\substack{j=i+1 \\ r(i)\neq r(j)}}^{N} C_{ij}\left[1 - h_{ij}z_{ij}\mathbf{w}_{l}^{T}\Big(\mathbf{x}(i) - \mathbf{x}(j)\Big)\right]_{+} + \frac{\lambda_1}{2}||\mathbf{w}_l||_2^2 \tag{6.8}$$

where h_{ij} is the hierarchical distance between training pair $\mathbf{x}(i)$ and $\mathbf{x}(j)$ (it should be noted that $C_{ij} \neq h_{ij}$). h_{ij} penalizes more the misclassified data points $\mathbf{x}(i)$ and $\mathbf{x}(j)$ that are further in the hierarchy compared to the ones nearby (whereas C_{ij} being the contribution term penalizes more the training pairs $\mathbf{x}(i)$ and $\mathbf{x}(j)$ that are further away from the node for which we are learning the model compared to the ones nearby).

Table 6.1: Dataset statistics

Name	# Leaf Node	Height	# Training	# Testing	# Features	# Training Pairs
Newsgroup	20	4	11269	7505	61188	60288759
CLEF	63	4	10000	1006	80	41960499
IPC	451	4	46324	28926	1123,497	863285980
DMOZ	1139	6	6323	1858	51033	9120885

6.3.4 Hierarchy Regularized Cost Sensitive Rank SVM ($rSVM_{CS}^{HR}$)

- This model is the combination of the cost sensitive and hierarchy regularized rank SVM models. Minimization problem for this model can be expressed as:

$$\min_{\mathbf{w}_l} \sum_{i=1}^{N-1} \sum_{\substack{j=i+1 \\ r(i)\neq r(j)}}^{N} C_{ij}\Big[1 - h_{ij} z_{ij} \mathbf{w}_l^T \Big(\mathbf{x}(i) - \mathbf{x}(j)\Big)\Big]_+ + \frac{\lambda_1}{2}||\mathbf{w}_l||_2^2 + \frac{\lambda_2}{2}||\mathbf{w}_l - \mathbf{w}_{\pi(l)}||_2^2 \tag{6.9}$$

6.4 Experimental Protocol

6.4.1 Datasets

We have performed an extensive set of experiments on a wide range of available image and text hierarchical datasets. Newsgroups[1] dataset is a collection of approximately 20,000 newsgroup documents partitioned (nearly) evenly across 20 different topics. CLEF [48] is a hierarchical dataset containing medical images annotated with Information Retrieval in Medical Applications (IRMA) codes. Each image is represented by the 80 features that are extracted using local distribution of edges method. IPC[2] is another hierarchical dataset containing patent documents classified according to the International Patent Classification (IPC). DMOZ[3] dataset contains hierarchical dataset that have been released as the part of

[1] http://qwone.com/∼jason/20Newsgroups/

[2] http://www.wipo.int/classifications/ipc/en/

[3] http://lib.iit.demokritos.g

the LSHTC challenge. Statistics about these used datasets are shown in Table 6.1. For all text datasets, we have applied the $tf - idf$ transformation with l_2-norm normalization to the word-frequency feature vector as the pre-processing step.

6.4.2 Parameter Estimation

For sub-gradient descent optimization we have set the learning rate α to 10^{-2} and the maximum number of iterations is fixed to 10^3. In all the models regularization parameters λ, λ_1 and λ_2 are tuned using a small validation set. The model is trained for a range of values for parameter λ, λ_1, λ_2 in the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ and the best model is selected using the validation set. We retrain the models using the best parameters on the entire training set and measure the performance on the held-out test dataset.

6.4.3 Hardware and Software details

All experiments were run on ARGO[4], a research computing cluster provided by the Office of Research Computing at George Mason University, VA. The pairwise method for learning ranking models is computationally expensive because of $O(N^2)$ possible pairs for a training set with N examples. To speedup the model learning we have implemented the algorithm in map-reduce. The experiments were performed on a MapReduce cluster using Hadoop version 1.2.1 on 15 Dell C8220 nodes with dual Intel Xeon E5-2670 8 core CPUs. Each node has a physical memory of 64 GB RAM. Due to resource sharing with other cluster processes only 2 map and 1 reducer slots per machine were available for Hadoop. Algorithm 5 and Algorithm 6 provides a mapper and reducer implementation of subgradient method for ranking models. It should be noted that training data that needs to be provided for ranking models is the difference of all possible input pairs $\mathbf{x}(i)$ and $\mathbf{x}(j)$.

Map-reduce implementation for the $lSVM$ models is done in a similar way.

[4] http://argo.orc.gmu.edu

Algorithm 5: Mapper Implementation

```
Data: key: $< row\ number >$, value: $< \mathbf{x}(i) - \mathbf{x}(j) >$
Result: Intermediate partial weight vector $\mathbf{w}_l^i$
for each input vector do
    /* check for support vectors (SV) */
    if $(((\mathbf{w}_l^{i-1})^T \mathbf{x}_{ij}) z_{ij} \leq 1)$ then
        for each $feature$ index $(fid)$ in $\mathbf{x}_{ij}$ do
            /* $x_{a,b}$ denotes $b$-th feature of $\mathbf{x}_a$ vector */
            $output(fid, -z_{ij} x_{ij,fid})$;
        end
    end
end
```

Algorithm 6: Reducer Implementation

```
Data: <key, value>: output from mapper
Result: Learned weight at $i$-th iteration $\mathbf{w}_l^i$
/* Aggregate support vectors (ASV) */
$ASV := 0$;
for each $value$ $of$ $fid$ do
    $ASV = ASV + < value(fid) >$;
end
/* Update weight vector for $i$-th iteration*/
$w_{l,fid}^i = w_{l,fid}^{i-1} - (\alpha)[\lambda w_{l,fid}^{i-1} + ASV]$;
$output(fID, w_{l,fid}^i)$;
```

6.4.4 Methods for Comparison

To evaluate the performance of our proposed approaches with the standard classification method, we implemented the various SVM model in a similar fashion as we developed the rank-based approaches. Equivalent models that we implemented are - Linear SVM ($lSVM$), Hierarchy Regularized lSVM ($lSVM^{HR}$), Cost Sensitive lSVM ($lSVM_{CS}$) and Hierarchy Regularized Cost Sensitive lSVM ($lSVM_{CS}^{HR}$). Minimization problem for this models can be formulated as:

$$\textbf{lSVM} \qquad \min_{(\mathbf{w}_l,b)} \sum_{i=1}^{N} \Big[1 - y(i)\Big(\mathbf{w}_l^T\mathbf{x}(i) + b\Big)\Big]_+ + \frac{\lambda}{2}||\mathbf{w}_l||_2^2 \tag{6.10}$$

$$\textbf{lSVM}^{\textbf{HR}} \qquad \min_{(\mathbf{w}_l,b)} \sum_{i=1}^{N} \Big[1 - y(i)\Big(\mathbf{w}_l^T\mathbf{x}(i) + b\Big)\Big]_+ + \frac{\lambda_1}{2}||\mathbf{w}_l||_2^2 + \frac{\lambda_2}{2}||\mathbf{w}_l - \mathbf{w}_{\pi(l)}||_2^2 \tag{6.11}$$

$$\textbf{lSVM}_{\textbf{CS}} \qquad \min_{(\mathbf{w}_l,b)} \sum_{i=1}^{N} \Big[1 - max(1, h_{li})y(i)\Big(\mathbf{w}_l^T\mathbf{x}(i) + b\Big)\Big]_+ + \frac{\lambda}{2}||\mathbf{w}_l||_2^2 \tag{6.12}$$

$$\textbf{lSVM}_{\textbf{CS}}^{\textbf{HR}} \qquad \min_{(\mathbf{w}_l,b)} \sum_{i=1}^{N} \Big[1 - max(1, h_{li})y(i)\Big(\mathbf{w}_l^T\mathbf{x}(i) + b\Big)\Big]_+ + \frac{\lambda_1}{2}||\mathbf{w}_l||_2^2 + \frac{\lambda_2}{2}||\mathbf{w}_l - \mathbf{w}_{\pi(l)}||_2^2 \tag{6.13}$$

where $\mathbf{x}(i)$ is the i-th input training instance, $y(i)$ =$\{\pm 1\}$ is the output label for i-th instance, $\mathbf{w}_l$ is the weight vector, b is the costant term and $\{\lambda, \lambda 1, \lambda 2\}$ are the regularization parameters.

6.5 Results

Table 6.2 shows the performance comparison of the different approaches across various datasets. we can see that $rSVM$ models which incorporate the rank-based loss function have better performance as compared to the $lSVM$ models because the ranking approach learns a more generalized model by performing pairwise comparison of the training examples and penalizing more the misclassified pairs which are far away in the hierarchy compared to the ones that are nearby. Among $rSVM$ models, the $rSVM_{CS}^{HR}$ model has the best performance. This model captures the parent-child relationship during learning along with the mispenalty cost based on hierarchical distance which captures how good/bad training pairs are, resulting in better generalized model.

