Filter based Taxonomy Modification for Improving Hierarchical Classification

Azad Naik  
Department of Computer Science  
George Mason University  
Fairfax, Virginia (USA)  
anai3@gmu.edu

Huzefa Rangwala  
Department of Computer Science  
George Mason University  
Fairfax, Virginia (USA)  
rangwala@cs.gmu.edu

ABSTRACT
Large scale classification of data organized as a hierarchy of classes has received significant attention in the literature. Top-Down (TD) Hierarchical Classification (HC), which exploits the hierarchical structure during the learning process, is an effective method for dealing with problems at scale due to its computational benefits. However, its accuracy suffers due to error propagation i.e., prediction errors made at higher levels in the hierarchy cannot be corrected at lower levels. One of the main reasons behind errors at the higher levels is the presence of inconsistent nodes and links that are introduced due to the arbitrary process of creating these hierarchies by domain experts. In this paper, we propose two efficient data driven filter based approaches for hierarchical structure modification: (i) Flattening (local and global) approach that identifies and removes inconsistent nodes present within the hierarchy and (ii) Rewiring approach modifies parent-child relationships to improve the classification performance of learned models. Our extensive empirical evaluation of the proposed approaches on several image and text datasets shows improved performance over competing approaches. Source code available for reproducibility at: www.cs.gmu.edu/~mlbio/TaxMod

Keywords
Top-down Hierarchical Classification, Inconsistency, Error Propagation, Flattening, Rewiring

1. INTRODUCTION
Taxonomy (hierarchy) is used widely to store large volumes of data. It provides a structured view of the domain where classes (categories) are organized from the most generic to the most specific in a top-down order. It has been used as a data representation in various application domains such as bioinformatics1, patent2, databases, computer vision3 and web-taxonomy4.

Hierarchies provide useful structural relationships (such as parent-child and siblings) among different classes that can be exploited for learning generalized classification models. In the past, researchers have demonstrated the usefulness of hierarchies for classification and have obtained promising results 4, 11, 15, 8, 19. However, in many situations the hierarchy used for learning classifiers is not consistent for classification and HC approaches are outperformed by flat classifiers that ignore the hierarchy 10, 22, 25.

In majority of the cases, the hierarchy available for training classifiers is manually designed by the experts based on domain knowledge. This manual process of hierarchy creation suffers from various issues that makes it inconsistent for achieving good classification performance. Specifically, (i) Hierarchies are designed for easy search and navigation. (ii) Hierarchies are generated by grouping semantically similar categories under a common parent category. Moreover, many different semantically sound hierarchies may exist for same set of classes. For e.g., 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 hierarchies may lead to significantly different classification results. (iii) A-priori 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. (iv) Large number of categories also pose a challenge for the manual design of a consistent hierarchy.

In this paper, our main focus is on generating an improved representation from the expert-defined hierarchy. Specifically, we propose scalable data driven taxonomy modification approaches which are suitable for HC, leading to better generalization capabilities of learned models. To summarize, our major contributions include:

1. We propose local and global flattening approaches that are able to better identify the set of inconsistent nodes that exists within the hierarchy, thereby improving the classification performance in comparison to the previous flattening approaches 2, 21.
2. We propose an efficient filter based taxonomy modification approach which unlike previous wrapper based approaches 20, 18, 12 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.

With these taxonomy modifications, TD HC methods are significantly better than the flat methods in classification performance and prediction efficiency. Taxonomy obtained using our proposed methods can also be used with various existing state-of-the-art HC approaches 10, 13, 23 for improving the classification performance.
2. LITERATURE REVIEW

HC approaches rely on the hierarchical structure for learning models [4] [8] [10] [11] [15] [19]. In most cases, domain experts define the hierarchical structure where semantically similar categories are grouped together into a parent category. Hierarchies generated using this method are independent of the data. Various approaches for hierarchy modification have been proposed in the literature. These approaches can be broadly categorized into two classes: (i) Flattening strategy where some identified inconsistent nodes (based on error rate, classification margins) are flattened and (ii) Rewiring strategy where parent-child relationships within the hierarchy is modified to improve the classification performance of learned models. Summary of the various existing methods for taxonomy modification and their inherent characteristics are shown in Table 1.

### 2.1 Flattening Strategy

Flattening refers to the removal of certain nodes or entire levels of hierarchies; one of the early approaches used for hierarchy modification [3] [14] [21]. The main intuition behind flattening is to selectively remove the problematic nodes that cause deterioration in classification performance. Based on levels that are flattened (removed), various methods of flattening exist. One of the simplest method is to remove (flatten) all the internal nodes. This is referred as flat hierarchy and is shown in Figure 1 (b). Flat hierarchy ignores the hierarchical structure completely which may contain some valuable information for improving the performance results of learned classification models (especially for rare categories). Top Level Flattening (TLF) as shown in Figure 1(c) modifies the hierarchy by removing the top level in the original hierarchy. Bottom Level Flattening (BLF) and Multiple Level Flattening (MLF), shown in Figures 1(d) and (e) are similar methods of hierarchy modification where bottom and multiple levels are removed, respectively. Recently, learning based approaches [3] for hierarchy modification are proposed, where each node in the hierarchy is recursively flattened and evaluated for performance improvement over the original (predefined) hierarchy. If there is a significant improvement then the node is marked as inconsistent and removed, otherwise the node is retained in the final hierarchy. This approach although useful for small datasets, is not scalable due to the expensive evaluation process after each node removal and is a wrapper approach. In other work [2], maximum margin-based approach for taxonomy adaptation (MTA) was proposed where some nodes are intelligently flattened based on a defined threshold and margin scores obtained for each node. Hierachy modification using this approach (see Figure 1(f)), is scalable and beneficial for classification and has been theoretically justified [9]. We followed a similar filter approach for hierarchy modification. However, we define a systematic approach for determining threshold criterion to identify the set of inconsistent nodes, based on deviation from mean.

