Large Scale Hierarchical Classification: Foundations, Algorithms and Applications

Huzefa Rangwala and Azad Naik

Department of Computer Science
MLBio+ Laboratory
Fairfax, Virginia, USA

SDM Tutorial, Miami, Florida

6th May, 2016
Part - I

1 Introduction and Background
   - Motivation
   - Hierarchical Classification (HC) problem description
   - Challenges
   - Methods for solving HC

2 State-of-the-Art HC Approaches
   - Parent-child regularization
   - Cost-sensitive learning
     - Package description/Software demo
Part - II

1 Inconsistent Hierarchy
   - Motivation
   - Methods for resolving inconsistency
   - Optimal hierarchy search in hierarchical space

2 Other HC Methods
   - Learning using multiple hierarchies
   - Extreme and deep classification

3 Conclusion
Motivation

- Exponential growth in data (image, text, video) over time
  - Big data era - megabytes & gigabytes to terabytes & petabytes
  - growth in almost all fields - astronomical, biological, web content
Data Organization

- Organize data into structure
  - tree, graph [LSHTC, BioASQ and ILSVRC challenge]

- Useful in various applications
  - query search, browsing and categorizing products
Hierarchical Structure

- Classes organized into the hierarchical structure
- **Generic** (↑) to **specific** (↓) categories in top-down order
Hierarchical Classification

Goal

Given hierarchy of classes exploit the hierarchical structure to learn models and classify unlabeled test examples (instances) to one or more nodes in the hierarchy.
Hierarchical Classification

Goal

Given hierarchy of classes exploit the hierarchical structure to learn models and classify unlabeled test examples (instances) to one or more nodes in the hierarchy

Solution

(i) Manual Classification
(ii) Automated Classification
Manual Classification

- Requires human understanding and expertise
- Infeasible for huge data

Diagram:
- Human Expert
- Hierarchy
- New Example
- Manual Classification
Automated Classification

- Trained expert (such as computer)
- Scalable for huge data
Challenges - I

Single label vs. multi-label

- **Single label classification** - each example belongs exclusively to one class only
- **Multi-label classification** - example may belong to more than one class
Mandatory leaf node vs. internal node prediction

- Example may be assigned to internal nodes

Mandatory leaf node prediction

Internal/Orphan node prediction
Mandatory leaf node vs. internal node prediction

- Example may be assigned to internal nodes
- **Orphan node** detection problem
Challenges - III

Rare categories

- Many classes with very few labeled examples
Challenges - III

Rare categories

- Many classes with very few labeled examples
- More prevalent in large scale datasets - $\geq 70\%$ have $\leq 10$ examples

![Diagram showing rare categories and their distribution in DMOZ datasets DMOZ-2010 and DMOZ-2012.](image)

# of training examples

Rare categories

- DMOZ dataset
- DMOZ-2010
- DMOZ-2012
Feature selection

- All features are **not essential** to **discriminate** between classes
- Identify features to improve classification performance
Feature selection

- All features are **not essential** to **discriminate** between classes
- Identify features to improve classification performance
Other Challenges

- **Parameter optimization**
  - incorporate relationships (*parent-child, siblings*) information
Other Challenges

- **Parameter optimization**
  - incorporate relationships *(parent-child, siblings)* information

- **Scalability**
  - large # of classes, features and examples require distributed computation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Training examples</th>
<th>#Leaf node (classes)</th>
<th>#Features</th>
<th>#Parameters</th>
<th>Parameter size (approx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMOZ-2010</td>
<td>128,710</td>
<td>12,294</td>
<td>381,580</td>
<td>4,652,986,520</td>
<td>18.5 GB</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>383,408</td>
<td>11,947</td>
<td>348,548</td>
<td>4,164,102,956</td>
<td>16.5 GB</td>
</tr>
</tbody>
</table>
Other Challenges

- **Parameter optimization**
  - incorporate relationships *(parent-child, siblings)* information

- **Scalability**
  - large # of classes, features and examples require **distributed computation**

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Training examples</th>
<th>#Leaf node (classes)</th>
<th>#Features</th>
<th>#Parameters</th>
<th>Parameter size (approx)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DMOZ-2010</td>
<td>128,710</td>
<td>12,294</td>
<td>381,580</td>
<td>4,652,986,520</td>
<td>18.5 GB</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>383,408</td>
<td>11,947</td>
<td>348,548</td>
<td>4,164,102,956</td>
<td>16.5 GB</td>
</tr>
</tbody>
</table>

- **Inconsistent hierarchy**
  - not suitable for classification (more details later)
### Notation

- $n = \#$ of training examples (instances)
- $D = \text{dimension of each instance}$
- $N = \text{set of nodes in the hierarchy}$
- $L = \text{set of leaf node (classes)}$
- $C(t) = \text{children of node } t$
- $\pi(t) = \text{parent of node } t$

#### Instance Matrix

$$X = \begin{bmatrix}
  x_1^1 & x_1^2 & \ldots & x_1^D \\
  x_2^1 & x_2^2 & \ldots & x_2^D \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  x_n^1 & x_n^2 & \ldots & x_n^D 
\end{bmatrix}$$

#### Label Matrix

$$Y = \begin{bmatrix}
  1 & 0 & \ldots & 1 & 0 \\
  0 & 1 & \ldots & 0 & 0 \\
  \vdots  & \vdots  & \ddots & \vdots  & \vdots  \\
  0 & 0 & \ldots & 1 & 1 \\
  1 & 0 & \ldots & 0 & 0 
\end{bmatrix}$$

#### Weight Matrix

$$W = \begin{bmatrix}
  w_1^1 & w_2^1 & \ldots & w_D^1 \\
  w_1^2 & w_2^2 & \ldots & w_D^2 \\
  \vdots  & \vdots  & \ddots & \vdots  \\
  w_1^L & w_2^L & \ldots & w_D^L 
\end{bmatrix}$$
Classification

**Training** - Learn mapping function using training data

\[
\begin{bmatrix}
(x_1, y_1) \\
(x_2, y_2) \\
\vdots \\
(x_n, y_n)
\end{bmatrix}
\]

**Testing** - Predict the label of test example

\[(x, ?) \rightarrow f(x) \rightarrow \hat{y}\]
Combination of two terms:

1. **Empirical loss** - controls how well the learnt models fits the training data
2. **Regularization** - prevent models from over-fitting and encodes additional information such as hierarchical relationships

\[
\min_{W} \mathcal{L}(f(X, W), Y) + \lambda \Omega(W)
\]
Different Approaches for Solving HC Problem

Hierarchical Classification (HC)

- Ignores Hierarchical Relationship Information
  - Flat Classification
    - Tzanetakis & Cook, SAP'07