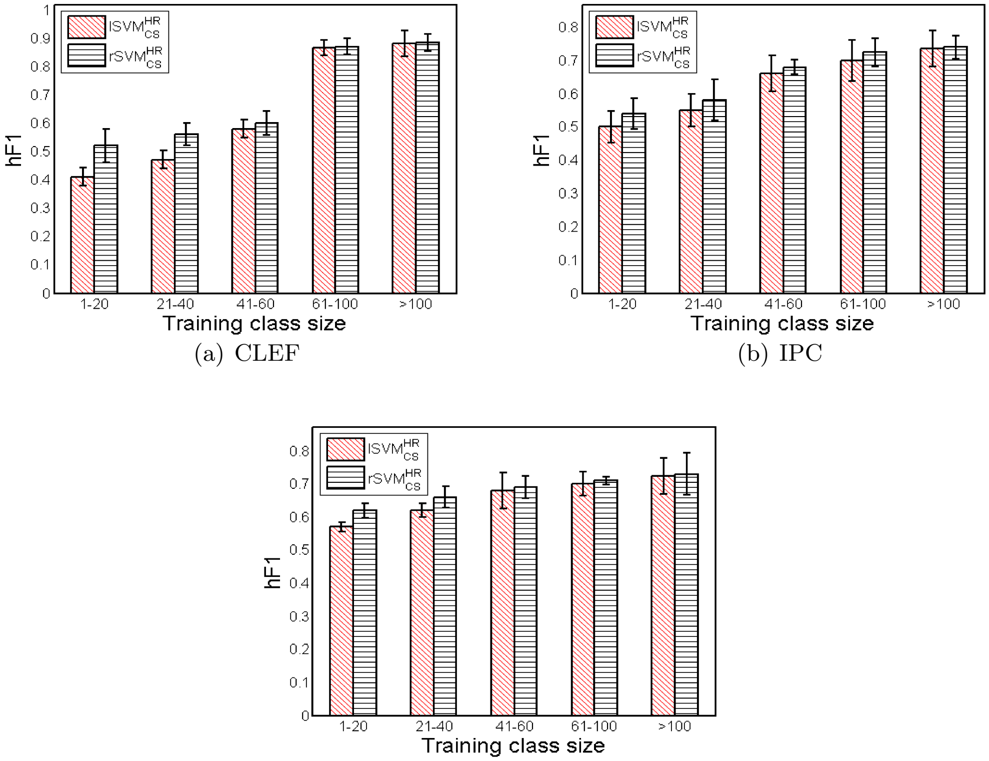
6.5.1 Observation based on number of training examples per class

Figure 6.3 shows the hF_1 comparison for best rank-based model ($rSVM_{CS}^{HR}$) and best baseline linear SVM model ($lSVM_{CS}^{HR}$) on CLEF, IPC and DMOZ-SMALL datasets with varying training sizes (NEWSGROUP dataset excluded because for all classes training size >100).

Table 6.2: Classification performance comparison of various datasets

	Linear Models				Ranking Models			
	$lSVM$	$lSVM^{HR}$	$lSVM_{CS}$	$lSVM_{CS}^{HR}$	$rSVM$	$rSVM^{HR}$	$rSVM_{CS}$	$rSVM_{CS}^{HR}$
Newsgroup								
$\mu F_1(\uparrow)$	0.72	0.76	0.74	**0.80**	0.73	0.75	0.74	0.78
	(0.0142)	(0.0324)	(0.0623)	**(0.0524)**	(0.0153)	(0.0183)	(0.0743)	(0.0888)
$MF_1(\uparrow)$	0.74	0.77	0.75	**0.80**	0.76	0.77	0.74	0.79
	(0.0426)	(0.1512)	(0.0323)	**(0.0642)**	(0.0627)	(0.0320)	(0.0727)	(0.0342)
$hF_1(\uparrow)$	0.79	0.83	0.80	0.83	0.80	0.83	0.79	**0.84**
	(0.0995)	(0.0298)	(0.0630)	(0.0348)	(0.0884)	(0.0582)	(0.0642)	**(0.0656)**
$TE(\downarrow)$	1.11	1.08	1.07	1.06	1.09	1.08	1.07	**1.05**
	(0.0894)	(0.0582)	(0.0224)	(0.0627)	(0.0720)	(0.0726)	(0.0447)	**(0.0744)**
CLEF								
$\mu F_1(\uparrow)$	0.77	0.78	0.77	0.79	0.78	0.79	0.79	**0.80**
	(0.0424)	(0.0342)	(0.0283)	(0.0723)	(0.0825)	(0.0421)	(0.0263)	**(0.0435)**
$MF_1(\uparrow)$	0.49	0.51	0.51	0.52	0.50	0.52	0.52	**0.54 ▲**
	(0.0314)	(0.0294)	(0.0632)	(0.0734)	(0.0528)	(0.0538)	(0.0780)	**(0.0647)**
$hF_1(\uparrow)$	0.79	0.80	0.80	0.81	0.81	0.82	0.82	**0.84**
	(0.0535)	(0.0435)	(0.0231)	(0.0532)	(0.0532)	(0.0264)	(0.0273)	**(0.0554)**
$TE(\downarrow)$	1.03	1.01	1.02	1.00	1.01	1.00	1.00	**0.98**
	(0.0432)	(0.0742)	(0.0532)	(0.0274)	(0.0244)	(0.0718)	(0.0123)	**(0.0326)**
IPC								
$\mu F_1(\uparrow)$	0.51	0.54	0.55	0.56	0.52	0.53	0.53	**0.57 ▲**
	(0.0605)	(0.0668)	(0.0135)	(0.0584)	(0.0521)	(0.0334)	(0.0433)	**(0.0414)**
$MF_1(\uparrow)$	0.45	0.48	0.48	0.49	0.46	0.48	0.49	**0.50 ▲**
	(0.0826)	(0.0742)	(0.0246)	(0.0466)	(0.0842)	(0.0418)	(0.0275)	**(0.0534)**
$hF_1(\uparrow)$	0.63	0.65	0.66	0.67	0.66	0.66	0.67	**0.69**
	(0.0682)	(0.0525)	(0.0624)	(0.0684)	(0.0600)	(0.0398)	(0.0852)	**(0.0508)**
$TE(\downarrow)$	2.20	2.04	2.02	1.96	2.05	2.00	1.98	**1.84**
	(0.0325)	(0.0732)	(0.0242)	(0.0683)	(0.0473)	(0.0587)	(0.0274)	**(0.0603)**
DMOZ-SMALL								
$\mu F_1(\uparrow)$	0.49	0.51	0.52	0.53	0.51	0.52	0.52	**0.54 ▲**
	(0.0632)	(0.0564)	(0.0365)	(0.0382)	(0.0832)	(0.0764)	(0.0643)	**(0.0385)**
$MF_1(\uparrow)$	0.32	0.33	0.35	0.36	0.34	0.37	0.37	**0.38 ▲**
	(0.0611)	(0.0523)	(0.0841)	(0.0742)	(0.0462)	(0.0648)	(0.0284)	**(0.0524)**
$hF_1(\uparrow)$	0.62	0.68	0.68	0.69	0.68	0.69	0.70	**0.71**
	(0.0624)	(0.0372)	(0.0423)	(0.0503)	(0.0614)	(0.0302)	(0.0274)	**(0.0688)**
$TE(\downarrow)$	2.20	2.11	2.07	2.08	2.12	2.08	2.06	**2.01**
	(0.0821)	(0.0322)	(0.0321)	(0.0652)	(0.0638)	(0.0293)	(0.0148)	**(0.0674)**

Table shows mean across five runs and (standard deviation) in bracket. Statistical significance test, sign test for μF_1 and non-parameteric wilcoxon rank test for MF_1, were performed on the results of best proposed ranking model with the best linear model. All statistically significant values at p-value of 5% is marked with ▲.



(a) CLEF

(b) IPC

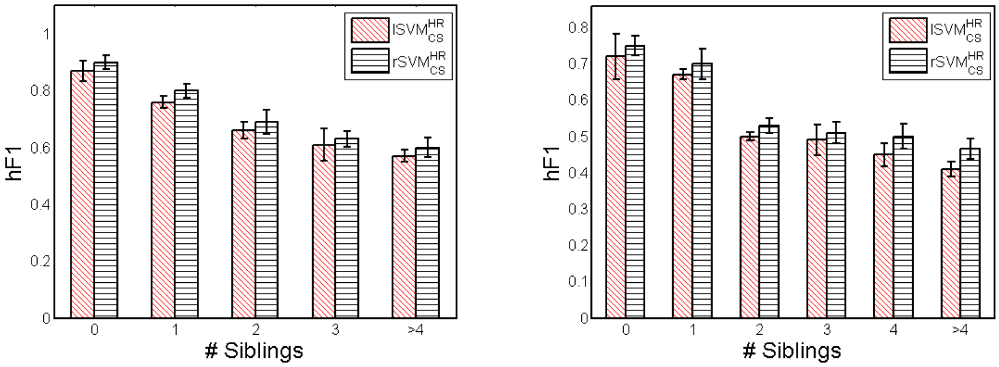
(c) DMOZ-SMALL

Figure 6.3: Performance comparison of models based on number of training examples per class for different datasets.

$rSVM^{HR}_{CS}$ model works very well compared to $lSVM^{HR}_{CS}$, especially when the number of positive training examples are small *i.e.* rare categories. The ranking models are able to use the information about the relative ordering of examples from different leaf nodes within the hierarchy far better than the one-versus-rest setup of non-ranking ($lSVM^{HR}_{CS}$) models.

6.5.2 Analyzing performance based on number of siblings per class

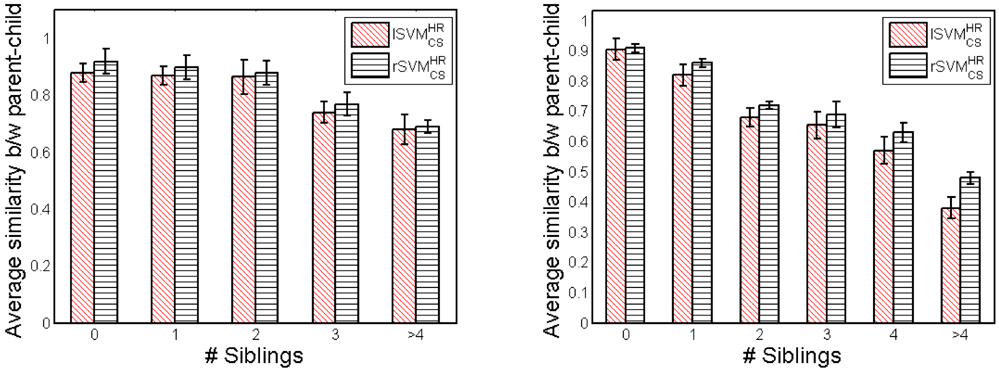
Figure 6.4 shows the hF_1 comparison for the $lSVM^{HR}_{CS}$ and $rSVM^{HR}_{CS}$ models based on number of siblings. Both models show a decrease in their hF_1 scores as the number of siblings increases because it becomes difficult to discriminate. In comparison, rank-based model are able to better discriminate between siblings compared to linear model.