### 2.2 Rewiring Strategy

Although, the flattening strategy shows performance improvement, it suffers from limitations. Figure 2(a) (a) shows the original expert-defined hierarchy where leaf nodes (categories) with higher similarities are marked by the same color and shape. In Figure 2(a), node C1 is incorrectly linked to node C instead of A, which makes learning a discriminative model difficult at higher level i.e., between nodes A and C. Flattening strategy (shown in Figure 2(b)) modifies this hierarchy by flattening (removing) the inconsistent node.
C that contains leaf categories with varying similarities between siblings. In spite of the fact that flattened hierarchy is comparatively better than the original hierarchy in terms of classification performance, it can be improved further to better leverage the hierarchy information by rewiring node C1 as a child of node A as done in Figure 2 (c) (a rewiring strategy). To summarize, flattening strategy cannot deal with inconsistencies that occurs in different branches (sub-trees) of the hierarchy. Rewiring strategy can be broadly categorized into two class of approaches: (i) Automated hierarchy generation and (ii) Predefined (original) hierarchy modification.

2.2.1 Automated Hierarchy Generation

In this approach, the hierarchy is generated from scratch ignoring the expert defined hierarchy. Most of the work in this category exploits hierarchical clustering algorithms for generating the hierarchy [13, 14, 15, 16, 17, 18, 19]. In [15], a two step method for constructing hierarchy is proposed. This method uses a linear discriminant for projecting the data into a lower dimensional space followed by the hierarchical agglomerative clustering to generate a binary tree. For classification, the binary tree is converted to a two-level tree according to cluster coherence. In other work proposed in [17], a divisive clustering approach is followed where the categories assigned to each cluster are recursively divided into sub-clusters using spherical K-means.

Constructing hierarchy using automated approaches is not popular due to predefined parameter requirements, making these approaches very ad-hoc. Moreover, these approaches generate binary trees which are not scalable for large datasets. In this paper we do not focus on automated hierarchy generation approaches.

2.2.2 Predefined (original) Hierarchy Modification

In the existing rewiring approaches [18, 19, 16], the predefined expert hierarchy is modified in steps to make it better suited (aligned) for the classification task. In [19], at each step the 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 that satisfies certain criteria is reached.

In [20], modifications are performed using three elementary operations – promote, demote and merge. Starting from the predefined (original) hierarchy, the current best hierarchy is modified by applying one of the three elementary operations at each iteration, until an optimal hierarchy with best classification performance is obtained. This approach produces significant performance improvement in comparison to a clustering based approach [12]. However, 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 to apply at each step. Similarly, hierarchy modification using genetic algorithms [18] requires many candidate hierarchies to be evaluated at each subsequent step. Other approaches focus on improving the runtime performance (apart from classification performance) [16], where the hierarchy is only evaluated in the modified sub-branches. Some improvements in results over the expert defined hierarchy have been reported. However, this approach is heuristic by nature and requires parameter tuning.

We propose a filter-based data-driven rewiring approach, where the taxonomy is modified based on certain relevance criterion (such as pairwise siblings similarities) between the different classes within the hierarchy. In contrast, wrapper based methods [18, 20, 16] iteratively modify the hierarchy by making one or few changes, which are then evaluated by training a classification model on a validation set to identify if the modified hierarchy has improved performance. As such, filter methods are scalable for large datasets as they are single step and do not require experimental evaluation.

Table 2: Notation description.

| Symbol | Description |
|--------|-------------|
| H     | original given hierarchy |
| L     | set of leaf categories (classes) |
| X_i   | input vector for i-th training example |
| y_i   | true label for i-th training example |
| y^*_i | binary label used for i-th training example to learn weight vector for n-th node in H, y^*_i = 1 iff y_i = n, -1 otherwise |
| N     | total number of training examples |
| Θ_n   | weight vector for n-th node |
| 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 |
| H_M   | set of nodes (except root) in H |
| N_k   | set of nodes at k-th level in H |
| I_k   | set of inconsistent nodes using Level-INR method |
| I_G   | set of inconsistent nodes using Global-INR method |
| μ(S)  | mean of samples in set S |
| σ(S)  | standard deviation of samples in set S |
| S_k   | set of f^+ values for node at k-th level in H |
| τ_k   | threshold for nodes flattening at k-th level in H |
| τ    | global threshold for nodes flattening or threshold for grouping similar classes in rewiring method |
| π(n)  | parent of the n-th node |
| ζ(n)  | siblings of the n-th node |
3. PROPOSED METHODS

Table 2 summarizes the common notations used in this paper. We use bold letters to indicate vector variables.

