Hierarchical Label Partitioning for Large Scale Classification

Raphael Puget, Nicolas Baskiotis

Hierarchical Label Partitioning for Large Scale Classification

Raphael Puget
Sorbonne Universités, UPMC Univ Paris 06,
CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris
firstname.lastname@lip6.fr

Nicolas Baskiotis
Sorbonne Universités, UPMC Univ Paris 06,
CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris
firstname.lastname@lip6.fr

***Abstract*—Extreme classification task where the number of classes is very large has received important focus over the last decade. Usual efficient multi-class classification approaches have not been designed to deal with such large number of classes. A particular issue in the context of large scale problems concerns the computational classification complexity : best multi-class approaches have generally a linear complexity with respect to the number of classes which does not allow these approaches to scale up.**

Recent works have put their focus on using hierarchical classification process in order to speed-up the classification of new instances. Using a priori information on labels such as a label hierarchy allows to build an efficient hierarchical structure over the labels in order to decrease logarithmically the classification time. However such information on labels is not always available nor useful. Finding a suitable hierarchical organization of the labels is thus a crucial issue as the accuracy of the model depends highly on the label assignment through the label tree.

We propose in this work a new algorithm to build iteratively a hierarchical label structure by proposing a partitioning algorithm which optimizes simultaneously the structure in terms of classification complexity and the label partitioning problem in order to achieve high classification performances. Beginning from a flat tree structure, our algorithm selects iteratively a node to expand by adding a new level of nodes between the considered node and its children. This operation increases the speed-up of the classification process. Once the node is selected, best partitioning of the classes has to be computed. We propose to consider a measure based on the maximization of the expected loss of the sub-levels in order to minimize the global error of the structure. This choice enforces hardly separable classes to be group together in same partitions at the first levels of the tree structure and it delays errors at a deep level of the structure where there is no incidence on the accuracy of other classes. Experiments on real big text data from recent challenge assess the performances of our model.

I. Introduction

Classification with large number of classes has received recently an increased interest. The democratization of devices able to record information has led to datasets always bigger. Bioinformatics, images annotation or text classification are few examples of tasks where efficient extreme classification could be used. These new datasets easily contain more than thousands of classes and it can reach the million of classes in some cases. Multi-class classification can concern either single label classification (an example belongs to a single class) or multi-label classification (multiple classes can be assigned for each example). In the following we will focus on the single label classification problem.

Methods that were specifically designed in the past decades for the multi-class classification are not well adapted to deal with a very large number of classes : these methods have a classification complexity linear with the number of classes at best. This complexity hinders the use of those methods in many industrial contexts where classification time efficiency is required (on-line services for instance). Moreover, with the constant increased of the number of classes, developing methods that have a classification complexity scaling better than linearly with the number of classes is of great interest.

Among the usual methods, the most popular one is the one-versus-all scheme, which consists in learning for each class a classifier separating the class from the other ones. The classification process for a given example consists in predicting the class corresponding to the classifier with the highest confidence score among all the classifiers. This method remains one of the most accurate classification methods while being excessively simple to implement. Even if it has been shown that in some contexts, it is possible to improve the performance of one versus all methods [1], [2], the simplicity and robustness of this framework [3] remains a solid choice when scalability is not an issue. It is also by nature easy to parallelize, although it has a classification complexity time linear with the number of classes which limits its usability in the context of fast large classification tasks.

Various models have been proposed in order to speed-up the classification time of the prediction of one example. Among them, ensemble methods offer an easy and natural way to control the computational complexity of a model by adapting the number of classifiers that are aggregated to compose the model. Such framework is provided by ECOC class encoding [4], [5], [6]. Several works [7], [8], [9] proposed alternatives by varying the coding and decoding processes. [10] showed an alternative and efficient way to learn an appropriate encoding in the context of large scale classification so as to improve the efficiency of each code bit. This approach provides a link between ECOCs and representation learning. Representation learning methods have been proposed in [11] and [12] where they try to embed the data using an existing taxonomy.

Another widely studied approach consists in organizing classifiers in a hierarchical fashion in order to limit the number of computations needed during classification process : to classify a given example, only few classifiers are used from

the available set which are in charge to refine at each step the set of label candidates. Precursor of these methods, [13], uses classifiers organized in a Directed Acyclic Graphs (DAGs) in order to improve accuracy and classification complexity, but this approach have to learn a quadratic number of classifiers with respect to the number of labels which is highly intractable in our context. Using a label tree structure is the most common hierarchical classification model. It consists in a tree where each inner node is associated with a subset of classes and a function (generally a classifier or a set of classifiers) which takes as input an example and predicts the next node in the hierarchy among the children of the considered node. Each leaf of the tree has one class associated to it thus any example reaching this leaf is then labeled with the class associated to that leaf-node. Considering a complete binary tree with a distinct label per leaf, the time complexity of the classification process is thus logarithmic with respect to the number of labels. However, embedding labels in such hierarchy is not an easy task and the accuracy of the model dramatically depends on the label partitioning and on the capacity to separate effectively at each node the children subsets of labels [14]. When a label hierarchy is provided with the learning task, the natural label tree deduced from the hierarchy can improve performances [2]. Otherwise, the label tree has to be inferred from the available data according to separability measures or other heuristics [12], [15], [16]. We will review in details recent research in hierarchical multi-class classification in section II.