- Utilizes Hierarchical Relationship Information
  - Local Relationship Information
    - Local Classifier per Node (LCN)
      - Fagni & Sebastiani, LTC'07
    - Local Classifier per Parent Node (LCPN)
      - Koller & Sahami, ICML'97
    - Local Classifier per Level (LCL)
      - Clare & King, BioInf.'03
  - Global Relationship Information
    - Global Classification
      - Gopal & Yang, SIGKDD'13
Flat Classification Approach

- Simplest method (ignores hierarchy)
- Learn discriminant classifiers for each leaf node in the hierarchy
- Unlabeled test example classified using the rule:

\[
\hat{y} = \arg \max_{y \in \mathcal{Y}} f(x, y|w)
\]
Local Classifier per Node (LCN)

- Learn binary classifiers for all non-root nodes
- Goal is to effectively discriminate between the siblings
- Top-down approach is followed for classifying unlabeled test examples
Local Classifier per Parent Node (LCPN)

- Learn multi-class classifiers for all non-leaf nodes
- Like LCN goal is to effectively discriminate between the siblings
- Top-down approach is followed for classifying unlabeled test examples
Local Classifier per Level (LCL)

- Learn multi-class classifiers for all levels in the hierarchy
- Least popular among local approaches
- Prediction inconsistency may occur and hence post-processing step is required
Global Classification Approach

- Learn global function considering all hierarchical relationships
- Often referred as Big-Bang approach
- Unlabeled test instance is classified using an approach similar to flat or local methods
Flat evaluation measures

- Misclassifications treated equally

Common evaluation metrics:

- **Micro-F1** - gives equal weightage to all examples, dominated by common class
- **Macro-F1** - gives equal weightage to each class
Hierarchical evaluation measures

- Hierarchical distance between the true and predicted class taken into consideration for performance evaluation

- Common evaluation metrics:
  - **Hierarchical-F1** - common ancestors between true and predicted class
  - **Tree Error** - average hierarchical distance b/w true and predicted class
Multi-Task Learning (MTL)

- Involves **joint training** of multiple related tasks to improve generalization performance.
- Independent learning problems can utilize the **shared knowledge**.
- Exploits **inductive biases** that are helpful to all the related tasks:
  - similar set of parameters
  - common feature space
- **Examples**:
  - personal email spam classification - many person with same spam
  - automated driving - brakes and accelerator
Motivation

- Traditional approach learn classifiers for each leaf node (task) to discriminate one class from other

\[
\min_{w_t} \frac{1}{2} ||w_t||^2_2 + C \sum_{i=1}^{n} \left[ 1 - Y_{it} w_T^T x_i \right]_+
\]

- Works well if:
  - Dataset is small
  - Balanced
  - Sufficient positive examples per class to learn generalized discriminant function

Drawbacks

- Real world datasets suffers from rare categories issue
  Remember: 70% classes have less than 10 examples per class

- Large number of classes (scalability issue)
Can we improve the performance of data sparse leaf nodes by taking advantage of data rich nodes at higher levels?

Incorporate inter-class dependencies to improve classification

- examples belonging to Soccer category is less likely to belong to Software category

\[
\min_{\mathbf{w}_t} \frac{1}{2} \| \mathbf{w}_t - \mathbf{w}_{\pi(t)} \|_2^2 + C \sum_{k \in C(t)} \sum_{i=1}^{n} \left[ 1 - Y_{ik} \mathbf{w}_t^T \mathbf{x}_i \right] + \]

Objective

- How to effectively incorporate the hierarchical relationships into the objective function to improve generalization performance

- Make it scalable for larger datasets
Proposed Formulation

- Enforces model parameters (weights) to be similar to the parent in regularization
- Proposed state-of-the-art: HR-SVM and HR-LR global formulation

**HR-SVM**

\[
\min_{\mathbf{W}} \sum_{t \in \mathcal{N}} \frac{1}{2} \| \mathbf{w}_t - \mathbf{w}_{\pi(t)} \|^2 + C \sum_{k \in \mathcal{L}} \sum_{i=1}^{n} \left[ 1 - Y_{ik} \mathbf{w}_k^T \mathbf{x}_i \right] +
\]

**Internal Node**

\[
\min_{\mathbf{w}_t} \frac{1}{2} \| \mathbf{w}_t - \mathbf{w}_{\pi(t)} \|^2 + \frac{1}{2} \sum_{c \in \mathcal{C}(t)} \| \mathbf{w}_c - \mathbf{w}_t \|^2
\]

**Leaf Node**

\[
\min_{\mathbf{w}_t} \frac{1}{2} \| \mathbf{w}_t - \mathbf{w}_{\pi(t)} \|^2 + \frac{1}{2} \sum_{i=1}^{n} \left[ 1 - Y_{it} \mathbf{w}_t^T \mathbf{x}_i \right] +
\]
HR-LR Models

- Similar formulation as **HR-SVM**
- **Logistic loss** instead of hinge loss

**HR-LR**

\[
\min_w \sum_{t \in \mathcal{N}} \frac{1}{2} \|w_t - w_{\pi(t)}\|_2^2 + C \sum_{k \in L} \sum_{i=1}^n \log(1 + \exp(-Y_{ik} w_k^T x_i))
\]

**Internal Node**

\[
\min_{w_t} \frac{1}{2} \|w_t - w_{\pi(t)}\|_2^2 + \frac{1}{2} \sum_{c \in C(t)} \|w_c - w_t\|_2^2
\]

**Leaf Node**

\[
\min_{w_t} \frac{1}{2} \|w_t - w_{\pi(t)}\|_2^2 + \frac{1}{2} \sum_{i=1}^n \log(1 + \exp(-Y_{it} w_t^T x_i))
\]
Proposed Parallel Implementation

- Each node is independent of all other nodes except its neighbours
- Objective function is block separable. Therefore, Parallel Block Coordinate Descent (CD) can be used for optimization

1. Fix odd-levels parameters, optimize even-levels in parallel
2. Fix even-levels parameters, optimize odd-levels in parallel
3. Repeat until convergence

Extended to graph by first finding the minimum graph coloring [NP-hard] and repeatedly optimizing nodes with the same color in parallel during each iteration
Experiments

Dataset description

- Wide range of single and multi-label dataset with varying number of features and categories were used for model evaluation