3.1 Hierarchical Classification

Given, a hierarchy H = (V, E), where V denotes the set of nodes and E denotes the set of edges. We train one-vs-rest classifiers for each of the nodes n ∈ N — where N is the set of all nodes excluding the root node — to discriminate its examples from other node examples in the hierarchy. In this paper, 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 L2-norm term (denoted by ||·||2) to control model complexity and prevent overfitting. The objective function fn for training a model corresponding to node n is provided in eq. (1).

$$\min_{\Theta_n} \left[ C \sum_{i=1}^{N} \log \left( 1 + \exp \left( -y_i^n \Theta_n^T x_i \right) \right) + \frac{1}{2} \|\Theta_n\|^2 \right]$$  \hspace{1cm} (1)

For each node n within the hierarchy, we solve eq. (1) to obtain the optimal weight vector denoted by \(\Theta_n\). The complete set of parameters for all the nodes \(\{\Theta_n\}_{n \in N}\) constitutes the learned model for the hierarchical TD classifier. For LR models the conditional probability for \(\hat{y}_i^n \in \pm 1\) given its feature vector \(x_i\) and the weight vector \(\Theta_n\) is given by eq. (2) and the decision function is given by eq. (3).

$$P(\hat{y}_i^n \mid x_i, \Theta_n) = \frac{1}{1 + \exp \left( -y_i^n \Theta_n^T x_i \right)}$$  \hspace{1cm} (2)

$$\hat{y}_i^n = \begin{cases} +1 & f_n(x_i) = \Theta_n^T x_i \geq 0 \\ -1 & \text{otherwise} \end{cases}$$  \hspace{1cm} (3)

For a test example with feature vector \(x_i\), the TD classifier predicts the class label \(\hat{y}_i \in L\) as shown in eq. (4), where \(L = \{1, \ldots, m\}\) denotes the set of children of node p. Essentially, the algorithm starts at the root nodes and recursively selects the best child nodes till it reaches a terminal node belonging to the set of leaf nodes L.

$$\hat{y}_i = \begin{cases} \text{initialize } & p := \text{root} \\ \text{while } p \notin L & p := \text{argmax}_{\gamma \in L \cap \mathcal{C}(p)} f_p(x_i) \\ \text{return } & p \end{cases}$$  \hspace{1cm} (4)

3.2 Flattening Approach

It has been shown \cite{11} that for any classifier that correctly classifies m random input-output pairs using a set of D decision nodes, the generalization error bound with probability greater than 1 - Δ is less than the expression shown in eq. (5).

$$\frac{\Delta}{m} \left[ \sum_{n \in D} \frac{1}{\gamma_n} \log(4em) \log(4m) + |D| \log(2m) - \log \left( \frac{2}{\Delta} \right) \right].$$  \hspace{1cm} (5)

where \(\gamma_n\) denotes the margin at node n ∈ 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) increasing the margin \(\gamma_n\) for nodes in the hierarchy or (ii) decreasing the number of decision nodes |D| involved in prediction. For learning the optimal model we need to balance the trade-off between the margin \(\gamma_n\) and the number of decision nodes |D|. Two of the extreme cases for learning hierarchical classifiers are flat and TD methods. For flat classifiers, we have to make a single decision (i.e., |D| = 1) but margin width \(\gamma_n\) is presumably small due to the large number of leaf categories, which makes it difficult to obtain larger margin. For TD hierarchical classifiers, we have to make a series of decisions from root to leaf nodes (i.e., |D| ≥ 1) but margin \(\gamma_n\) is larger due to the fewer number of categories that need to be distinguished at each of the decision nodes. In this paper, we propose a method that removes some of the inconsistent nodes in the hierarchy \(\mathcal{H}\), thereby, increasing the value of margin \(\gamma_n\) for nodes in the hierarchy, while minimizing the number of decision nodes.

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

**Level-wise (local) Inconsistent Node Removal** - In this approach, referred as Level-INR, we select a different threshold \(\gamma_k\) for each level k of the hierarchy. Algorithm 2 presents the level-wise approach that selects inconsistent nodes at each level in a top-down manner. The threshold \(\gamma_k\) for level k is computed as the sum of mean and standard deviation of the set of values \(\{f_n^*\}_{n \in N_k}\), where \(N_k\) represents the set of nodes in level k. All nodes \(n \in N_k\) that satisfy \(f_n^* > \gamma_k\) are marked as inconsistent and added to the set of inconsistent nodes denoted by \(I_L\). This procedure is repeated for all levels of the hierarchy. Finally, we flatten the nodes in set \(I_L\) — remove n ∈ \(I_L\) and corresponding edges, and add edges from children of n to n’s parent node. The modified hierarchy \(\mathcal{H}_M\) obtained by flattening is used to re-train a top-down classifier.

**Global Inconsistent Node Removal** - Different from the Level-INR approach, which selects different thresholds for each level, the global method computes a single threshold value for all levels. The threshold \(\tau\) is computed as the sum of mean and standard deviation of the set of values \(\{f_n^*\}_{n \in N}\), where \(N\) represents the set of all nodes except the root node. \(\tau\) is used to identify the set of inconsistent nodes \(I_G\) in the hierarchy (i.e., all nodes n with \(f_n^* > \tau\) ), the modified hierarchy \(\mathcal{H}_M\) obtained by flattening the nodes present in \(I_G\) is used to re-train a top-down classifier. In this paper we refer to this approach as Global-INR.

3.3 Rewiring Approach

To overcome the drawback associated with flattening approach (discussed in Section 2.2), we propose an efficient filter based data driven rewiring approach for taxonomy modification.