(a) CLEF

(b) IPC

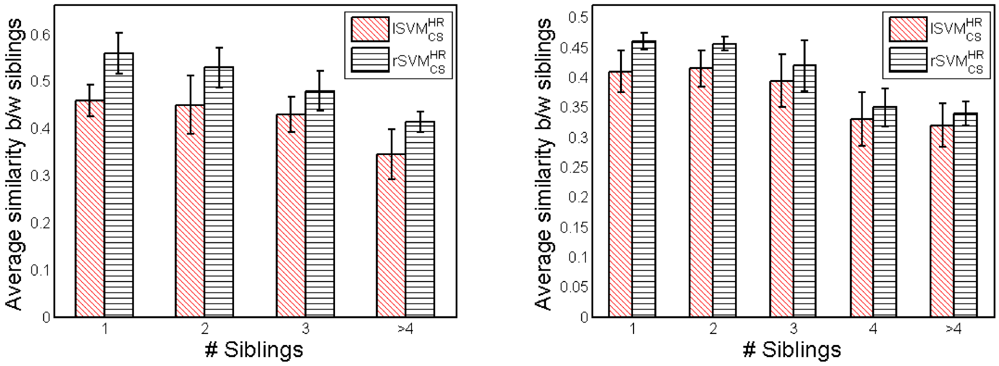
Figure 6.4: Performance comparison of models based on number of siblings per class.



(a) CLEF

(b) IPC

Figure 6.5: Performance comparison based on average similarity between parent-child.



(a) CLEF (b) IPC

Figure 6.6: Performance comparison based on average similarity between siblings.

In Figure 6.5 and Figure 6.6, we show the average pairwise similarity (dot-product) between the parent-children model vectors and the average similarity between th sibling model vectors for the CLEF and IPC dataset in relation to the number of siblings, respectively. $rSVM_{CS}^{HR}$ models have high degree of parent-child similarity compared to the $lSVM_{CS}^{HR}$ models. On comparing the average similarities between siblings, it can be seen that as the number of siblings increases the pairwise sibling similarity decreases to discriminate between the more number of siblings. Specifically, the $rSVM_{CS}^{HR}$ model have high pairwise sibling similarity ~5-10% compared to the $lSVM_{CS}^{HR}$ models, because the parent-child regularization term forces the siblings to have models similar to their parents.

6.5.3 Comparison with other Hierarchical Baseline Models

Table 6.3 shows the performance comparison of best rank-based model with the flat $lSVM$ model and hierarchical models. Hierarchical models include: (i) $lSVM^{HR}$ model that incorporates parent-child relationship [6] during model learning. (ii) Simple top-down hierarchical model using support vector classifier at each node [44]. We refer to this model as TD-SVM, and (ii) Top-down hierarchical model using orthogonal transfer method as proposed in Zhou *et al.* [9]. We refer to this model as TD-OT. Prediction for unseen test

Table 6.3: Performance comparison of best ranking model with flat and hierarchical models.

	Best Model (ranking)	Flat Model	Hierarchical Models		
	$rSVM_{CS}^{HR}$	$lSVM$	$lSVM^{HR}$	TD-SVM	TD-OT
Newsgroup					
$\mu F_1(\uparrow)$	**0.78**	0.72	0.76	0.70	0.74
$MF_1(\uparrow)$	**0.79▲**	0.74	0.77	0.74	0.76
$hF_1(\uparrow)$	**0.84**	0.83	0.83	0.78	0.79
$TE(\downarrow)$	**1.05**	1.11	1.08	1.17	1.11
CLEF					
$\mu F_1(\uparrow)$	**0.80**	0.77	0.78	0.72	0.74
$MF_1(\uparrow)$	**0.54▲**	0.49	0.52	0.35	0.38
$hF_1(\uparrow)$	**0.84**	0.79	0.80	0.74	0.76
$TE(\downarrow)$	**0.98**	1.03	1.01	1.24	1.18
IPC					
$\mu F_1(\uparrow)$	**0.57▲**	0.51	0.54	0.51	-
$MF_1(\uparrow)$	**0.50▲**	0.45	0.48	0.40	-
$hF_1(\uparrow)$	**0.69**	0.63	0.65	0.54	-
$TE(\downarrow)$	**1.84**	2.20	2.04	2.15	-
DMOZ-SMALL					
$\mu F_1(\uparrow)$	**0.54▲**	0.49	0.51	0.35	0.36
$MF_1(\uparrow)$	**0.38▲**	0.32	0.33	0.20	0.18
$hF_1(\uparrow)$	**0.71**	0.62	0.68	0.38	0.40
$TE(\downarrow)$	**2.01**	2.20	2.11	2.54	2.50

'-' denotes not scalable. ▲ denotes statistically significant results at p-value of 5% on comparing best proposed (ranking) model with the best baseline (flat, hierarchical) model.

Table 6.4: Training time (in mins) comparison in map-reduce (reported per class)

Name	$lSVM$	$lSVM^{HR}$	$lSVM_{CS}$	$lSVM_{CS}^{HR}$	$rSVM$	$rSVM^{HR}$	$rSVM_{CS}$	$rSVM_{CS}^{HR}$
Newsgroup	12.4	13.6	13.8	14.2	82.4	84.8	85.5	86.1
CLEF	11.2	13.0	13.5	14.4	68.7	70.3	70.8	72.0
IPC	36.4	37.8	38.0	38.2	614.8	626.8	632.2	634.6
DMOZ	8.1	8.8	8.9	9.1	35.2	36.1	37.1	38.5

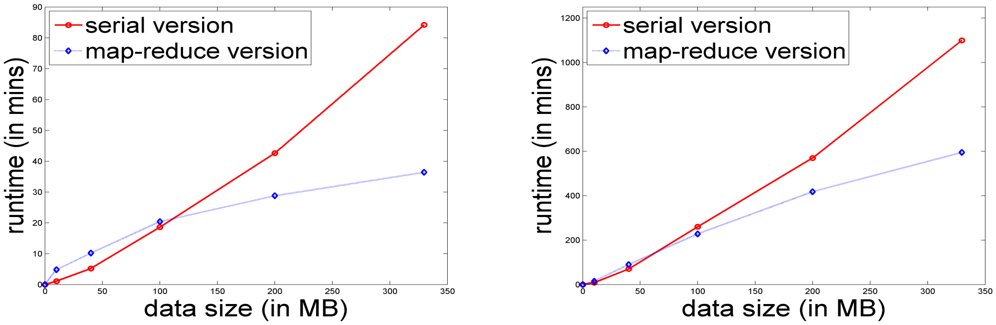
instance in hierarchical top-down model is done by starting at root node and then recursively selecting the best child node till it reaches a terminal node belonging to the set of leaf nodes $\mathcal{L}$.

Table shows that the proposed rank-based model has superior performance compared to the flat and hierarchical models. Top-down models have poor performance because of the error propagation *i.e*, misclassified examples at the top level cannot be corrected at the lower level.

6.5.4 Runtime Comparison

Table 6.4 shows the average training time (in mins) per class required to learn the models in map-reduce framework for the different datasets. As expected, $lSVM$ model takes the least time to learn the model because no overhead is involved in incorporating additional constraints. Ranking models take more time compared to there linear counterparts because ranking models have $O(N^2)$ possible input training pairs for datasets with N examples and hence is more computationally expensive. Other $lSVM$ models, $lSVM^{HR}$, $lSVM_{CS}$ and $lSVM_{CS}^{HR}$ have the overhead of either incorporating the parent-child relationship in the regularization constraint or adding hierarchical distance for cost sensitive learning or both, and hence, takes more time. Similar type of overahead also occurs in case of $rSVM^{HR}$, $rSVM_{CS}$ and $rSVM_{CS}^{HR}$ models when compared to the $rSVM$ models.

In Figure 6.7 we show the training runtime performance comparison for $lSVM$ and $rSVM$ models for serial and map-reduce implementation on the IPC dataset. It can be seen that for datasets of small size, the serial version has better performance because the



(a) $lSVM$ Model

(b) $rSVM$ Model

Figure 6.7: Serial vs. map-reduce runtime comparison (in mins) with varying size on IPC dataset.

architectural overhead in map-reduce framework exceeds the amount of parallelism that can be obtained, whereas the map-reduce version outperforms the serial version when the datasets size is large.

6.6 Summary

In this work we developed a hierarchical classification approach that trains local leaf nodes using a ranking model. We improve this approach by introducing regularization constraints that force the parent and children nodes within the hierarchy to have similar learned models. We also incorporate a hierarchy influenced cost-sensitive loss function within the optimization formulation. The performance of ranking models was compared with the flat classification approaches ($lSVM$) along with state-of-the-art HC baselines. Our empirical results on a wide variety of datasets demonstrated the strength of the proposed HC method in terms of reducing the prediction errors. We are also able to show improved performance using rank-based model with training datasets having few instances per class. Further analysis on parent-child and siblings similarity shows more degree of relatedness for rank-based models in comparison to its linear counterpart. Finally, we developed the map-reduce based algorithm for distributed training of individual models.

Chapter 7: Integrated Framework for Large-scale Hierarchical Classification

7.1 Introduction

State-of-the-art HC approaches embed the parent-child relationships from the hierarchy either, within the regularization term [6] or the loss term [53]. The intuition behind Hierarchy Regularized Logistic Regression (HR-LR) or Hierarchy Regularized Support Vector Machine (HR-SVM) approach is that data-sparse child nodes benefit during training from data-rich parent nodes and has shown to achieve the best performance on standard HC benchmarks (LSHTC datasets). In case of HierCost, a cost-sensitive learning approach was adapted. This method intuitively captures the hierarchical information by treating misclassifications differently based on the commonalities (common ancestors) between the true and the predicted labels. Another approach involves using feature selection in conjunction with HC [15, 39]. It helps to improve the accuracy and efficiency of model training along with reduce memory requirement which is crucial for large-scale problem. Other developed approaches for solving HC can be found in [2, 3, 44].

The hierarchical structure has a significant impact on the classification performance of model being trained [2]. If the hierarchy used has too many inconsistent relationships such as parent-child or siblings, less cohesive or overlapping categories, than the performance can be poor. In fact for some datasets HC methods are outperformed by flat methods that ignore the hierarchy [4, 6]. Oftentimes, the hierarchy is manually designed by domain experts based on semantic relationships. This type of curated hierarchy is prone to inconsistencies and is not optimal for achieving good classification performance. Moreover, partial or incomplete domain knowledge may result in less cohesive categories that are not easily separable and vice-versa. Furthermore, constant change in data distribution over time requires new categories (orphan nodes) to be identified to improve the taxonomic representation and hence classification performance [86]. Other problem associated with large-scale classification is the highly skewed distribution between classes, where majority of the classes have few instances (examples) for training (rare categories problem). For example, more than 75% of the categories in the ODP[1] and Yahoo![2] directory have five or fewer positive instances [6, 44]. Learning a classification model for these classes suffer from statistical challenges where mis-predictions tend to favor the larger classes. As a result, overall performance on the rare categories classes are unsatisfactory.