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Features</th>
<th># Categories</th>
<th>Type</th>
<th>Avg # labels (per instance)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEF</td>
<td>89</td>
<td>87</td>
<td>Single-label</td>
<td>1</td>
</tr>
<tr>
<td>RCV1</td>
<td>48,734</td>
<td>137</td>
<td>Multi-label</td>
<td>3.18</td>
</tr>
<tr>
<td>IPC</td>
<td>541,869</td>
<td>552</td>
<td>Single-label</td>
<td>1</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td>51,033</td>
<td>1,563</td>
<td>Single-label</td>
<td>1</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td>381,580</td>
<td>15,358</td>
<td>Single-label</td>
<td>1</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>348,548</td>
<td>13,347</td>
<td>Single-label</td>
<td>1</td>
</tr>
<tr>
<td>DMOZ-2011</td>
<td>594,158</td>
<td>27,875</td>
<td>Multi-label</td>
<td>1.03</td>
</tr>
<tr>
<td>SWIKI-2011</td>
<td>346,299</td>
<td>50,312</td>
<td>Multi-label</td>
<td>1.85</td>
</tr>
<tr>
<td>LWIKI</td>
<td>1,617,899</td>
<td>614,428</td>
<td>Multi-label</td>
<td>3.26</td>
</tr>
</tbody>
</table>

Table: Dataset statistics
Comparison Methods

Flat baselines

- **SVM** - one-vs-rest binary support vector machines
- **LR** - one-vs-rest regularized logistic regression

Hierarchical baselines

- **Top-down SVM (TD)**[Liu et al., SIGKDD’05] - a pachinko machine style SVM
- **Hierarchical SVM (HSVM)**[Tsochantaridis et al., JMLR’05] - a large-margin discriminative method with path dependent discriminant function
- **Hierarchical Orthogonal Transfer (OT)**[Lin et al., ICML’11] - a large-margin method enforcing orthogonality between the parent and the children
- **Hierarchical Bayesian Logistic Regression (HBLR)**[Gopal et al., NIPS’12] - a bayesian methods to model hierarchical dependencies among class labels using multivariate logistic regression
Figure: Performance improvement: HR-SVM vs. SVM
Figure: Performance improvement: HR-LR vs. LR
## Hierarchical Baselines Comparison

<table>
<thead>
<tr>
<th>Datasets</th>
<th>HR-SVM</th>
<th>HR-LR</th>
<th>TD</th>
<th>HSVM</th>
<th>OT</th>
<th>HBLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEF</td>
<td>80.02</td>
<td>80.12</td>
<td>70.11</td>
<td>79.72</td>
<td>73.84</td>
<td>81.41</td>
</tr>
<tr>
<td>RCV1</td>
<td>81.66</td>
<td>81.23</td>
<td>71.34</td>
<td>NA</td>
<td>NS</td>
<td>NA</td>
</tr>
<tr>
<td>IPC</td>
<td>54.26</td>
<td>55.37</td>
<td>50.34</td>
<td>NS</td>
<td>NS</td>
<td>56.02</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td>45.31</td>
<td>45.11</td>
<td>38.48</td>
<td>39.66</td>
<td>37.12</td>
<td>46.03</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td>46.02</td>
<td>45.84</td>
<td>38.64</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>57.17</td>
<td>53.18</td>
<td>55.14</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>DMOZ-2011</td>
<td>43.73</td>
<td>42.27</td>
<td>35.91</td>
<td>NA</td>
<td>NS</td>
<td>NA</td>
</tr>
<tr>
<td>SWIKI-2011</td>
<td>41.79</td>
<td>40.99</td>
<td>36.65</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>LWIKI</td>
<td>38.08</td>
<td>37.67</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

[NA - Not Applicable; NS - Not Scalable]

**Table:** Micro-F1 performance comparison
Runtime Comparison - flat baselines

HR-SVM vs. SVM

HR-LR vs. LR
## Runtime Comparison - hierarchical baselines

<table>
<thead>
<tr>
<th>Datasets</th>
<th>HR-SVM</th>
<th>HR-LR</th>
<th>TD</th>
<th>HSVM</th>
<th>OT</th>
<th>HBLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEF</td>
<td>0.42</td>
<td>1.02</td>
<td>0.13</td>
<td>3.19</td>
<td>1.31</td>
<td>3.05</td>
</tr>
<tr>
<td>RCV1</td>
<td>0.55</td>
<td>11.74</td>
<td>0.21</td>
<td>NA</td>
<td>NS</td>
<td>NA</td>
</tr>
<tr>
<td>IPC</td>
<td>6.81</td>
<td>15.91</td>
<td>2.21</td>
<td>NS</td>
<td>NS</td>
<td>31.20</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td>0.52</td>
<td>3.73</td>
<td>0.11</td>
<td>289.60</td>
<td>132.34</td>
<td>5.22</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td>8.23</td>
<td>123.22</td>
<td>3.97</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>36.66</td>
<td>229.73</td>
<td>12.49</td>
<td>NS</td>
<td>NS</td>
<td>NS</td>
</tr>
<tr>
<td>DMOZ-2011</td>
<td>58.31</td>
<td>248.07</td>
<td>16.39</td>
<td>NA</td>
<td>NS</td>
<td>NA</td>
</tr>
<tr>
<td>SWIKI-2011</td>
<td>89.23</td>
<td>296.87</td>
<td>21.34</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>LWIKI</td>
<td>2230.54</td>
<td>7282.09</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

[NA - Not Applicable; NS - Not Scalable]

**Table:** Training runtime comparison (in mins)
Motivation

- Drawbacks of Recursive Regularization
  - scalable, but more expensive to train than flat classification
  - requires specialized implementation and communication between processing node
  - Does not deal with class imbalance directly

Objective

- Decouple models so that they can be trained in parallel without dependencies between models
- Account for class imbalance in the optimization framework
Hierarchical Regularization Re-examination - I

Huzefa Rangwala and Azad Naik
George Mason University

6th May, 2016

Hierarchical Regularization

\[
\min_{\mathbf{W}} \mathcal{L}(f(\mathbf{X}, \mathbf{W}), \mathbf{Y}) + \lambda \Omega(\mathbf{W})
\]

Empirical Loss
Regularization

Regularizer promotes similarity

One-vs-rest classification

# vs (+, x)
+ vs (#, x)
x vs (+, #)

Loss promotes dissimilarity
Opposing learning influences:

- **loss term** - model for a node is forced to be dissimilar to all other nodes
- **regularization term** - model is forced to be similar to its neighbors; greater similarity to nearer neighbors

Resultant effect:

- Mistakes on negative examples that come from near nodes is less severe than those coming from far nodes while still taking advantage of the hierarchy
Consider the loss term for class "t" which is separable over examples

\[ \sum_i \text{loss}(y_i, w_i^T x_i) \]

Each loss value is multiplied by importance of the example for this class

\[ \sum_i \text{loss}(y_i, w_i^T x_i) \times \phi(t, y_i) \]

This is an example of "instance-based" cost sensitive learning

\[ c_i^t = \phi(t, y_1) \]
Hierarchical Costs

How to define costs based on hierarchy?