**Filter based Taxonomy Modification - Algorithm 2**
Algorithm 1 Level-wise Inconsistent Node Removal

Data: Original Hierarchy \( H \), input-output \( (x_i, y_i) \)
Result: Modified Hierarchy \( H_M \)
Train \( \ell_2 \)-regularized LR model in a top-down order

/* for each inconsistent node, initially empty */
\[ I_L := \Phi; \]
for \( k := 1 \ldots \) endlevel do
  /* Set of all nodes */
  \[ S_k := S_k \cup \{ f_n \}; \]
  /* Identify inconsistent node in level */
  \[ \tau_n := \mu(S_k) + \sigma(S_k); \]
  /* inconsistent pair check */
  if \( \{ f_n, \tau_n \in S_k \} \) then
    \[ I_L := I_L \cup \{ n \}; \]
  end
end
/* New hierarchy with inconsistent node(s) removed */
\[ H_M = H - (I_L); \]
return \( H_M \)

Algorithm 2 Filter based Taxonomy Modification

Data: Original Hierarchy \( H \), input-output \( (x_i, y_i) \)
Result: Modified Hierarchy \( H_M \)
/* Initialization */
\[ H_M := H; \]
/* First step: Grouping Similar Classes */
/* similarity computation */
Compute the similarity between all possible class pairs using cosine similarity
/* similar class grouping */
Group the most similar class pair based on empirically defined threshold parameter \( \tau \). Let \( c \) denotes the number of similar class pair represented by the set \( S = \{ s_1, s_2, \ldots, s_c \} \), where \( i \)-th pair \( s_i \) is represented using \( (s_i^{(1)}, s_i^{(2)}) \)
/* Second step: Inconsistency Identification and Correction */
for \( i = 1 \) to \( |c| \) do
  /* perform node creation */
  \[ \text{rewire}[1] = \text{yes}; \]
  \[ \text{rewire}[2] = \text{yes}; \]
  /\* inconsistent pair check */
  if \( \pi(s_i^{(1)}) \neq \pi(s_i^{(2)}) \) then
    /* check similarity to all siblings */
    foreach \( j \in \pi(s_i^{(1)}) \) do
      if \( (j, s_i^{(2)}) \notin S \) then
        \[ \text{rewire}[2] = \text{no}; \]
        break;
      end
    endforeach
    /* perform node creation */
    \[ \text{rewire}[1] = \text{no}; \]
  end
  /* perform node deletion */
  if \( \text{rewire}[1] == \text{no} \) and \( \text{rewire}[2] == \text{no} \) then
    /* create new node */
    \[ [H_M] = \text{nodeCreation}(N_{new} ightarrow \text{lc}(s_i), s_i \rightarrow N_{new}, H_M); \]
  else
    if \( \text{rewire}[1] == \text{yes} \) then
      \[ [H_M] = \text{parent-childRewiring}(s_i^{(1)} ightarrow \pi(s_i^{(2)}), H_M); \]
    else
      \[ [H_M] = \text{parent-childRewiring}(s_i^{(2)} ightarrow \pi(s_i^{(1)}), H_M); \]
    end
  end
end
/* perform node deletion */
\[ [H_M] = \text{nodeDeletion}(H_M); \]
return \( H_M \)

Figure 3: Modified hierarchical structures (b)-(d) obtained after applying elementary operation to original hierarchy (H). Leaf nodes are marked with 'rectangle' and structural changes are shown by blue color.

describes our proposed approach for taxonomy modification. It consists of two steps:

(i) **Grouping Similar Classes** - To ensure similar classes are grouped together to the same parent node in the modified taxonomy; this step identifies and groups the most similar classes that exists within the predefined experts hierarchy.

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. Cosine similarity is used as the similarity metric in our experiments and we use a separate validation dataset for computing similarities. Once the similarity scores are computed, we determine the set \( S \) of most similar pairs of classes using an empirically defined cut-off threshold for a dataset.

(ii) **Inconsistency Identification and Correction** - Similar classes determined in the previous stage are used as an input for this step. To obtain the consistent hierarchy, we need to 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 inconsistency i.e., if the pairs of classes are in different branches (sub-trees). In order to resolve the identified inconsistencies, we take the corrective measures using three elementary operation: node creation, parent-child rewiring and node deletion. Figure 3 illustrates the various hierarchical structures that are obtained after the execution of elementary operation on predefined (original) hierarchy, Figure 3(a).

**Node Creation (NC)** - This helps to group together the identified similar class pairs in different branches (sub-trees) of the hierarchy using a newly created node, with parent as the lowest common ancestors of similar classes. Figure 3(b) illustrate this operation where 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).
datasets, we have applied the tf-idf transformation with
for loop removed from the modified hierarchy.
algorithm to refine the hierarchy, where irrelevant nodes are
inconsistencies have been addressed, algorithm calls the
in the hierarchy that does not have any associated leaf
node as its descendant. As shown in Figure 3(d), internal
node in the hierarchy that does not have any associated leaf
nodes that can be classified by node B.
For resolving inconsistency, our algorithm determines (outer
for loop) the best corrective measures (node creation or
parent-child rewiring) that needs to be taken. Once all the
inconsistencies have been addressed, algorithm calls the
node deletion procedure as a final modification step where unnecessary
internal nodes with 0 or 1 child are deleted.