In this work, we propose a solution to handle multiple issues that affect the HC performance. To summarize, our main contributions are as follows:

- We develop an *integrated framework* which consists of a multi-stage embarrassingly parallel pipeline to improve the HC performance.
- We propose the exploratory learning approach for orphan node identification that can be easily incorporated in our framework.
- An extensive case study was performed to analyze the strength and effectiveness of each step in the integrated framework.

7.2 Methods

In this section we discuss the modification strategies for handling multiple issues. We are interested in following issues that are faced by HC.

1. Hierarchical structure inconsistency.
2. Rare categories.
3. Less cohesive or overlapping categories and
4. Orphan node identification.

[1] http://www.dmoz.org

[2] http://dir.yahoo.com

7.2.1 Modification Strategies

Firstly, we provide solution to handle each of these issues independently and then we discuss an integrated framework to combine these solutions. It must be noted that we are also interested in solving the large-scale problem and therefore desire solutions that are easily parallelizable.

1. **Hierarchical structure inconsistency**: Cross-linkage of inconsistent relationships is an effective method to solve hierarchical structural inconsistency. The decision to perform cross-linkage is based on the average pairwise cosine similarity score between classes and their relative positions in the hierarchy. Intuitively, the idea is to group classes with high similarities together to a common parent. To this end, first we identify a set of class pairs with high similarity scores and than we iteratively perform check on these class pairs if they are grouped together. If not than we apply elementary operations – node creation, parent-child rewiring and node deletion to correct inconsistencies as done in Naik *et al.* [10]. Node creation helps to group similar classes from different hierarchical branches by creating new node with only children as the similar classes. This operation is used only when a proper subset of the classes from different branches are similar, otherwise parent-child rewiring operation is performed which simply reassigns the parent of one class to other. Finally, node deletion operation helps to remove irrelevant nodes with 0 or 1 children.

2. **Rare categories**: Feature Selection (FS) is an effective method for dealing with rare categories [15]. It helps to prevent the model over-fitting by considering only the relevant features; thereby improving the classification performance on rare categories classes. We propose to use the Gini-index measure to determine the relevance of each feature in classification as it gives comparatively better results over other measures and can be easily parallelized. Gini-index corresponding to feature f_i can be computed as shown in eq. (7.1). Smaller the value of Gini-index better the feature is for

classification.

$$\mathbf{Gini-index}(f_i) = 1 - \sum_{k=1}^{\mathcal{L}} \Big(p(k|f_i)\Big)^2 \tag{7.1}$$

where $p(k|f_i)$ is the conditional probability of class k given feature f_i. Appropriate set of features $S_{\mathcal{F}}$ at each internal node is determined using small validation dataset.

3. **Less cohesive or overlapping categories**: Overlapping categories have a high degree of similarities between them. In order to identify overlapping categories (possible only between siblings), we first compute the pairwise cosine feature similarity and consider two or more siblings as overlapping if the similarity score exceeds a certain threshold. For experimental evaluations, we set the threshold value as 0.5. To determine the less cohesive categories, we use the feedback from classification results on a validation dataset [11]. Categories are identified to be less cohesive if $P_c << \overline{P}$ where P_c is the precision of a category and $\overline{P}$ is the average precision of siblings categories. Moreover, we use k-means clustering for splitting less cohesive categories.

4. **Orphan node identification**: We followed an exploratory learning approach to identify orphan nodes [12]. Algorithm 7 shows how the orphan nodes are determined during the prediction phase. Essentially, we start predicting the label of instances in the top-down order by computing the probability score given in eq. (7.2) and selecting the best child with highest probability score. At each level, we check if new node (*i.e.* orphan) needs to be created for better classification. The decision to create a new node is based on the probability score of children nodes; nearly uniform score (computed using MinMax partition, shown in Algorithm 8 [12]) indicates that the test instance cannot be confidently assigned to any of the children nodes and hence we create a new node. Otherwise we select the best child (with highest probability score) and proceed down the level. This process is repeated until the leaf node is reached or new orphan

```
Algorithm 7: Orphan Node Identification
Data: Test instance $\hat{\mathbf{x}}(i)$, Hierarchy $\mathcal{H}$
Result: Optimal label assignment $\hat{y}(i)$, Modified Hierarchy $\mathcal{H}_M$
$n = root$;
DetectOrphanNode = 0;
while (($n \notin \mathcal{L}$) and (DetectOrphanNode == 0)) do
    Compute probability score for children of n-th node using eq. (7.2)
    /* orphan node check */
    if $NearlyUniform\Big(\forall_{c \in \mathcal{C}(n)}\, p(\mathcal{C}_n^c|\hat{\mathbf{x}}(i))\Big)$ then
        Create new class $c'$ at level $l$
        Initialize parameters for new class $\mathbf{w}_n^{c'}$ with training instance as $\hat{\mathbf{x}}(i)$
        Parent($c'$) = $n$;
        $\hat{y}(i) = c'$;
        $\mathcal{L} = \mathcal{L} \cup \{c'\}$;
        $H_M = OrphanNode(\mathcal{H})$; /* modified hierarchy with new orphan node */
        DetectOrphanNode = 1;
    else
        $n = \underset{c \in \mathcal{C}(n)}{\mathbf{argmax}} \Big(p(\mathcal{C}_n^c|\hat{\mathbf{x}}(i))\Big)$;
        $\hat{y}(i) = n$;
    end
end
return $\mathcal{H}_M, \hat{y}(i)$
```

node is created.

$$p(\mathcal{C}_n^c|\hat{\mathbf{x}}(i)) = \frac{exp\Big((\mathbf{w}_n^c)^T\hat{\mathbf{x}}(i)\Big)}{\sum_{c \in \mathcal{C}(n)} exp\Big((\mathbf{w}_n^c)^T\hat{\mathbf{x}}(i)\Big)} \tag{7.2}$$

7.2.2 An Integrated Framework for HC

Figure 7.1 presents our proposed integrated framework. The order in which the above steps are executed is important to achieve the maximum HC performance improvement. Global restructuring must be performed first because full set of features are important to identify inconsistent parent-child, followed by the feature selection which selects relevant features amongst siblings categories for effective classification. Less-cohesive or overlapping

Algorithm 8: Criterion to determine nearly uniform probability score [12]

```
Data: Probability score of n-th children $p(\mathcal{C}_n^c|\hat{\mathbf{x}}(i))$
Result: $True$ or $False$
$maxProbability = \underset{c \,\in\, \mathcal{C}(n)}{\mathbf{argmax}} \left(p(\mathcal{C}_n^c|\hat{\mathbf{x}}(i))\right)$
$minProbability = \underset{c \,\in\, \mathcal{C}(n)}{\mathbf{argmin}} \left(p(\mathcal{C}_n^c|\hat{\mathbf{x}}(i))\right)$
if $\left(\frac{maxProbability}{minProbability} < 2\right)$ $and$ $\left(maxProbability < \frac{2}{|\mathcal{C}(n)|}\right)$ then
    $NearlyUniform = True$
else
    $NearlyUniform = False$
end
return $NearlyUniform$
```

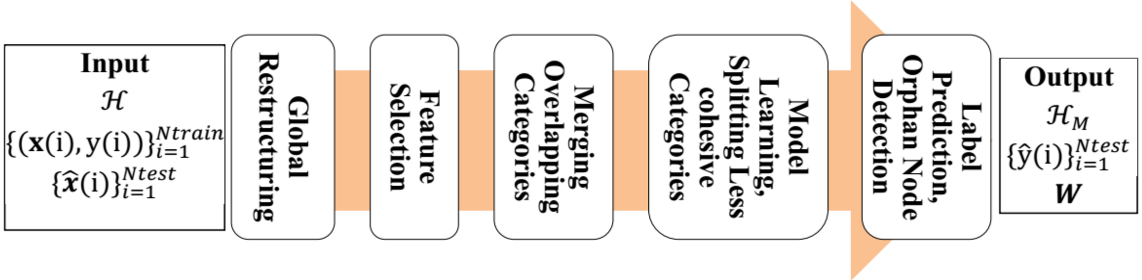

Input
$\mathcal{H}$
$\{(\mathbf{x}(i), y(i))\}_{i=1}^{Ntrain}$
$\{\hat{\mathbf{x}}(i)\}_{i=1}^{Ntest}$
Global Restructuring
Feature Selection
Merging Overlapping Categories
Model Learning, Splitting Less cohesive Categories
Label Prediction, Orphan Node Detection
Output
$\mathcal{H}_M$
$\{\hat{y}(i)\}_{i=1}^{Ntest}$
$\boldsymbol{W}$

Figure 7.1: Integrated Framework pipeline for Large-scale Hierarchical Classification.

categories are determined next which is based on feature similarities computed using features selected in the previous step. Finally, orphan node identification step is performed that is invoked during the prediction phase.

Scalability - As noted earlier, we are interested in solution which can be easily parallelized at different steps. Our proposed integrated framework is embarrassingly parallel which makes it favorable for large-scale problems. To summarize parallelization can be exploited at following steps: 1) Computing similarities between different classes. 2) Selecting features using Gini-index at each internal node and 3) Learning optimal model weight vectors.

Table 7.1: Dataset statistics

Name	Domain	Total Node	Categories	Levels	Training	Testing	Features
CLEF	Image	88	63	3	10000	1006	80
IPC	Text	553	451	3	46324	28926	1123497
DMOZ-SMALL	Text	2388	1139	5	6323	1858	51033
DMOZ-2010	Text	17222	12294	5	128710	34880	381580
DMOZ-2012	Text	13963	11947	5	383408	103435	348548

7.2.3 Model Learning

Given a hierarchy $\mathcal{H}$, we train multi-class classifiers for each of the internal nodes $n \in \mathcal{N}$ in the hierarchy— to discriminate between its children nodes $\mathcal{C}(n)$. In this work, we have used Logistic Regression (LR) as the underlying base model for training. The LR objective uses logistic loss to minimize the empirical risk and squared l_2-norm term (denoted by $||\cdot||_2^2$) to control model complexity and prevent overfitting. The objective function f_n^c for training a model corresponding to *c-th* child of node n is provided in eq. (7.3).

$$f_n^c = \min_{\mathbf{w}_c^n} \left[\lambda \sum_{i=1}^{T(n)} \log \left(1 + \exp\left(-y_n^c(i)\big(\mathbf{w}_n^c\big)^T \mathbf{x}(i)\right)\right) + ||\mathbf{w}_n^c||_2^2 \right] \tag{7.3}$$

where $\lambda > 0$ is a mis-classification penalty parameter.