- **Tree Distance (TrD)** - undirected graph distance between nodes

- **Number Common Ancestors (NCA)** - the number of ancestors in common to target class and class label

- **Exponentiated Tree Distance (ExTrD)** - squash tree distance into a suitable range using validation
Imbalance Costs

- Using the same formulation of cost-sensitive learning, data imbalance can also be addressed

\[ c_i = 1 + \frac{L}{1 + \exp|n - n_0|} \]

- Due to very large skew, inverse class size can result in extremely large weights. Fix using squashing function shown in Fig.

- Multiply to combine with Hierarchical costs

\[ n_i = \text{num examples} \]
\[ n_0, L = \text{user defined constants} \]
Experiments

Dataset
- For comparison purpose same dataset has been used as proposed in the paper [Gopal and Yang, SIGKDD’13]

Comparison Methods
Flat baseline
- LR - one-vs-rest binary logistic regression is used in the conventional flat classification setting

Hierarchical baselines
- Top-down Logistic Regression (TD-LR) - one-vs-rest multi-class classifier trained at each internal node
- HR-LR [Gopal and Yang, SIGKDD’13] - a recursive regularization approach based on hierarchical relationships
## Results (Hierarchical Costs)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Micro-F1 (↑)</th>
<th>Macro-F1 (↑)</th>
<th>hF1 (↑)</th>
<th>TE (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLEF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>79.82</td>
<td>53.45</td>
<td>85.24</td>
<td>0.994</td>
</tr>
<tr>
<td>TrD</td>
<td>80.02</td>
<td>55.51</td>
<td>85.39</td>
<td>0.984</td>
</tr>
<tr>
<td>NCA</td>
<td>80.02</td>
<td>57.48</td>
<td>85.34</td>
<td>0.986</td>
</tr>
<tr>
<td>ExTrD</td>
<td><strong>80.22</strong></td>
<td><strong>57.55†</strong></td>
<td>85.34</td>
<td><strong>0.982</strong></td>
</tr>
<tr>
<td><strong>DMOZ-SMALL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>46.39</td>
<td>30.20</td>
<td>67.00</td>
<td>3.569</td>
</tr>
<tr>
<td>TrD</td>
<td><strong>47.52‡</strong></td>
<td><strong>31.37‡</strong></td>
<td><strong>68.26</strong></td>
<td><strong>3.449</strong></td>
</tr>
<tr>
<td>NCA</td>
<td><strong>47.36‡</strong></td>
<td><strong>31.20‡</strong></td>
<td>68.12</td>
<td>3.460</td>
</tr>
<tr>
<td>ExTrD</td>
<td><strong>47.36‡</strong></td>
<td><strong>31.19‡</strong></td>
<td>68.20</td>
<td>3.456</td>
</tr>
<tr>
<td><strong>IPC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>55.04</td>
<td>48.99</td>
<td>72.82</td>
<td>1.974</td>
</tr>
<tr>
<td>TrD</td>
<td><strong>55.24‡</strong></td>
<td><strong>50.20‡</strong></td>
<td>73.21</td>
<td>1.954</td>
</tr>
<tr>
<td>NCA</td>
<td><strong>55.33‡</strong></td>
<td><strong>50.29‡</strong></td>
<td><strong>73.28</strong></td>
<td><strong>1.949</strong></td>
</tr>
<tr>
<td>ExTrD</td>
<td><strong>55.31‡</strong></td>
<td><strong>50.29‡</strong></td>
<td>73.26</td>
<td>1.951</td>
</tr>
<tr>
<td><strong>RCV1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>LR</td>
<td>78.43</td>
<td>60.37</td>
<td>80.16</td>
<td>0.534</td>
</tr>
<tr>
<td>TrD</td>
<td><strong>79.46‡</strong></td>
<td>60.61</td>
<td>82.83</td>
<td>0.451</td>
</tr>
<tr>
<td>NCA</td>
<td><strong>79.74‡</strong></td>
<td>60.76</td>
<td><strong>83.11</strong></td>
<td><strong>0.442</strong></td>
</tr>
<tr>
<td>ExTrD</td>
<td><strong>79.33‡</strong></td>
<td><strong>61.74†</strong></td>
<td>82.91</td>
<td>0.466</td>
</tr>
</tbody>
</table>

**Table:** Performance comparison of hierarchical costs
## Results (Imbalance Costs)

<table>
<thead>
<tr>
<th>Datasets</th>
<th>IMB + LR</th>
<th>IMB + TrD</th>
<th>IMB + NCA</th>
<th>IMB + ExTrD</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CLEF</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMB + LR</td>
<td>79.52</td>
<td>79.92</td>
<td>79.62</td>
<td>80.32</td>
</tr>
<tr>
<td></td>
<td>53.11</td>
<td>52.84</td>
<td>51.89</td>
<td>58.45</td>
</tr>
<tr>
<td>hF1</td>
<td>85.19</td>
<td>85.59</td>
<td>85.34</td>
<td>85.69</td>
</tr>
<tr>
<td>TE</td>
<td>1.002</td>
<td>0.978</td>
<td>0.994</td>
<td>0.966</td>
</tr>
<tr>
<td><strong>DMOZ-SMALL</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMB + LR</td>
<td>48.55‡</td>
<td>49.03‡</td>
<td>48.87‡</td>
<td>tbf49.03‡</td>
</tr>
<tr>
<td></td>
<td>32.72‡</td>
<td>33.21‡</td>
<td>33.27‡</td>
<td>33.34‡</td>
</tr>
<tr>
<td>hF1</td>
<td>68.62</td>
<td>69.41</td>
<td>69.37</td>
<td>69.54</td>
</tr>
<tr>
<td>TE</td>
<td>3.406</td>
<td>3.334</td>
<td>3.335</td>
<td>3.322</td>
</tr>
<tr>
<td><strong>IPC</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMB + LR</td>
<td>55.04</td>
<td>55.60‡</td>
<td>55.33</td>
<td>55.67‡</td>
</tr>
<tr>
<td></td>
<td>49.00</td>
<td>50.45†</td>
<td>50.29</td>
<td>50.42</td>
</tr>
<tr>
<td>hF1</td>
<td>72.82</td>
<td>73.56</td>
<td>73.28</td>
<td>73.58</td>
</tr>
<tr>
<td>TE</td>
<td>1.974</td>
<td>1.933</td>
<td>1.949</td>
<td>1.931</td>
</tr>
<tr>
<td><strong>RCV1</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IMB + LR</td>
<td>78.59‡</td>
<td>79.63‡</td>
<td>79.61</td>
<td>79.22</td>
</tr>
<tr>
<td></td>
<td>60.77</td>
<td>61.04</td>
<td>61.04</td>
<td>61.33</td>
</tr>
<tr>
<td>hF1</td>
<td>81.27</td>
<td>83.13</td>
<td>82.65</td>
<td>82.89</td>
</tr>
<tr>
<td>TE</td>
<td>0.511</td>
<td>0.435</td>
<td>0.458</td>
<td>0.469</td>
</tr>
</tbody>
</table>