4. EXPERIMENTS AND RESULTS

4.1 Datasets

We have used an extensive set of text and image datasets for
evaluating the performance of our proposed approaches.
Various statistics of the datasets used are listed in Table 3. All these datasets are single labeled and the examples are assigned to the leaf nodes in the hierarchy. For all text datasets, we have applied the tf-idf transformation with f2-norm normalization on the word-frequency feature vector.
Description of the dataset used for experiments are:

**Image Datasets**
- **CLEF**: Medical images represented by 80 features that are extracted using local distribution of edges.
- **DIATOMS**: Diatom images where features are created using various feature extraction techniques.
- **DMOZ-SMALL, DMOZ-2010 and 2012**: Multiple web documents organized in various classes using the hierarchical structure. Dataset has been released as the part of the LSHTC challenge in the year 2010 and 2012. For evaluating the DMOZ-2010 and DMOZ-2012 datasets we have used the provided test split. The results reported for these two benchmarks are blind prediction obtained from web-portal interface.

**Text Datasets**
- **IPC**: Collection of patent documents organized in International Patent Classification (IPC) hierarchy.
- **DMOZ-2010, DMOZ-2012**: Multiple web documents organized in various classes using the hierarchical structure. Dataset has been released as the part of the LSHTC challenge in the year 2010 and 2012.

### Table 3: Dataset statistics.

| Dataset     | # Total Node | # Leaf Node | Height | # Training | # Testing | # Features |
|-------------|--------------|-------------|--------|------------|-----------|------------|
| CLEF        | 399          | 311         | 4      | 19,400     | 983       | 371        |
| DIATOMS     | 553          | 451         | 4      | 46,324     | 28,926    | 1,123,497  |
| IPC         | 2,388        | 1,139       | 6      | 6,323      | 1,858     | 51,033     |
| DMOZ-SMALL  | 17,222       | 12,394      | 6      | 128,710    | 34,880    | 381,580    |
| DMOZ-2010   | 13,963       | 11,947      | 6      | 383,408    | 103,435   | 348,548    |

5http://www.wipo.int/classifications/ipc/en/
6http://dmoz.org
7http://lshtc.iit.demokritos.gr/

**Parent-child Rewiring (PCRewire)** - As shown in Figure 3(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 as similar to all sibling leaf nodes within the given hierarchy branch. For example, in Figure 3(d) 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 classification perspective to assign 6 as node B child rather than C.

**Node Deletion (ND)** - This refers to deletion of internal node in the hierarchy that does not have any associated leaf node as its descendant. As shown in Figure 3(d), internal node B is deleted because it does not help in classification as 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, where irrelevant nodes are removed from the modified hierarchy.

For resolving inconsistency, our algorithm determines (outer for loop) the best corrective measures (node creation or parent-child rewiring) that needs to be taken. Once all the inconsistencies have been addressed, algorithm calls the node deletion procedure as a final modification step where unnecessary internal nodes with 0 or 1 child are deleted.

**4.2 Evaluation Metrics**

**Flat Measures** - We have used the standard metrics micro-$F_1 (\mu F_1)$ and macro-$F_1 (MF_1)$ for evaluating the performance of various methods. To compute $\mu F_1$, 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 score as:

$$P = \frac{\sum_{c \in L} TP_c}{\sum_{c \in L} (TP_c + FP_c)}$$  \hspace{1cm} (6)

$$R = \frac{\sum_{c \in L} TP_c}{\sum_{c \in L} (TP_c + FN_c)}$$ \hspace{1cm} (7)

$$\mu F_1 = \frac{2PR}{P + R}$$  \hspace{1cm} (8)

Unlike $\mu F_1$, $MF_1$ gives equal weight to all the categories so that the average score is not skewed in favor of the larger categories. It is defined as follows:

$$P_c = \frac{TP_c}{TP_c + FP_c}$$ \hspace{1cm} (9)

$$R_c = \frac{TP_c}{TP_c + FN_c}$$ \hspace{1cm} (10)

$$MF_1 = \frac{1}{|L|} \sum_{c \in L} \frac{2P_cR_c}{P_c + R_c}$$ \hspace{1cm} (11)

where, $|L|$ is the number of categories (classes).

**Hierarchical Measures** - This metrics incorporate hierarchy for evaluating the classifier performance. The hierarchy based measures include hierarchical $F_1 (hF_1)$ (harmonic mean of hierarchical precision ($hP$), hierarchical recall ($hR$)) are defined as follows:

$$hP = \frac{\sum_{i=1}^{N} |A(y_i) \cap A(\hat{y}_i)|}{\sum_{i=1}^{N} |A(y_i)|}$$ \hspace{1cm} (12)

$$hR = \frac{\sum_{i=1}^{N} |A(y_i) \cap A(\hat{y}_i)|}{\sum_{i=1}^{N} |A(\hat{y}_i)|}$$ \hspace{1cm} (13)

$$hF_1 = \frac{2 * hP * hR}{hP + hR}$$ \hspace{1cm} (14)

Note that for consistent evaluation of hierarchical measures we have used the original hierarchy for all methods unless noted.

**4.3 Methods for Comparison**

3http://lshtc.iit.demokritos.gr/node/81
9http://lshtc.iit.demokritos.gr/LSHTC3_oracleUpload
4.3.1 Flat Method

For the flat classifiers we train one-versus-rest regularized logistic regression (LR) classifiers for each of the leaf categories, ignoring the hierarchical structure. The prediction decision is based on the maximum prediction score achieved when compared across the one-versus-rest classifiers. This method is referred as LR.