For each child c of node n within the hierarchy, we solve eq. (7.3) to obtain the optimal weight vector denoted by $\mathbf{w}_n^c$. The complete set of parameters for all the children nodes $\mathbf{W}_n = [\mathbf{w}_n^c]_{c \in \mathcal{C}(n)}$ constitutes the learned multi-class classifiers at node n whereas total parameters for all internal nodes $\mathbf{W} = [\mathbf{W}_n]_{n \in \mathcal{N}}$ constitutes the learned model for Top-Down (TD) classifier. Label prediction for a test instance $\hat{\mathbf{x}}(i)$ is done in conjunction with the orphan node identification as shown in Algorithm 7.

7.3 Experimental Protocol

7.3.1 Datasets

We have used an extensive set of image and text datasets for evaluation purposes. Various statistics of the datasets used in our experiments are listed in Table 7.1. All these datasets are single labeled and the examples are assigned to the leaf nodes in the hierarchy. For text datasets, we have preprocessed the datasets by applying the tf-idf transformation with $l2$-norm normalization on the word-frequency feature vector. Further, these datasets do not have orphan nodes therefore for evaluation we randomly remove 1% of the classes from training dataset. Descriptions of the datasets used in our experiments are as follows:

CLEF [48] - Medical images annotated with medical applications codes. Each image is represented by 80 features extracted using local distribution of edges.

IPC[3] - Collection of patent documents organized in International Patent Classification (IPC) hierarchy.

DMOZ-SMALL, DMOZ-2010 and 2012[4] - Multiple web documents organized in the hierarchical structure. Datasets have been released as the part of the LSHTC[5] challenge in the year 2010, 2012. Since, we don't have access to true labels for test instances we split the train dataset into 80:20 ratio for evaluation.

7.3.2 Orphan Node Evaluation Metrics

We have used the standard set based performance measures μF_1 and MF_1 for evaluating the performance of learned models. However, for test dataset with orphan nodes, results are reported by considering only the *seed* classes *i.e.* having at least one training instance as reported in Dalvi *et al.* [12].

[3] http://www.wipo.int/classifications/ipc/en
[4] http://dmoz.org
[5] http://lshtc.iit.demokritos.gr

7.3.3 Methods for Comparison

We use various TD methods for comparison purpose.

Top-Down Logistic Regression (TD-LR): In this method, we use the original hierarchy for model learning. For predicting the labels of unlabeled test instances, we start at root node and recursively select the best child nodes until leaf node is reached. This is the baseline model used in our experiments.

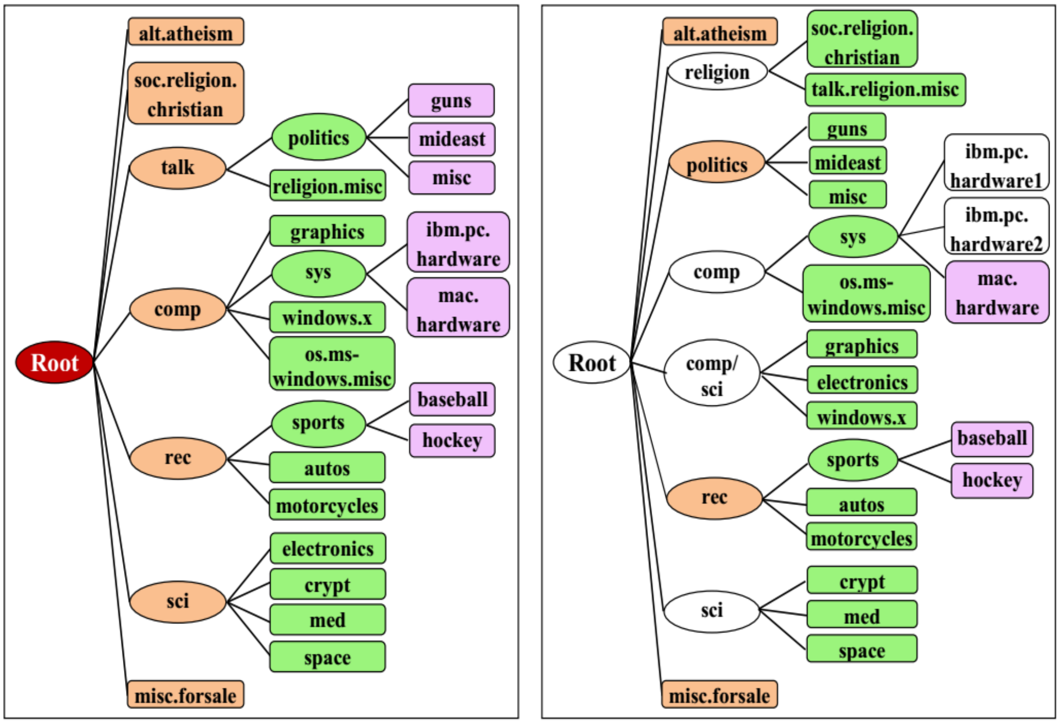
HC with Incomplete Class Hierarchies [12]: Similar to TD-LR, we use the original hierarchy for model learning. However, prediction for unlabeled test instances are done using Algorithm 7. This method has the advantage that it can detect the orphan nodes during the testing (prediction) phase. We refer to this method as TD-LR + ICH.

HC with Integrated Framework: This is our proposed method discussed in Figure 7.1. We refer to this method as TD-LR + IF.

7.3.4 Experimental Details

For all the experiments, we divide the training dataset into train and small validation dataset in the ratio 90:10. The train dataset is used to train TD model whereas the validation dataset is used to tune the parameters. The model is trained for a range of mis-classification penalty parameter (λ) values in the set $\{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ with best value selected using a validation dataset. Adapting the best parameter, we retrain the model on the entire training dataset and measure the performance on a separate held-out test dataset. For feature selection, we choose the best set of features $S_{\mathcal{F}}$ using the validation dataset. We use the liblinear solver[6] for efficient training of TD model.

[6] http://www.csie.ntu.edu.tw/∼cjlin/liblinear



(a) Original Hierarchy ($\mathcal{H}$)

(b) Modified Hierarchy ($\mathcal{H}_M$)

Figure 7.2: (a) Original (predefined) hierarchy and (b) Modified hierarchy obtained after: (i) Restructuring and (ii) Less cohesive or overlapping categories steps for Newsgroup dataset. Structural changes are marked in white color.

Table 7.2: Performance comparison with and without Orphan Node identification

Removed Nodes for Experiment	**Identified Orphan Nodes**	**Metric**	**Accuracy**	
			with Orphan Node	w/o Orphan Node
soc.religion.christian, guns, windows.x, electronics	*guns, electronics*	μF_1	**76.34** (0.0123)	75.23 (0.0182)
		MF_1	**76.00** (0.0224)	74.68 (0.0682)
windows.x, misc.forsale mideast, crypt	*misc.forsale, mideast, crypt*	μF_1	**76.48** (0.0624)	75.22 (0.0213)
		MF_1	**76.26** (0.0817)	74.64 (0.0518)

Table shows mean and (standard deviation) in bracket across five runs.

7.4 Results

7.4.1 Case Study

To understand the strength and effectiveness of each step in the integrated framework we perform case study on smaller Newsgroup[7] dataset containing 20 classes, 61188 features, 7505 test and 11269 training instances (evenly distributed across 20 classes). Figure 7.2(b) shows the modified hierarchy obtained after restructuring and less cohesive or overlapping categories on the original Newsgroup dataset shown in Figure 7.2(a). For evaluating each of this hierarchy we randomly choose five different sets of training and test split in the same ratio as original dataset. Modified hierarchy achieves an average MF_1 score of 82.16 as opposed to original hierarchy that achieves a score of 77.94. This result shows that modified hierarchy is more beneficial for HC.

To understand the benefits of feature selection step we perform experiments by varying the number of instances in each class between 5 and 50. Figure 7.3 shows the comparison of feature selection over using full set of features. Rare categories with 5-10 instances per class benefits significantly with feature selection. Moreover, the improvement in performance decreases as the number of instances per class increases because more training instances leads to better generalized models with full set of features. It should be noted that feature selection is also helpful in terms of training, prediction time and memory requirements to store model weight vectors.

Finally, to access the orphan node identification step, we performed experiments by

[7] http://qwone.com/∼jason/20Newsgroups

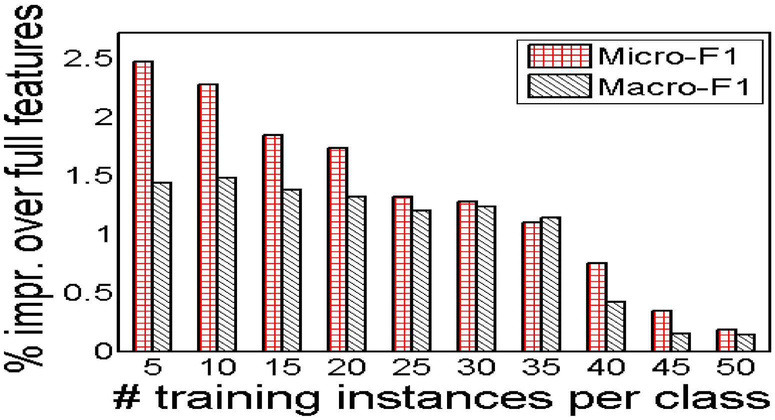


Figure 7.3: Performance comparison of feature selection with full features.

randomly removing four classes from training dataset. Results obtained with different sets of orphan nodes is shown in Table 7.2. Our approach effectively identifies a subset of orphan nodes which is helpful in improving the HC performance of seed classes.

7.4.2 Accuracy Comparison

Table 7.3 shows the μF_1 and MF_1 performance comparison of various models. Each experiment is repeated five times by randomly selecting the train and test dataset. Our proposed integrated framework consistently outperforms the other approaches across different datasets. To access the performance improvement we conducted pairwise statistical significance tests between the baseline model TD-LR and {TD-LR + ICH, TD-LR +IF} model. Specifically, we perform micro-sign test for μF_1[52] and non-parametric wilcoxon rank test for MF_1 scores (it should be noted that significance tests are between two models for single run). Statistical significant results are denoted by ▲(△) for significance with p-value 0.01 (0.05). We can see from the table that most of the results obtained are statistically significant with our proposed framework.