**Table:** Performance comparison with imbalance cost included
<table>
<thead>
<tr>
<th>Datasets</th>
<th>Micro-F1 (↑)</th>
<th>Macro-F1 (↑)</th>
<th>hF1 (↑)</th>
<th>TE (↓)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEF</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TD-LR</td>
<td>73.06</td>
<td>34.47</td>
<td>79.32</td>
<td>1.366</td>
</tr>
<tr>
<td>LR</td>
<td>79.82</td>
<td>53.45</td>
<td>85.24</td>
<td>0.994</td>
</tr>
<tr>
<td>HR-LR</td>
<td>80.12</td>
<td>55.83</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>HierCost</td>
<td><strong>80.32</strong></td>
<td><strong>58.45†</strong></td>
<td><strong>85.69</strong></td>
<td><strong>0.966</strong></td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TD-LR</td>
<td>40.90</td>
<td>24.15</td>
<td><strong>69.99</strong></td>
<td><strong>3.147</strong></td>
</tr>
<tr>
<td>LR</td>
<td>46.39</td>
<td>30.20</td>
<td>67.00</td>
<td>3.569</td>
</tr>
<tr>
<td>HR-LR</td>
<td>45.11</td>
<td>28.48</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>HierCost</td>
<td><strong>49.03‡</strong></td>
<td><strong>33.34‡</strong></td>
<td>69.54</td>
<td>3.322</td>
</tr>
<tr>
<td>IPC</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TD-LR</td>
<td>50.22</td>
<td>43.87</td>
<td>69.33</td>
<td>2.210</td>
</tr>
<tr>
<td>LR</td>
<td>55.04</td>
<td>48.99</td>
<td>72.82</td>
<td>1.974</td>
</tr>
<tr>
<td>HR-LR</td>
<td>55.37</td>
<td>49.60</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>HierCost</td>
<td><strong>55.67‡</strong></td>
<td><strong>50.42†</strong></td>
<td><strong>73.58</strong></td>
<td><strong>1.931</strong></td>
</tr>
<tr>
<td>RCV1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TD-LR</td>
<td>77.85</td>
<td>57.80</td>
<td><strong>88.78</strong></td>
<td><strong>0.524</strong></td>
</tr>
<tr>
<td>LR</td>
<td>78.43</td>
<td>60.37</td>
<td>80.16</td>
<td>0.534</td>
</tr>
<tr>
<td>HR-LR</td>
<td><strong>81.23</strong></td>
<td>55.81</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>HierCost</td>
<td><strong>79.22‡</strong></td>
<td><strong>61.33</strong></td>
<td>82.89</td>
<td><strong>0.469</strong></td>
</tr>
</tbody>
</table>

**Table:** Performance comparison of HierCost with other baseline methods
## Runtime comparison

<table>
<thead>
<tr>
<th>Datasets</th>
<th>TD-LR</th>
<th>LR</th>
<th>HierCost</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEF</td>
<td>&lt;1</td>
<td>&lt;1</td>
<td>&lt;1</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td>4</td>
<td>41</td>
<td>40</td>
</tr>
<tr>
<td>IPC</td>
<td>27</td>
<td>643</td>
<td>453</td>
</tr>
<tr>
<td>RCV1</td>
<td>20</td>
<td>29</td>
<td>48</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td>196</td>
<td>15191</td>
<td>20174</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>384</td>
<td>46044</td>
<td>50253</td>
</tr>
</tbody>
</table>

**Table:** Total training runtimes (in mins)
Demo of Software

- Freely available for research and education purpose at:
  - [https://cs.gmu.edu/~mlbio/HierCost/](https://cs.gmu.edu/~mlbio/HierCost/)

- Software: implemented in python using scikit-learn machine learning and svmlight-loader package

- Other prerequisite package:
  - numpy
  - scipy
  - networkx
  - pandas
Two main CLI is exposed for easy training and classification

**train.py**
- `-d` : train data file path
- `-t` : hierarchy file path
- `-m` : path to save learned model parameters
- `-f` : number of features
- `-r` : regularization parameter (0; default = 1)
- `-i` : to incorporate imbalance cost
- `-c` : cost function to use (lr, trd, nca, etrd) (default = 'lr')
- `-u` : for multi-label classification (default = single-label classification)
- `-n` : set of nodes to train (for parallelization)

**predict.py**
- `-p` : path to save prediction for test examples
- `-m`, `-d`, `-t`, `-f`, `-u` : similar functionalities
Part II

Inconsistent Hierarchy
(5 minutes break)
Motivation

Predefined Hierarchy

- Hierarchy defined by the domain experts
- Reflects human-view of the domain - may not be optimal for machine learning classification algorithms
Case Study

Question

- Can we trust the predefined expert’s hierarchy for achieving the good classification performance?
- Can we tweak (adjust) the hierarchy to improve the performance?

Answer

Case study on subset of newsgroup dataset

“Adjusted hierarchy classification performance comparatively better than semantically sound hierarchy”
Case Study

Question
- Can we trust the predefined expert’s hierarchy for achieving the good classification performance?
- Can we tweak (adjust) the hierarchy to improve the performance?

Answer
- Case study on subset of newsgroup dataset

"Adjusted hierarchy classification performance **comparatively better** than semantically sound hierarchy"
Hierarchy is designed for the sole purpose of easy search and navigation without taking classification into consideration.
Hierarchy is designed for the sole purpose of **easy search and navigation** without taking classification into consideration.

Hierarchy is created based on semantics which is independent of data characteristics whereas classification depends on **data characteristics** such as term frequency.

“Our expectation: **data-driven hierarchy** can be much powerful”
Hierarchy is designed for the sole purpose of easy search and navigation without taking classification into consideration

Hierarchy is created based on semantics which is independent of data whereas classification depends on data characteristics such as term frequency

"Our expectation: data-driven hierarchy can be much powerful”

Apriori it is not clear to domain experts when to generate new nodes (hierarchy expansion) or merge two or more nodes (link creation) in the hierarchy
Given list of categories: different experts may come up with different hierarchies with completely different classification results
Given list of categories: different experts may come up with **different hierarchies** with completely different classification results.

Large number of classes with confusing labels pose a unique challenge for manual design of good hierarchy.
Dynamic changes can affect hierarchical relationships

“Flood” is the sub-group of geography class but during chennai flood it becomes political news

2015: Chennai FLOOD Image

Sub-group of Geography class

During chennai flood becomes sub-group of Political class
What we want?

“Predefined Hierarchy”
What we want?

“Predefined Hierarchy”

↓

“Data-driven Hierarchy”

(for improving classification performance)
Motivation

- For large scale datasets top-down (TD) hierarchical models are preferred over flat models due to computational benefits (training and prediction time).
- TD models performance suffers due to error propagation i.e. compounding of errors from misclassifications at higher levels which cannot be rectified at the lower levels.