4.3.2 Top-Down Hierarchical Methods

For all TD hierarchical baselines we train one-vs-rest classifiers for each of the nodes (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. (4)). We choose one-vs-rest over one-vs-sibling for learning models because our preliminary experiments showed better performance with one-vs-rest method. Depending upon the hierarchy that we use during the training process, we use the following TD HC methods for comparison purpose.

Top-Down Logistic Regression (TD-LR): We use original hierarchy provided by domain experts for training the classifiers.

Level Flattening Approach [21]: Modified hierarchy obtained by flattening level(s) is used for classifiers training. Depending on level(s) flattened we have TLF, BLF and MLF as discussed in Section 2.1. For MLF, we flatten the first and third level nodes as proposed in [21].

Margin based Taxonomy Adaptation (MTA) [2]: Hierarchy is modified using the computed margin value and threshold defined at each node in the hierarchy.

Optimal Hierarchy Search [20]: Optimal hierarchy is searched in the hierarchical space by modifying the predefined hierarchy using elementary operation – promote, demote and merge. For reducing the number of hierarchy evaluation, we have restricted the modification to the hierarchy branches where we encountered the maximum errors. This modified approach is referred as T-Easy. In the original paper largest evaluated dataset has 244 classes and 15795 instances.

4.4 Experimental Protocol

To make the experimental evaluation results comparable to previously published results, we have used the same train-test split as provided by the benchmarks. 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 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 \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}. The best parameter is selected using a validation set. The best parameters are used to retrain the models on the entire training set and the performance is measured on a held out test set. All experiments were conducted using a modified version of liblinear software and were run on a compute cluster with Dell C8220 compute nodes, each with dual Intel Xeon E5-2670 (2.60GHz) 8 core CPUs.

4.5 Evaluation Results

4.5.1 Case Study

To understand the qualitative difference of our proposed approaches, we present results on the newsgroup dataset containing 20 classes and 10,000 training examples. Figure 4 (b) and (c) shows the modified hierarchical structure obtained using our best proposed flattening approach (Global-INR) and rewiring approach, respectively. Performance evaluation on these hierarchy shows improved classification results in comparison to the original (predefined) hierarchy shown in Figure 4 (a). On comparing flattened and rewired hierarchy, performance of rewired hierarchy (\( \mu F_1 = 81.24 \)) is found to be significantly better than the flattened hierarchy (\( \mu F_1 = 79.42 \)) because of the flattening approach limitation that it cannot group together the classes from different hierarchical branches (for e.g. soc.religion.christian and religion.misc or electronics and graphics), limiting the performance improvement. On the contrary, rewiring ap-
access the performance improvement. We perform sign-test method. Significance test between models is performed to comparing Global-INR with the best TD hierarchical baseline. The results show that our proposed method, Global-INR, consistently outperforms other TD HC methods for all the datasets. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics.

We can see that our proposed method, Global-INR, consistently outperforms other TD HC methods for all the datasets. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics.

## 4.5.2 Flattening Approaches

**Performance based on Flat Metrics** - Table 4 shows the $\mu F_1$ and $\mu F_1$ performance comparison of our proposed flattening methods with different TD hierarchical baselines. We can see that our proposed method, Global-INR, consistently outperforms other TD HC methods for all the datasets. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics.

| CLEF | $\mu F_1$ | $\mu F_1$ |
|------|-----------|-----------|
| MTA  | 63.42 (0.28) | 61.98 (0.56) |
| New  | 63.37 (0.44) | 62.52 (0.38) |

Table 4: $\mu F_1$ and $\mu F_1$ performance comparison of various TD hierarchical baseline models against our proposed flattening INR models. Table shows mean and (standard deviation) in brackets across five runs. ‘X’ denotes MLF not possible. The significance-test results are denoted as $\dagger$ for a p-value less than 5% and $\ddagger$ for p-value less than 1%. Significance test are between best proposed, Global-INR model and best hierarchical baseline model. Test cannot be performed on DMOZ-2010 and DMOZ-2012 dataset because we don’t have access to true labels.

The results show that our proposed method, Global-INR, consistently outperforms other TD HC methods for all the datasets. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics.

On comparing our two proposed flattening approaches, Global-INR has better performance than Level-INR because the Level-INR method strictly enforces some of the nodes to be removed from each levels although their $f^*_n$ value may be much lower than the other nodes at different levels in the hierarchy and vice-versa. In contrast, Global-INR method takes all nodes into consideration while making a decision to identify the inconsistent nodes and hence it determines a better set of inconsistent nodes. Maximum margin based approach, MTA has poor performance due to the similar issues as with Level-INR approach. Level flattening methods such as TLF, BLF and MLF, have poor performance because these methods remove the entire level in the hierarchy and do not take into consideration whether any node in that level is important or not, resulting in poor performance. The baseline method, TD-LR has worst performance because of the presence of inconsistent nodes in the hierarchy which leads to the error propagation.