Baseline TD-LR model has the worse performance due to the presence of inconsistencies within the hierarchy along with rare categories, less cohesive or overlapping categories and orphan nodes. On comparison, TD-LR + ICH model is slightly better because this method

Table 7.3: μF_1 and MF_1 performance comparison

Dataset	**TD-LR** **(Baseline)**		**TD-LR + ICH** **(Dalvi *et al.* [12])**		**TD-LR + IF** **(Proposed)**	
	μF_1	MF_1	μF_1	MF_1	μF_1	MF_1
CLEF	73.06	33.86	73.56	34.13	**74.63**	**34.98**△
	(0.0231)	(0.0821)	(0.0526)	(0.0642)	(0.0521)	(0.0916)
IPC	47.24	41.25	48.92▲	43.00▲	**49.26▲**	**44.12▲**
	(0.2645)	(0.1737)	(0.2536)	(0.6150)	(0.1427)	(0.3638)
DMOZ-SMALL	40.12	23.63	40.29	24.17△	**42.34▲**	**26.23▲**
	(0.2913)	(0.1245)	(0.1734)	(0.2234)	(0.2236)	(0.1194)
DMOZ-2010	36.18	21.42	36.20	21.44	**38.24▲**	**23.11▲**
	(0.2437)	(0.1470)	(0.2522)	(0.0167)	(0.1243)	(0.1850)
DMOZ-2012	38.42	23.88	38.92	24.24△	**40.87▲**	**26.34▲**
	(0.1029)	(0.0324)	(0.0124)	(0.1142)	(0.0243)	(0.185)

Table shows mean and (standard deviation) in bracket across five runs. ▲ (△) indicates that improvements are statistically significant with 0.01 (0.05) significance level. Comparison is between TD-LR model and other models.

Table 7.4: Runtime (in mins) and memory requirements comparison

Dataset	**TD-LR** **(Baseline)**			**TD-LR + ICH** **(Dalvi *et al.* [12])**			**TD-LR + IF** **(Proposed)**		
	train	predict	memory	train	predict	memory	train	predict	memory
CLEF	0.28	< 1	27.52 KB	0.28	< 1	28.16 KB	3.12	< 1	19.00 KB
IPC	68.58	1.91	02.46 GB	68.58	2.31	02.50 GB	102.54	2.00	01.48 GB
DMOZ-SMALL	3.17	< 1	00.48 GB	3.17	1.24	00.50 GB	18.29	1.06	00.31 GB
DMOZ-2010	6418	20.60	26.10 GB	6418	28.24	26.56 GB	10826	24.08	15.95 GB
DMOZ-2012	19193	49.9	19.30 GB	19193	62.49	19.90 GB	28107	53.46	14.02 GB

can deal with orphan nodes. Our approach TD-LR + IF is best performing method because it can effectively deal with issues that are faced by other two models.

7.4.3 Runtime and Memory Comparison

Table 7.4 shows the average training, prediction time and memory comparison of various models. Our model (TD-LR + IF) has expensive training time due to overhead associated with feature selection and pairwise similarity computation steps. However, we would like to emphasize that both these steps are embarrassingly parallel. On comparing prediction time, our model and TD-LR + ICH model are expensive due to extra time required for orphan node identification. Comparatively, TD-LR + ICH is more expensive due to mixed integer programming being solved [12] for making the final predictions. Baseline TD-LR model takes least time because predicting label involves recursive selection of best child node along tree path only.

On comparing memory required to store model parameters, our approach requires the least space due to feature selection at each internal node which reduces the dimensions of learned model weight vectors and hence memory. In fact, for large-scale DMOZ-2010 and DMOZ-2012 datasets improvement is much more significant with ∼40% reduction in memory space.

7.5 Summary

In this work we propose an integrated framework to solve the major issues faced by large-scale HC. Unlike previous HC approaches, our framework can identify the orphan nodes during the testing phase. We also performed an extensive analysis with case study to determine the importance of each step in HC. Our experimental evaluation on various datasets shows that the proposed framework can achieve significant improvements in terms of accuracy and memory requirements. Our approach is scalable due to high degree of parallelization at each step that can be exploited to improve the training time of model learning.

Chapter 8: Conclusion and Future Work

8.1 Conclusion

To conclude, this thesis provides solution to two of the well known HC issues namely, hierarchical inconsistencies and rare categories. For hierarchical inconsistencies, inconsistent node flattening and rewiring approach are proposed. Flattening methods, that selectively removes the inconsistent nodes from the hierarchy based on the deviation from mean, are effective for improving the classification performance. However, they cannot deal with inconsistencies in different parts of the hierarchy resulting in limited performance improvement. In contrary, the rewiring approach can overcome this limitation by modifying the existing parent-child relationships based on the similarities between the classes resulting in better HC performance. For solving rare categories issue, different rank-based approaches have been proposed which results in better classification performance especially for classes with rare categories. Additionally, to solve the large-scale HC problem this thesis also discusses about various approaches for integrating state-of-the-art feature selection methods into HC framework which results in better accuracy and runtime while requiring lesser memory for storing model parameters. Finally, we discussed about an integrated approach to handle multiple issues together that are faced by HC resulting in improved performance while being scalable for larger datasets.

8.2 Future Work

8.2.1 Large-scale Multi-Task Learning

Multi-Task Learning (MTL) has been used effectively in many different applications [87–94]. It aims to learn generalized model by jointly learning multiple related tasks together.

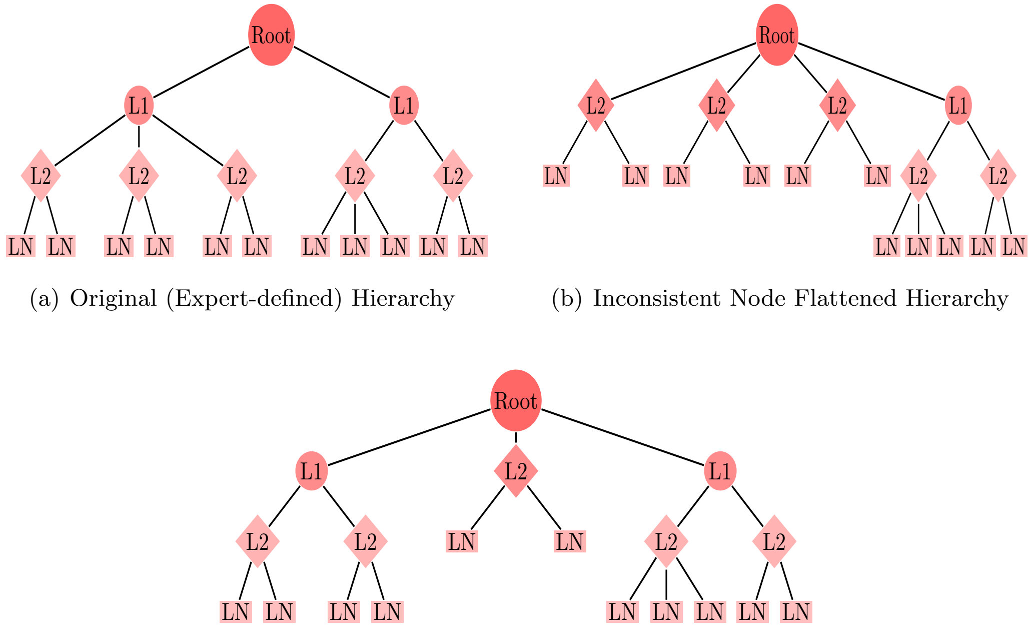
Intuition behind MTL methods is that the training signal present in related tasks can help each of the tasks learn better model. It also allows for learning of better models with fewer labeled examples (*i.e.* classes with rare categories). However, there are two major problem with MTL that needs to be addressed.

1. Determining related task to consider for joint optimization is difficult.
2. It is computationally expensive due to joint learning and cannot be scaled for large-scale problem.

Naik *et al.* [47] proposed a kNN based tanimoto similarity measure that can identify related task between two hierarchies (DMOZ and Wikipedia). However, the major problem associated with this approach is that k value is unified across all target DMOZ classes to determine the related task from source Wikipedia classes and vice-versa. Although, the improvement is observed with different values of k but there is no consensus regarding what value of k will work better for different datasets. To overcome this, we would like to explore more flexible ways to determine the best value of k for each target DMOZ classes. Specifically, we are interested in finding related tasks by projecting instances from each of the DMOZ classes to the Wikipedia classes. To find the optimal value of k for DMOZ classes we make use of Cover Ratio which is defined as follows:

Definition 2 (Cover Ratio)**.** *Given, two hierarchies represented by $\mathcal{H}_{S1}$ and $\mathcal{H}_{S2}$ with set of classes $\mathcal{C}_{S1}$ and $\mathcal{C}_{S2}$, respectively. Cover Ratio between i^{th} class of $\mathcal{H}_{S1}$ i.e. $\mathcal{C}^{i}_{S1}$ and j^{th} class of $\mathcal{H}_{S2}$ i.e. $\mathcal{C}^{j}_{S2}$ is defined as the percentage of instances from $\mathcal{C}^{i}_{S1}$ that projects into the $\mathcal{C}^{j}_{S2}$ class where projection is done by computing similarity between each of the instance belonging to class $\mathcal{C}^{i}_{S1}$ and the centroid of each class belonging to hierarchy $\mathcal{H}_{S2}$ and assigning the instance to a class having maximum similarity score.*

$$Cover\ Ratio(\mathcal{C}^{i}_{S1}, \mathcal{C}^{j}_{S2}) = \frac{\left| \underset{k \in \mathcal{C}^{i}_{S1}}{\forall} \ \mathbf{argmax}\Big(similarity(k, \mathcal{C}_{S2}) \Big) == j \right|}{|\mathcal{C}^{i}_{S1}|} \tag{8.1}$$



(a) Original (Expert-defined) Hierarchy

(b) Inconsistent Node Flattened Hierarchy

(c) Hierarchy with Partial Flattening of Inconsistent Node

Figure 8.1: Different hierarchical structures (b)-(c) obtained by flattening original (expert-defined) hierarchy (a).

For each DMOZ class, Wikipedia classes with Cover Ratio above certain specified threshold are considered as the related task for joint optimization in MTL. This approach if implemented will not only help to select optimal value of k for different target DMOZ classes but will also boost the accuracy performance by preventing negative *inductive* transfer between the identified k related tasks.