Objective

- Modify predefined hierarchy by removing (flattening) inconsistent nodes to improve the classification performance of TD models.
- Reduces top-down error propagation due to less number of decisions for classifying unlabeled example.
Flatten Hierarchies, Wang and Lu, ICDIM’10

Level Flattening Techniques

- Single or multiple levels within the hierarchy is flattened
- Based on level(s) flattened various methods exist, for e.g. TLF, BLF, MLF

- Drawback - all nodes in the level(s) is identified as inconsistent which may not be true; resulting in poor classification performance
Selected Inconsistent Node Removal (Flattening)

- Rather than flattening entire level(s) only **subset** of the inconsistent nodes are removed from the hierarchy

![Original Hierarchy](image1)
![Inconsistent Node Removed](image2)
![Final Hierarchy](image3)

- Criterion to decide inconsistent nodes - degree of error made at the node, margin-based or learning-based strategy
- Comparatively better performance than level flattening methods
Local Approach for INR (Level-INR)

- Inconsistent set of nodes determined for each level based on loss function values (such as logistic loss) obtained for nodes at that level.
- Criterion for flattening nodes - mean and standard deviation per level.
- Different levels have different threshold for node flattening.
Global Approach for INR (Global-INR)

- Inconsistent node determined by considering loss function of all internal nodes in the hierarchy
- Criterion for flattening nodes - mean and standard deviation of all nodes
- All levels have same threshold for node flattening

Original

Global-INR

Modified Hierarchy
Comparison Methods

- One-vs-rest models is trained for each node (except root) in the hierarchy
- Predictions are made starting from the root node and recursively selecting the best child nodes until a leaf node is reached

Hierarchical baselines

- **Top-down Logistic Regression (TD-LR)** - predefined hierarchy used for training the models
- **Level Flattening** [Wang and Lu, ICDIM’10] - flattened hierarchy used for training the models, based on level flattened we have:
  - TLF - Top level flattened
  - BLF - Bottom level flattened and
  - MLF - Multiple level flattened
- **MTA** [Babbar et al., NIP’13] - hierarchy is modified using the margin value computed at each node
Performance Results

- Level-INR and Global-INR performed comparatively better than other TD hierarchical baselines.
- Global approach has better performance compared to local approach.

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Top-down Hierarchical Baselines</th>
<th>Proposed Models</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TD-LR</td>
<td>TLF</td>
</tr>
<tr>
<td>CLEF</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>75.84</td>
</tr>
<tr>
<td></td>
<td>$MF_1(\uparrow)$</td>
<td>35.92</td>
</tr>
<tr>
<td>DIATOMS</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>56.93</td>
</tr>
<tr>
<td></td>
<td>$MF_1(\uparrow)$</td>
<td>44.46</td>
</tr>
<tr>
<td>IPC</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>51.28</td>
</tr>
<tr>
<td></td>
<td>$MF_1(\uparrow)$</td>
<td>42.51</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>45.48</td>
</tr>
<tr>
<td></td>
<td>$MF_1(\uparrow)$</td>
<td>30.65</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>41.32</td>
</tr>
<tr>
<td></td>
<td>$MF_1(\uparrow)$</td>
<td>28.37</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>50.32</td>
</tr>
<tr>
<td></td>
<td>$MF_1(\uparrow)$</td>
<td>29.89</td>
</tr>
</tbody>
</table>
Drawbacks of Flattening Strategy

- Flattening strategy although useful up to certain extent has few limitations
  - Inability to deal with inconsistencies in different branches of the hierarchy

- Rewiring strategy can be used to resolve inconsistencies that occur in different branch
Hierarchy Adjustment using Elementary Operation - I, Tang et al., SIGKDD’06

- Elementary operation: promote, demote, merge
**Assumption:** the optimal hierarchy is near the neighborhood of predefined taxonomy

Search for **constrained optimal hierarchy** by applying sequence of elementary operations and searching in the hierarchy space.
Proposed Hierarchy Adjustment Algorithm

- Wrapper based approach for hierarchy modification; requires hierarchy evaluation after each modification which is computationally expensive

**Input:** Predefined hierarchy \((H_0)\), Training data \((D_t)\), Validation data \((D_v)\)

1. Generate neighbor hierarchies for \(H_0\)
2. Train hierarchical classification models for each neighbor on \(D_t\)
3. Evaluate hierarchical classifiers on \(D_v\)
4. Pick the best neighbor hierarchy as \(H_0\)
5. Until no improvement, repeat from step 1
Sub-branch from AOL database hierarchy is used for evaluation

Data is small \textit{w.r.t} no. of features and classes

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#Total Node</th>
<th>#Leaf Node</th>
<th>Height</th>
<th>#Training</th>
<th>#Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Soc</td>
<td>83</td>
<td>69</td>
<td>4</td>
<td>5,248</td>
<td>34,003</td>
</tr>
<tr>
<td>Kids</td>
<td>299</td>
<td>244</td>
<td>5</td>
<td>15,795</td>
<td>48,115</td>
</tr>
</tbody>
</table>

\textbf{Table: Dataset Statistics}
Adjusted hierarchy shows significant performance improvement in comparison to predefined (original) hierarchy and hierarchy generated using clustering approach.

**Figure:** Soc (left) and Kids (right) dataset results
Other Evaluation Results

<table>
<thead>
<tr>
<th>Fold</th>
<th>Iterations</th>
<th>Evaluations</th>
<th>Operations</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6</td>
<td>582</td>
<td>43</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>6</td>
<td>585</td>
<td>37</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>265</td>
<td>33</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>449</td>
<td>41</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>805</td>
<td>67</td>
<td>13</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>754</td>
<td>68</td>
<td>6</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>448</td>
<td>48</td>
<td>9</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>494</td>
<td>42</td>
<td>11</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>268</td>
<td>44</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>8</td>
<td>748</td>
<td>62</td>
<td>7</td>
</tr>
<tr>
<td>ave</td>
<td>5.6</td>
<td>539.8</td>
<td>48.5</td>
<td>7.9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fold</th>
<th>Iterations</th>
<th>Evaluations</th>
<th>Operations</th>
<th>Height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>12</td>
<td>4244</td>
<td>217</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>9</td>
<td>3014</td>
<td>172</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>3483</td>
<td>224</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>8</td>
<td>2275</td>
<td>190</td>
<td>13</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>2379</td>
<td>159</td>
<td>12</td>
</tr>
<tr>
<td>6</td>
<td>12</td>
<td>4181</td>
<td>166</td>
<td>15</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>3640</td>
<td>230</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>3312</td>
<td>185</td>
<td>18</td>
</tr>
<tr>
<td>9</td>
<td>11</td>
<td>3733</td>
<td>215</td>
<td>12</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>3174</td>
<td>221</td>
<td>16</td>
</tr>
<tr>
<td>ave</td>
<td>9.7</td>
<td>3343.5</td>
<td>197.9</td>
<td>13.8</td>
</tr>
</tbody>
</table>