**Performance based on Hierarchical Metrics** - Table 5 shows the $h F_1$ performance comparison of best TD hierarchical baseline model and proposed flattening models over original and new modified hierarchy. $h F_1$ score for DMOZ-2010 dataset is not available from online evaluation and DMOZ-2012 dataset $h F_1$ score cannot be computed on new hierarchy.

| CLEF | Original | New |
|------|----------|-----|
| $\mu F_1$ | 0.61 (0.74) | 0.61 (0.74) |

Table 5: $h F_1$ performance comparison of best TD hierarchical baseline model and proposed flattening models over original and new modified hierarchy. $h F_1$ score for DMOZ-2010 dataset is not available from online evaluation and DMOZ-2012 dataset $h F_1$ score cannot be computed on new hierarchy.

The results show that our proposed method, Global-INR, consistently outperforms other TD HC methods for all the datasets. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics. In fact, for image dataset we see a relatively better performance improvement of $\sim 30\%$ across all metrics.
4.5.3 Rewiring Approach

Classification Performance - Table 6 shows the performance comparison of our proposed filter based rewiring approach with best flattening approach, Global-INR and competing rewiring approach, T-Easy. We can make following observation from the Table 6 (i) The rewiring approaches significantly outperforms the best flattening approach, Global-INR (ii) Our proposed filter based model achieves competitive performance results in comparison to T-Easy model. However, for most of the smaller datasets T-Easy model has better performance because it searches for the optimal hierarchy in the hierarchical space which can gradually reach the performance improvement (or even better) that is achieved by our model. Nevertheless, main drawback of T-Easy approach is that it needs a lot of computationally expensive hierarchy evaluations for reaching the optimal hierarchy, which makes this approach intractable for larger datasets (DMOZ).

Runtime Comparison - In Table 7 we compare the training times of different models. For training, we learn the models in parallel for different classes using multiple compute nodes which is 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 7 that TD-LR takes the least time as there is no extra hierarchy modification overhead; followed by the Global-INR model which requires models re-training after hierarchy flattening. Rewiring approaches are most expensive because of the compute intensive task of either similarity computation using our proposed approach or multiple hierarchy evaluations using the T-Easy approach. Comparatively, the T-Easy method takes the longest time due to large number of expensive hierarchy evaluation after each elementary operation until the optimal hierarchy is reached. Table 6 shows the number of elementary operations executed using the T-Easy and our proposed approach. We can see that T-Easy approach executes extremely large number of operations even for smaller datasets (for e.g., 412 operations for IPC datasets), making it expensive.

Empirical Study for Threshold (τ) Selection to

Figure 5: $M_F$ performance comparison of Flat (marked in dotted red) and best TD classifier, Global-INR (marked in solid blue) after varying threshold (τ) for CLEF and DMOZ-SMALL dataset.

Figure 6: Sorted cosine similarity scores for DMOZ-SMALL dataset.

Table 6: $\mu F_1$, $M F_1$ and $h F_1$ performance comparison of rewiring approaches against best flattening model. ‘NS’ denotes not scalable. The significance-test results are denoted as † for a p-value less than 5% and ‡ for p-value less than 1%. Test are between rewiring and flattening approach. Test cannot be performed on DMOZ-2010 and DMOZ-2012 dataset because we do not have access to true labels.

Table 7: Total training runtimes (in mins). ‘NS’ denotes not scalable.

Group Similar Classes - Figure 6 shows the sorted (descending order) class pairs cosine similarity scores for DMOZ-SMALL dataset. We can see that similarity scores becomes nearly constant after 1000 pairs (and drops further after 6000, not shown in the Figure) that does not provides any interesting similar classes grouping information for taxonomy modification. So for this dataset choosing threshold τ as the similarity score of the 1000-th class pair is a reasonable choice. Similar approach to determine the threshold can be applied for other datasets as well.

Effect of Varying the Training Size - Figure 7 shows the $M F_1$ comparison of best flattening, Global-INR and rewiring approaches on CLEF and DMOZ-SMALL datasets with varying percentage of training size. For both datasets we can see that rewiring approaches outperforms the flattening.
Figure 7: $MF_1$ performance comparison of rewar
ing approaches with best flattening, Global-INR model with varying % of training size. T-Easy approach is not scalable for DMOZ-SMALL dataset.

Table 8: Number of elementary operation executed for rewar
ing approach. Interestingly, for CLEF dataset with smaller train
ing percentage our proposed rewar
ing approach has bet
ter performance. Reason for this behaviour might be the
to-fitting of the optimal hierarchy with the training data in
case of the T-Easy approach, which results in poor perform
cance on unseen examples. For training dataset with eno
g examples as expected, T-Easy method gives the best perfo
mance but at the cost of expensive runtime. We cannot run T-Easy on DMOZ datasets.

| # elementary operation executed | CLEF | DIATOMS | IPC |
|---------------------------------|------|---------|-----|
| T-Easy [20] (promote, demote, merge) | 52   | 156     | 412 |
| Proposed Rewiring Filter Model (NC, PCRewire, ND) | 25   | 34      | 42  |

4.5.4 Comparison to Flat method

Table 9 shows the $MF_1$ and $hF_1$ performance compari
on of our proposed best flattening, Global-INR model and the
filter based rewar
ing model against the flat one-vs-rest LR
model on larger DMOZ datasets broken by varying dis
tribution 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
online evaluation. We use $MF_1$ for comparison because it
gives equal importance to all classes while evaluation.
For $hF_1$, we have used the original hierarchy for consisten
evulation. We can see that for rare categories (few training ex
amples per class) our proposed filter based and Global-INR
model outperforms the flat LR model. The hierarchy pro
vides useful information in categorizing these classes with
less training examples. However, for classes with enough train
ning examples flat LR model gives better results due to
better generalized models learning. It should be noted that
flat models have very expensive prediction time in compar
ison to our proposed TD models because it invokes all the
models while making prediction. For DMOZ-2010 dataset,
flat model takes ~90 mins for predicting the labels of test in
stances whereas our models takes ~20 mins. On comparing,
our proposed filter based rewar
ing model with the Global
INR model, we can see that rewar
ing model has better perfo
mance because the modified hierarchy using rewar
ing approach is based on the similarity score which leverages the
hierarchy better in comparison to flattened hierarchy.