For reducing the runtime performance of MTL based methods, feature selection could be an effective tool [9, 15, 58]. Large-scale problem have high-dimensional features which increases the runtime between optimization iterations resulting in longer runtime. Feature selection will be helpful to reduce the dimensions of instances thereby reducing the time taken for each iteration and improving the runtime performance along with improved accuracy (by removing redundant and irrelevant features) and lesser memory to store the model parameters.

8.2.2 Partial Flattening of Inconsistent Nodes

Removing inconsistent nodes from the orginial (expert-defined) hierarchy are beneficial for improving the HC performance [2,4,31]. However, flattening all children of the inconsistent nodes may not be an optimal choice for classification. It is quiet possible that some of the children of inconsistent nodes are actually benefitting from these nodes by leveraging structural information (especially rare categories). However, due to overall optimization objective value, the node is identified as inconsistent and all its children are flattened. To overcome this drawback, we plan to perform more regressive analysis with partial flattening of inconsistent nodes where only the subset of children are flattened as shown in Figure 8.1.

Validation dataset can be used to identufy the subset of children that performs comparatively better with the inconsistent nodes presence.

8.2.3 Multi-linear Models

As stated earlier, Top-Down (TD) methods are effective for dealing with large-scale HC. However, the performance of TD may be poor due to error propagation *i.e.* errors made at the higher levels in the hierarchy cannot be corrected at the lower levels. To overcome this problem various methods have been proposed that modify hierarchy [4, 32] or combine multiple predictions [5, 7] for improving the classification performance. Still, the margin of errors at higher levels in the hierarchy is more compared to lower levels [2] because at higher levels each of the discriminative node consist of the multiple sub-categories which may not be easily separable with the linear classifiers. Alternatively, non-linear classifiers [95] can be used to overcome this problem but they are computationally expensive which makes them unsuitable for large-scale problem.

An ideal classifier must poses the classifying properties of non-linear classifiers while being computationally efficient like linear classifiers. Recently, multi-linear methods have been propsed by Huang and Lin [96] which address this issue for binary classification problem. Multi-linear models take advantage of both, linear and non-linear models. While multi-linear models are more accurate than linear models, it is also computationally efficient than non-linear models. One of the logical research extension of this work could be for multi-class HC problem.

8.2.4 Extreme Classification

Overtime, number of labels (categories) keeps on increasing. Extreme Classification is the problem of dealing with extremely large label spaces. To motivate consider the example of social media such as twitter where new hashtag is being created by the user frequently. Obviously, when new hashtag appears they do not have enough instances to train generalized models for classifying future tweets. In such cases it would be beneficial to fetch instances from other related hashtags. Problem that would be interesting to solve is how to determine the related hashtags from such a extremely large space of hashtags. Many works in this direction have been proposed in the literature [97–102]. Similar approach can be extended to the HC problem where labels (tags) can be organized into the hierarchy and mapping of unlabeled tags can be done easily by recursively selecting the best set of tags in the hierarchy.

At another level given streaming data the following situation can arise which will lead to interesting learning problems.

1. Assume, we are given streaming data with tweets. Before a hashtag gets popular, users may choose not to use a particular one of interest at all or make up their own. In this setting can we reassign previous tweets before the hashtag originated to the particular hashtag using classification.

2. Given, the limited data and treating hashtag assignments as an extreme classification problem can we use the data from step 1 to make predictions on future tweets. Can we know that a tweet generated needs a new hashtag that does not exist in the previous pool i.e., orphan prediction in the classification literature.

Bibliography

Bibliography

[1] T. Li, S. Zhu, and M. Ogihara, "Hierarchical document classification using automatically generated hierarchy," *Journal of Intelligent Information Systems*, vol. 29, no. 2, pp. 211–230, 2007.

[2] A. Naik and H. Rangwala, "Inconsistent node flattening for improving top-down hierarchical classification," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2016, pp. 379–388.

[3] C. N. Silla Jr and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 22, no. 1-2, pp. 31–72, 2011.

[4] R. Babbar, I. Partalas, E. Gaussier, and M.-R. Amini, "On flat versus hierarchical classification in large-scale taxonomies," in *Advances in Neural Information Processing Systems*, 2013, pp. 1824–1832.

[5] N. Cesa-Bianchi, C. Gentile, and L. Zaniboni, "Hierarchical classification: combining bayes with svm," in *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, 2006, pp. 177–184.

[6] S. Gopal and Y. Yang, "Recursive regularization for large-scale classification with hierarchical and graphical dependencies," in *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2013, pp. 257–265.

[7] P. N. Bennett and N. Nguyen, "Refined experts: improving classification in large taxonomies," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, 2009, pp. 11–18.

[8] L. Cai and T. Hofmann, "Hierarchical document categorization with support vector machines," in *Proceedings of the thirteenth ACM International Conference on Information and Knowledge Management*, 2004, pp. 78–87.

[9] D. Zhou, L. Xiao, and M. Wu, "Hierarchical classification via orthogonal transfer," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 801–808.

[10] A. Naik and H. Rangwala, "Filter based taxonomy modification for improving hierarchical classification," *http://arxiv.org/abs/1603.00772*, 2016.

[11] L. He and X. Sun, "Automatic maintenance of the category hierarchy," in *Proceedings of the International Conf, on Semantics, Knowledge and Grids*, 2013, pp. 218–221.

[12] B. Dalvi, A. Mishra, and W. W. Cohen, "Hierarchical semi-supervised classification with incomplete class hierarchies," in *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM)*, 2016, pp. 193–202.

[13] S. Gopal and Y. Yang, "Distributed training of large-scale logistic models." in *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013, pp. 289–297.

[14] S. Gopal, Y. Yang, B. Bai, and A. Niculescu-Mizil, "Bayesian models for large-scale hierarchical classification," in *Advances in Neural Information Processing Systems*, 2012, pp. 2411–2419.

[15] A. Naik and H. Rangwala, "Embedding feature selection for large-scale hierarchical classification," in *Proceedings of the IEEE International Conference on Big Data*, 2016.

[16] ——, "A ranking-based approach for hierarchical classification," in *IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, 2015, pp. 1–10.

[17] ——, "Integrated framework for improving large-scale hierarchical classification," in *submitted*, 2017.

[18] C.-J. Hsieh, K.-W. Chang, C.-J. Lin, S. S. Keerthi, and S. Sundararajan, "A dual coordinate descent method for large-scale linear svm," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008, pp. 408–415.

[19] K. Crammer and Y. Singer, "On the learnability and design of output codes for multiclass problems," *Machine learning*, vol. 47, no. 2-3, pp. 201–233, 2002.

[20] A. Zimek, F. Buchwald, E. Frank, and S. Kramer, "A study of hierarchical and flat classification of proteins," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 7, no. 3, pp. 563–571, 2010.

[21] M. Ceci and D. Malerba, "Classifying web documents in a hierarchy of categories: a comprehensive study," *Journal of Intelligent Information Systems*, vol. 28, no. 1, pp. 37–78, 2007.

[22] R. Eisner, B. Poulin, D. Szafron, P. Lu, and R. Greiner, "Improving protein function prediction using the hierarchical structure of the gene ontology," in *Proceedings of the 2005 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, 2005, pp. 1–10.

[23] T. Fagni and F. Sebastiani, "On the selection of negative examples for hierarchical text categorization," in *Proceedings of the 3rd Language and Technology Conference*, 2007.

[24] E. Costa, A. Lorena, A. Carvalho, and A. Freitas, "Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions," in *Advances in Bioinformatics and Computational Biology.* Springer, 2008, pp. 35–46.

[25] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng, "Improving text classification by shrinkage in a hierarchy of classes." in *Proceedings of the 15th International Conference on Machine Learning (ICML)*, vol. 98, 1998, pp. 359–367.

[26] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu, "Deep classification in large-scale text hierarchies," in *Proceedings of the 31st annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2008, pp. 619–626.

[27] L. Tang, J. Zhang, and H. Liu, "Acclimatizing taxonomic semantics for hierarchical content classification," in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2006, pp. 384–393.

[28] K. Nitta, "Improving taxonomies for large-scale hierarchical classifiers of web documents," in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, 2010, pp. 1649–1652.

[29] X. Qi and B. D. Davison, "Hierarchy evolution for improved classification," in *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, 2011, pp. 2193–2196.

[30] R. Babbar, I. Partalas, E. Gaussier, and M.-R. Amini, "Maximum-margin framework for training data synchronization in large-scale hierarchical classification," in *Neural Information Processing*, 2013, pp. 336–343.

[31] X.-L. Wang and B.-L. Lu, "Flatten hierarchies for large-scale hierarchical text categorization," in *Proceedings of the fifth International Conference on Digital Information Management (ICDIM)*, 2010, pp. 139–144.

[32] H. Malik, "Improving hierarchical svms by hierarchy flattening and lazy classification," in *Large-Scale Hierarchical Classification Workshop of ECIR*, 2010.

[33] Y. Yang, "An evaluation of statistical approaches to text categorization," *Information Retrieval*, vol. 1, no. 1-2, pp. 69–90, 1999.

[34] A. Kosmopoulos, I. Partalas, E. Gaussier, G. Paliouras, and I. Androutsopoulos, "Evaluation measures for hierarchical classification: a unified view and novel approaches," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 820–865.

[35] O. Dekel, J. Keshet, and Y. Singer, "Large margin hierarchical classification," in *Proceedings of the twenty-first International Conference on Machine Learning (ICML)*, 2004, p. 27.

[36] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)*, 1997, pp. 170–178.

[37] S. Dumais and H. Chen, "Hierarchical classification of web content," in *Proceedings of the 23rd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000, pp. 256–263.

[38] A. Sun and E.-P. Lim, "Hierarchical text classification and evaluation," in *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, 2001, pp. 521–528.

[39] L. Xiao, D. Zhou, and M. Wu, "Hierarchical classification via orthogonal transfer," in *Proceedings of the 28th International Conference on Machine Learning (ICML)*, 2011, pp. 801–808.

[40] T.-Y. Liu, Y. Yang, H. Wan, Q. Zhou, B. Gao, H.-J. Zeng, Z. Chen, and W.-Y. Ma, "An experimental study on large-scale web categorization," in *Special interest tracks and posters of the 14th international conference on World Wide Web*, 2005, pp. 1106–1107.

[41] A. Naik and H. Rangwala, "Large-scale hierarchical classification with rare categories and inconsistencies," *AI Matters*, vol. 2, no. 3, pp. 27–29, 2016.