**Figure:** Soc (left) and Kids (right) performance statistics
Filter based Rewiring Strategy

Motivation
- For large scale datasets wrapper based approaches are intractable due to multiple hierarchy evaluations

Objective
- Modify predefined hierarchy using filter based rewiring strategy that does not requires multiple hierarchy evaluations
- Without significant loss in performance
Proposed Rewiring Strategy

- Elementary operation: node creation, parent-child rewiring, node deletion
Proposed Rewiring Strategy Algorithm

- Filter based approach for hierarchy modification

**Input:** Predefined hierarchy \((H_0)\), Train data \((D_t)\)

1. Compute pairwise similarity between classes defined in \(H_0\) on \(D_t\)
2. Group together most similar classes
3. Identify inconsistencies within the hierarchy
4. Apply elementary operations: node creation or parent-child rewiring to correct inconsistencies and obtain new hierarchy \(H_1\)
5. Perform post-processing step (node deletion) on \(H_1\) to obtain new hierarchy \(H_2\)
6. Train and evaluate hierarchical classification models on \(H_2\)
<table>
<thead>
<tr>
<th>Datasets</th>
<th>Flattening Best TD Model (Global-INR)</th>
<th>Rewiring Tang et al.</th>
<th>Proposed Filter Model</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu F_1(\uparrow)$</td>
<td>$\mu F_1(\uparrow)$</td>
<td>$\mu F_1(\uparrow)$</td>
</tr>
<tr>
<td>CLEF</td>
<td>77.14 (0.01)</td>
<td>78.12 (0.16)</td>
<td>78.00 (0.22)</td>
</tr>
<tr>
<td></td>
<td>46.54 (0.06)</td>
<td>48.83 (0.08)</td>
<td>47.10 (0.03)</td>
</tr>
<tr>
<td></td>
<td>79.06 (0.01)</td>
<td>81.43 (0.03)</td>
<td>80.14 (0.02)</td>
</tr>
<tr>
<td>DIATOMS</td>
<td>61.31 (0.53)</td>
<td>62.34 (0.28)</td>
<td>62.05 (0.10)</td>
</tr>
<tr>
<td></td>
<td>51.85 (0.23)</td>
<td>53.81 (0.11)</td>
<td>52.14 (0.14)</td>
</tr>
<tr>
<td></td>
<td>62.80 (0.04)</td>
<td>64.28 (0.22)</td>
<td>63.24 (0.13)</td>
</tr>
<tr>
<td>IPC</td>
<td>52.30 (0.12)</td>
<td>53.94 (0.24)</td>
<td>54.28 (0.18)</td>
</tr>
<tr>
<td></td>
<td>45.65 (0.11)</td>
<td>46.10 (0.21)</td>
<td>46.04 (0.22)</td>
</tr>
<tr>
<td></td>
<td>64.73 (0.12)</td>
<td>67.23 (0.24)</td>
<td>68.34 (0.18)</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td>46.61 (0.28)</td>
<td>NS</td>
<td>48.25 (0.13)</td>
</tr>
<tr>
<td></td>
<td>31.26 (0.64)</td>
<td>NS</td>
<td>33.92 (0.22)</td>
</tr>
<tr>
<td></td>
<td>63.37 (0.44)</td>
<td>NS</td>
<td>66.18 (0.15)</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td>42.37 (0.27)</td>
<td>NS</td>
<td>43.10 (0.28)</td>
</tr>
<tr>
<td></td>
<td>30.11 (0.64)</td>
<td>NS</td>
<td>31.21 (0.34)</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>50.64 (0.22)</td>
<td>NS</td>
<td>51.82 (0.02)</td>
</tr>
<tr>
<td></td>
<td>30.58 (0.28)</td>
<td>NS</td>
<td>31.24 (0.12)</td>
</tr>
<tr>
<td></td>
<td>73.19 (0.02)</td>
<td>NS</td>
<td>74.21 (0.03)</td>
</tr>
</tbody>
</table>
## Runtime Comparison

<table>
<thead>
<tr>
<th>Datasets</th>
<th>Flattening Best TD Model (Global-INR)</th>
<th>Rewiring Tang et al.</th>
<th>Proposed Filter Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLEF</td>
<td>3.5</td>
<td>59</td>
<td>7.5</td>
</tr>
<tr>
<td>DIATOMS</td>
<td>10</td>
<td>268</td>
<td>24</td>
</tr>
<tr>
<td>IPC</td>
<td>830</td>
<td>26432</td>
<td>1284</td>
</tr>
<tr>
<td>DMOZ-SMALL</td>
<td>65</td>
<td>NS</td>
<td>168</td>
</tr>
<tr>
<td>DMOZ-2010</td>
<td>25600</td>
<td>NS</td>
<td>42000</td>
</tr>
<tr>
<td>DMOZ-2012</td>
<td>63000</td>
<td>NS</td>
<td>94800</td>
</tr>
</tbody>
</table>

**Table:** Total training runtimes (in mins)

---

Huzefa Rangwala and Azad Naik  
George Mason University  
6th May, 2016  
79 / 108
## Elementary Operation Comparisons

<table>
<thead>
<tr>
<th># elementary operation executed</th>
<th>CLEF</th>
<th>DIATOMS</th>
<th>IPC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tang et al. (promote, demote, merge)</td>
<td>52</td>
<td>156</td>
<td>412</td>
</tr>
<tr>
<td>Proposed Rewiring Filter Model (node creation, PCRewire, node deletion)</td>
<td>25</td>
<td>34</td>
<td>42</td>
</tr>
</tbody>
</table>

Huzefa Rangwala and Azad Naik
George Mason University
6th May, 2016
Motivation

- Hierarchies are so common that sometimes multiple hierarchies classify similar data
- Heterogeneous label view provides additional knowledge which should be exploited by learners

Examples

- **protein structure classification** - several hierarchical schemes for organizing proteins based on curation process or 3D structure
- **web-page classification** - several hierarchy exist for categorizing such as DMOZ and wikipedia datasets

Objective

- Utilize multiple hierarchical label views in multi-task learning context to improve classification performance
(i) **Single Task Learning (STL)** - each task model parameters learned independently
(ii) **Single Hierarchy Multi-Task Learning (SHMTL)** - relationship between tasks within a hierarchy are combined individually.
(iii) **Multiple Hierarchy Multi-Task Learning (MHMTL)** - relationship between tasks from different hierarchies are extracted using common examples
MTL Formulations