Table 9: $MF_1$ and $hF_1$ comparison of our proposed approaches against flat LR model for DMOZ datasets with varying distribution of training examples per class.

5. CONCLUSION AND FUTURE WORKS

In this paper, we proposed two different filter-based data-driven approaches for taxonomy modification. We derived
a flattening based approach, that selectively removes the inco
sistent nodes from the hierarchy based on the deviation from
mean, and rewar
ing approach, that modifies the
existing parent-child relationships based on the similarities be
tween the classes. Our proposed global rewar
ing method ex
clusively outperform the existing flattening methods in the
literature, whereas our proposed rewar
ing method gives com
petitive results with much better runtime performance that
allow HC on significantly larger datasets (DMOZ). Extensive
analysis with varying training percentage and distribution per class is done to better understand the behaviour of
our proposed approaches with existing methods. In future, we
plan to extend our proposed methods for hierarchical multi
label datasets where instances can belong to more than one
class. We also plan to study the effect of our methods in con
juction with feature selection.

6. ACKNOWLEDGEMENTS

NSF grant # 1447489 and 1252318.

7. REFERENCES

[1] C. C. Aggarwal, S. C. Gates, and P. S. Yu. On the merits of
building categorization systems by supervised clustering. In
SIGKDD, pages 352–356, 1999.
[2] 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, pages 336–343, 2013.
[3] R. Babbar, I. Partalas, E. Gaussier, and M.-R. Amini. On
flat versus hierarchical classification in large-scale
taxonomies. In NIPS, pages 1824–1832, 2013.
[4] L. Cai and T. Hofmann. Hierarchical document
categorization with svms. In CIKM, pages 78–87, 2004.
[5] S.-L. Chuang and L.-F. Chien. A practical web-based
approach to generating topic hierarchy for text segments.
In CIKM, pages 127–136, 2004.
[6] I. Dimitrovi}c, D. Kocev, S. Loskovska, and S. Dzeroski.
Hierarchical classification of diatom images using predictive
clustering trees. Ecological Informatics, 7:19–29, 2012.
[7] I. Dimitrovi}c, D. Kocev, S. Loskovska, and S. Dzeroski.
Hierarchical annotation of medical images. Pattern
Recognition, 44(10):2436–2449, 2011.
[8] S. Dumais and H. Chen. Hierarchical classification of web content. In *SIGIR*, pages 256–263, 2000.
[9] T. Gao and D. Koller. Discriminative learning of relaxed hierarchy for large-scale visual recognition. In *ICCV*, pages 2072–2079, 2011.
[10] S. Gopal and Y. Yang. Recursive regularization for large-scale classification with hierarchical and graphical dependencies. In *SIGKDD*, pages 257–265, 2013.
[11] D. Koller and M. Sahami. Hierarchically classifying docs using very few words. In *ICML*, pages 170–178, 1997.
[12] T. Li, S. Zhu, and M. Oghihara. Hierarchical document classification using automatically generated hierarchy. *J. of Intelligent Information Systems*, 29(2):211–230, 2007.
[13] 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. *SIGKDD*, 7(1):36–43, 2005.
[14] H. Malik. Improving hierarchical svms by hierarchy flattening and lazy classification. In *Large-Scale Hierarchical Classification Workshop of ECIR*, 2010.
[15] A. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *ICML*, pages 359–367, 1998.
[16] K. Nitta. Improving taxonomies for large-scale hierarchical classifiers of web docs. In *CIKM*, pages 1649–1652, 2010.
[17] K. Punera, S. Rajan, and J. Ghosh. Automatically learning document taxonomies for hierarchical classification. In *WWW: Special interest tracks and posters*, 2005.
[18] X. Qi and B. D. Davison. Hierarchy evolution for improved classification. In *CIKM*, pages 2193–2196, 2011.
[19] A. Sun and E.-P. Lim. Hierarchical text classification and evaluation. In *ICDM*, pages 521–528, 2001.
[20] L. Tang, J. Zhang, and H. Liu. Acclimatizing taxonomic semantics for hierarchical content classification. In *SIGKDD*, pages 384–393, 2006.
[21] X.-L. Wang and B.-L. Lu. Flatten hierarchies for large-scale hierarchical text category. In *ICDM*, pages 139–144, 2010.
[22] L. Xiao, D. Zhou, and M. Wu. Hierarchical classification via orthogonal transfer. In *ICML*, pages 801–808, 2011.
[23] G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *SIGIR*, pages 619–626, 2008.
[24] Y. Yang and X. Liu. A re-examination of text categorization methods. In *SIGIR*, pages 42–49, 1999.
[25] A. Zimek, F. Buchwald, E. Frank, and S. Kramer. A study of hierarchical and flat classification of proteins. *TCBB*, 7(3):563–571, 2010.