[42] C. Vens, J. Struyf, L. Schietgat, S. Džeroski, and H. Blockeel, "Decision trees for hierarchical multi-label classification," *Machine Learning*, vol. 73, no. 2, pp. 185–214, 2008.

[43] O. Dekel, "Distribution-calibrated hierarchical classification," in *Advances in Neural Information Processing Systems*, 2009, pp. 450–458.

[44] T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma, "Support vector machines classification with a very large-scale taxonomy," *ACM SIGKDD Explorations Newsletter*, vol. 7, no. 1, pp. 36–43, 2005.

[45] C. Hadley and D. Jones, "A systematic comparison of protein structure classifications: Scop, cath and fssp," *Structure*, vol. 7, no. 9, pp. 1099–1112, 1999.

[46] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2011, pp. 2072–2079.

[47] A. Naik, A. Charuvaka, and H. Rangwala, "Classifying documents within multiple hierarchical datasets using multi-task learning," in *Proceedings of the IEEE 25th International Conference on Tools with Artificial Intelligence*, 2013, pp. 390–397.

[48] I. Dimitrovski, D. Kocev, S. Loskovska, and S. Džeroski, "Hierarchical annotation of medical images," *Pattern Recognition*, vol. 44, no. 10, pp. 2436–2449, 2011.

[49] ——, "Hierarchical classification of diatom images using predictive clustering trees," *Ecological Informatics*, vol. 7, pp. 19–29, 2012.

[50] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *Journal of Machine Learning Research*, vol. 9, no. Aug, pp. 1871–1874, 2008.

[51] R. Ghani, "Using error-correcting codes for text classification," in *Proceedings of the 17th International Conference on Machine Learning (ICML)*, 2000, pp. 303–310.

[52] Y. Yang and X. Liu, "A re-examination of text categorization methods," in *Proceedings of the 22nd annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1999, pp. 42–49.

[53] A. Charuvaka and H. Rangwala, “Hiercost: Improving large scale hierarchical classification with cost sensitive learning,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, 2015, pp. 675–690.

[54] K. Punera, S. Rajan, and J. Ghosh, “Automatically learning document taxonomies for hierarchical classification,” in *Special interest tracks and posters of the 14th International Conference on World Wide Web*, 2005, pp. 1010–1011.

[55] C. C. Aggarwal, S. C. Gates, and P. S. Yu, “On the merits of building categorization systems by supervised clustering,” in *Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 1999, pp. 352–356.

[56] S.-L. Chuang and L.-F. Chien, “A practical web-based approach to generating topic hierarchy for text segments,” in *Proceedings of the thirteenth ACM International Conference on Information and Knowledge Management*, 2004, pp. 127–136.

[57] M. Steinbach, L. Ertöz, and V. Kumar, “The challenges of clustering high dimensional data,” in *New directions in statistical physics.* Springer, 2004, pp. 273–309.

[58] B. Heisele, T. Serre, S. Prentice, and T. Poggio, “Hierarchical classification and feature reduction for fast face detection with support vector machines,” *Pattern Recognition*, vol. 36, no. 9, pp. 2007–2017, 2003.

[59] J. Tang, S. Alelyani, and H. Liu, “Feature selection for classification: A review,” *Data Classification: Algorithms and Applications*, p. 37, 2014.

[60] M. Dash and H. Liu, “Feature selection for classification,” *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[61] H. Peng, F. Long, and C. Ding, “Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[62] Z. Zheng, X. Wu, and R. Srihari, “Feature selection for text categorization on imbalanced data,” *ACM Sigkdd Explorations Newsletter*, vol. 6, no. 1, pp. 80–89, 2004.

[63] T. Joachims, “Text categorization with support vector machines: Learning with many relevant features,” in *European Conference on Machine Learning*, 1998, pp. 137–142.

[64] R. Kohavi and G. H. John, “Wrappers for feature subset selection,” *Artificial intelligence*, vol. 97, no. 1, pp. 273–324, 1997.

[65] P. Ristoski and H. Paulheim, “Feature selection in hierarchical feature spaces,” in *International Conference on Discovery Science.* Springer, 2014, pp. 288–300.

[66] W. Wibowo and H. E. Williams, “Simple and accurate feature selection for hierarchical categorisation,” in *Proceedings of the 2002 ACM symposium on Document Engineering*, 2002, pp. 111–118.

[67] H. Ogura, H. Amano, and M. Kondo, “Feature selection with a measure of deviations from poisson in text categorization,” *Expert Systems with Applications*, vol. 36, no. 3, pp. 6826–6832, 2009.

[68] W. Shang, H. Huang, H. Zhu, Y. Lin, Y. Qu, and Z. Wang, "A novel feature selection algorithm for text categorization," *Expert Systems with Applications*, vol. 33, no. 1, pp. 1–5, 2007.

[69] C. Ding and H. Peng, "Minimum redundancy feature selection from microarray gene expression data," *Journal of Bioinformatics and Computational Biology*, vol. 3, no. 02, pp. 185–205, 2005.

[70] C. Strobl and A. Zeileis, "Danger: High power! exploring the statistical properties of a test for random forest variable importance," in *In Proceedings of the 18th International Conference on Computational Statistics*, 2008.

[71] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *ACM SIGCOMM computer communication review*, vol. 29, no. 4, 1999, pp. 251–262.

[72] C. Burges, "From ranknet to lambdarank to lambdamart: An overview," *Microsoft Research Technical Report MSR-TR-2010-82*, 2010.

[73] A. Karatzoglou, L. Baltrunas, and Y. Shi, "Learning to rank for recommender systems," in *Proceedings of the 7th ACM conference on Recommender systems*, 2013, pp. 493–494.

[74] S. Agarwal, D. Dugar, and S. Sengupta, "Ranking chemical structures for drug discovery: a new machine learning approach," *Journal of chemical information and modeling*, vol. 50, no. 5, pp. 716–731, 2010.

[75] N. J. Risch, "Searching for genetic determinants in the new millennium," *Nature*, vol. 405, no. 6788, pp. 847–856, 2000.

[76] T.-Y. Liu, "Learning to rank for information retrieval," *Foundations and Trends in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[77] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Transactions on Information Systems (TOIS)*, vol. 7, no. 3, pp. 183–204, 1989.

[78] P. Li, Q. Wu, and C. J. Burges, "Mcrank: Learning to rank using multiple classification and gradient boosting," in *Advances in Neural Information Processing Systems*, 2007, pp. 897–904.

[79] K. Crammer, Y. Singer *et al.*, "Pranking with ranking." in *NIPS*, vol. 1, 2001, pp. 641–647.

[80] Y. Freund, R. Iyer, R. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *JMLR*, vol. 4, pp. 933–969, 2003.

[81] T. Joachims, "Optimizing search engines using clickthrough data," in *Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2002, pp. 133–142.

[82] D. Sculley, "Large scale learning to rank," in *NIPS 2009 Workshop on Advances in Ranking*, 2009.

[83] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th International Conference on Machine learning (ICML)*, 2007, pp. 129–136.

[84] J. Xu and H. Li, "Adarank: a boosting algorithm for information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, 2007, pp. 391–398.

[85] H. Narasimhan and S. Agarwal, "On the relationship between binary classification, bipartite ranking, and binary class probability estimation," in *Advances in NIPS*, 2013, pp. 2913–2921.

[86] B. Dalvi, W. W. Cohen, and J. Callan, "Exploratory learning," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD)*, 2013, pp. 128–143.

[87] L. Zhao, Q. Sun, J. Ye, F. Chen, C.-T. Lu, and N. Ramakrishnan, "Multi-task learning for spatio-temporal event forecasting," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2015, pp. 1503–1512.

[88] R. Caruana, "Multitask learning," in *Learning to learn*, 1998, pp. 95–133.

[89] J. Zhou, J. Chen, and J. Ye, "Malsar: Multi-task learning via structural regularization," *Arizona State University*, 2011.

[90] A. Naik, "Using multi-task learning for large-scale document classification," Master's thesis, George Mason University, Fairfax, Virginia (USA), 2013.

[91] F. Mordelet and J.-P. Vert, "Prodige: Prioritization of disease genes with multitask machine learning from positive and unlabeled examples," *BMC bioinformatics*, vol. 12, no. 1, p. 1, 2011.

[92] Y. Xue, X. Liao, L. Carin, and B. Krishnapuram, "Multi-task learning for classification with dirichlet process priors," *Journal of Machine Learning Research*, vol. 8, no. Jan, pp. 35–63, 2007.

[93] S. Bickel, J. Bogojeska, T. Lengauer, and T. Scheffer, "Multi-task learning for hiv therapy screening," in *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008, pp. 56–63.

[94] R. Caruana, "Algorithms and applications for multitask learning," in *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, 1996, pp. 87–95.

[95] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

[96] H.-Y. Huang and C.-J. Lin, "Linear and kernel classification: When to use which?" in *SIAM International Conference on Data Mining (SDM)*, 2016.

[97] Y. Prabhu and M. Varma, "Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning," in *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data mining*, 2014, pp. 263–272.

[98] P. Mineiro and N. Karampatziakis, "A hierarchical spectral method for extreme classification," *http://arxiv.org/abs/1511.03260*, 2016.

[99] O. Dekel and O. Shamir, "Multiclass-multilabel classification with more classes than examples." in *AISTATS*, 2010, pp. 137–144.

[100] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, "Sparse local embeddings for extreme multi-label classification," in *Advances in Neural Information Processing Systems*, 2015, pp. 730–738.

[101] H. Jain, Y. Prabhu, and M. Varma, "Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, pp. 935–944.

[102] D. Belanger and A. McCallum, "Structured prediction energy networks," *arXiv preprint arXiv:1511.06350*, 2015.

Curriculum Vitae

Azad Naik received his Bachelor's degree in Computer Science and Engineering from Indian Institute of Technology (ISM), Dhanbad, India in 2009. He received his Master's degree in Computer Science from George Mason University (GMU), Fairfax, VA, USA in 2013. He was employed as a Graduate Research Assistant in Data Mining laboratory during the course of his doctorate study and received a Doctor of Philosophy degree in Computer Science from GMU in 2017. Before joining GMU, he has worked as a Software Developer for 1 year at Aricent, Gurgaon, India and as a Senior Software Developer for over 1 year at Samsung India Software Operations (SISO), Bangalore, India. Upon graduation, he will be joining Microsoft at their head office in Redmond, Washington.