- General MTL formulation:

\[ \sum_{t=1}^{T} \sum_{i=1}^{n_t} \mathcal{L}(f(x_i, W_t), y_i) + \lambda \Omega(\{W_t\}_{t=1}^{T}) \]

- Different MTL formulation based on regularization:
  - **Sparse** - All tasks share a single set of useful features
    \[ \Omega(W) = \|W\|_{2,1} \]
  - **Graph Regularization** - Related tasks have similar parameters
    \[ \Omega(W) = \sum_{(a,b) \in \mathcal{E}} \|W_a - W_b\|_2^2 \]
  - **Trace** - Task parameters are drawn from a low dimensional sub-space
    \[ \Omega(W) = \|W\|_* = TraceNorm(W) \]
Performance: AUC Comparison

SCOP

CATH
Extreme Classification

Motivation

- Many real world problems with multi-class and multi-label involving an extremely large number of labels or output space.
- Learning classifiers corresponding to each label is almost an impossible task.
- Inter-label dependency not available.
- Examples:
  - Predict hashtags from tweets.
  - LSHTC (Kaggle competition): Predict Wikipedia tags from documents.

Objective

- Given huge set of labels, identify the labels that can be assigned to unlabeled instances (examples), efficiently and accurately.
Extreme Classification Challenges

- **Statistical Challenges**
  - increase in number of classes is prone to decrease in accuracy due to complexity in discriminating between different classes

- **Computational Challenges**
  - training classifiers for large number of classes is computationally infeasible
  - predicting label for unlabeled test instances is also compute intensive task
Eigenpartition trees, Mineiro and Karampatziakis, NIPS workshop’15

- Compute small set of plausible labels using eigenpartition decomposition at each node in the tree

- At each node try to send each classs examples exclusively left or right

- While sending roughly the same number of examples left or right in aggregate

  - can be achieved through tree decomposition

  [Choromanska and Langford, NIPS’15]

- Optimization function

\[
\text{maximize } w^T (X^T X) w \\
\text{s.t. } w^T w \leq 1 \\
1^T X^T w = 0
\]

- Invoke expensive classifiers on set of plausible labels only
# Experiments

## Dataset description

- Large-scale text dataset used for model evaluation

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Average Labels Per Example</th>
<th>Examples</th>
<th>Features</th>
<th>Classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Twitter</td>
<td>1.27</td>
<td>25M</td>
<td>1M</td>
<td>264K</td>
</tr>
<tr>
<td>ALOI</td>
<td>1</td>
<td>97K</td>
<td>128</td>
<td>1K</td>
</tr>
<tr>
<td>ODP</td>
<td>1</td>
<td>1.5M</td>
<td>0.5M</td>
<td>100K</td>
</tr>
<tr>
<td>LSHTC</td>
<td>3.26</td>
<td>2.4M</td>
<td>1.6M</td>
<td>325K</td>
</tr>
</tbody>
</table>
Results - LSHTC dataset

- FastXML [Prabhu and Varma, SIGKDD’14] - node partitioning formulation which optimized an nDCG based ranking loss over all the labels.

- X1 [Bhatia et al., NIPS’15] - effective number of labels is reduced by projecting the high dimensional label vectors onto a low dimensional linear subspace.

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision at 1</th>
<th>Precision at 3</th>
<th>Precision at 5</th>
<th>Inference Speed (ex/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastXML</td>
<td>49.35%</td>
<td>32.69%</td>
<td>24.03%</td>
<td>2000*</td>
</tr>
<tr>
<td>X1</td>
<td>55.57%</td>
<td>33.84%</td>
<td>24.07%</td>
<td>125*</td>
</tr>
<tr>
<td>Tree + ILR</td>
<td>53.0%</td>
<td>33.9%</td>
<td>24.8%</td>
<td>1058</td>
</tr>
<tr>
<td>$k = 6000$</td>
<td>53.7%</td>
<td>34.5%</td>
<td>25.3%</td>
<td>688</td>
</tr>
<tr>
<td>$k = 12000$</td>
<td>54.3%</td>
<td>34.9%</td>
<td>25.6%</td>
<td>370</td>
</tr>
</tbody>
</table>
Motivation

- Large scale taxonomies are more prevalent due to the more specific topic related class information that is beneficial in several domains.

Examples

- **web search browsing** - finding documents relevant to query
- **modeling user’s for personalized web search** - ”java” means different to tourist and programmer
- **advertisement matching** - finding related ads corresponding to web-page

Traditional algorithms cannot be directly scaled to large scale problems due to several drawbacks:

- **large scale hierarchies**
- **longer training time** and
- incorporating **structural information** into learning framework
Observations

- Related categories corresponding to the query document are smaller than the number of unrelated categories.
- Performance on smaller set of categories is easier and much better compared to the large set of categories.

Objective

- Given large and deep hierarchies identify the relevant subset of categories for effectively finding the label of unlabeled test instances.
Two Stages for Classification

- **First stage - Search stage**
  - Identify related candidate categories corresponding to the test example

- **Second stage - Classification stage**
  - Select the best candidate categories using classification algorithm as the label for test document
First Stage

- Large hierarchy is pruned into smaller subset of hierarchy with candidate categories and its ancestors only
Second Stage

- Classifiers are trained on candidate categories
- Best category for test example is selected using trained classifiers
Dataset

- Open Directory Projects (ODP) dataset used for evaluation
- Dataset statistics
  - Training dataset - 1,174,586 web pages, 130,000 categories organized into 15 levels
  - Test dataset - 130,000 web pages
Results - Micro-F1 performance comparison

- **Search based Strategy** - best neighbor is chosen as the label for test example
- **Hierarchical SVM** [Liu et al., SIGKDD’05] - a pachinko machine style SVM
- **Deep Classification** - top ten neighbors are used as the candidate categories
As the number of candidate categories chosen by the search stage increases; chances for finding the correct label for test example in the classification stage increases.

Evaluation time increases with increasing number of category selection.

![Graph showing the relationship between number of candidate categories and Micro-F1 scores for different levels.](image)
Conclusion

- Large scale hierarchical classification is an important research problem in machine learning community due to its wide applicability across several domains
- Discussed various challenges associated with the hierarchical classification
- Discussed various state-of-the-art existing approaches; Demo of the software package developed by the author
- Emerging topics:
  - Large-scale classification with deep hierarchies
  - Orphan node prediction


Qi, Xiaoguang, and Brian D. Davison. ”Hierarchy evolution for improved classification.” CIKM, 2011.


Choromanska, Anna E., and John Langford. ”Logarithmic time online multiclass prediction.” NIPS, 2015.


Acknowledgement

Presenter:

Huzefa Rangwala
Azad Naik

Slides available for download at:

https://cs.gmu.edu/~mlbio/sdm2016tutorial.html
Thank You!