In this work, we propose a new hierarchical classification algorithm for large scale multi-class problems where no supplementary information on labels is available (such as a class taxonomy). It is composed by two components, the first one in charge to optimize the hierarchical structure of the label tree in order to improve classification complexity and the second one in charge to optimize the label partitioning at a given node in order to improve the accuracy of the model. Starting from a flat tree structure with only one inner node (the root node) and as many leaves than labels, our algorithm iteratively selects a node to expand, i.e. to introduce intermediary nodes between the considered node and its children in order to reduce the classification complexity. The second component is a label partitioner that will choose how to distribute the labels in the new children in order to maintain a high classification accuracy. The partitioning process is based on a simple reasoning: two classes that cannot be distinguished are doomed to make errors in the hierarchy [25], [12]. Thus, the idea is to prevent these classes to affect negatively the rest of the classification process and the node responsible for the separation between these two classes has to be as deep as possible in the label tree. The contribution of our model regarding this aspect is to provide a new iterative algorithm that computes a distance measure between one class and a cluster in order to refine the partitioning at each iteration. As a result, our approach is able to iteratively drag deep down in the hierarchy these classifier type specific errors which will result in two wanted outputs: 1/ the hard separation will be eased if there are few classes to be taken into account by the classifier and 2/ the upper level separation problems will be easier to deal with. The Figure 1 shows a basic example of the partitioning intuitive functioning.

The paper is organized as follow : notations and related work are presented in the section II; Section III presents our algorithm and its two components; Section IV presents experimental results on real world dataset from recent large scale multi-class challenges. Finally, section V presents conclusions and perspectives of the proposed model.

II. Hierarchical multi-class classification

In this section, we introduce the hierarchical multi-class classification problem and the notations we will use in the following; next, we review and discuss recent approaches.

A. Notations

Given a label set $Y = \{0, \dots, K\}$ with K very large ($K > 10,000$) and a training dataset of examples $\{(x_i, y_i)\}$ with $x_i \in \mathbb{R}^D$ and $y_i \in Y$ the label corresponding to the example x_i, the considered classification task is to build a model f able to output correctly the label corresponding to a new example. We will consider in the following the usual zero-one loss :

$$\underset{L}{\text{minimize}} \sum_i \mathbb{1}(f(x_i) \neq y_i) \tag{1}$$

A label tree is noted by the tuple $T = (N, E, F, L)$, with N the set of nodes indexed by $\{1, \dots, n\}$, $E = \{(i, j), \dots\}$ the set of edges (parent, child), $L = \{s_1, \dots, s_n\}$ the set of labels in each node. By definition, the node 1 is the root node. Thus we have $s_1 = Y$. For each leaf-node e, we have $|s_e| = 1$. For each pair of nodes p, c with $(p, c) \in E$ (c child of p), we have $s_c \subset s_p$. Let us precise that we are manipulating a tree structure here and not a DAG, thus there is only one path between the root and each leaf.

The set $F = \{f_2, \dots, f_n\}$ corresponds to the set of decision functions associated to each node which will be used to compute the path from the root to a leaf node for an instance to classify. Each function f_i is in fact associated to the edge just *above* the node i and indicates how likely is an instance to belong to one of the labels of the node s_i. Thus, the root node has no such function. The classification process for a new instance begins at the root node which *contains* every possible labels. At each step, given the current node, the next node is computed by selecting among the children the one whose the associated decision function outputs the maximal value on the given instance. The instance goes through all levels of the tree until it reaches a leaf. The reached leaf gives then the label predicted for the considered instance. The Algorithm 1 details this classification process. The computational complexity of the process is the sum of the degrees of the nodes in the classification path. Thus the complexity depends on the structure of the tree, from linear in number of classes when the label tree is a flat structure with a root node and K leaves, to logarithmic with respect to the total number of classes by considering a full binary tree as label tree.

B. Related Work

The task of learning the tuple T has been largely studied [1]. When prior hierarchical information is provided with the datasets, the task focuses mainly on the learning of the classifiers of the set F. These classifiers can be local to the nodes of the hierarchy [17], [18], [19] or one can train a global classifier taking into account the underlying hierarchy [20],

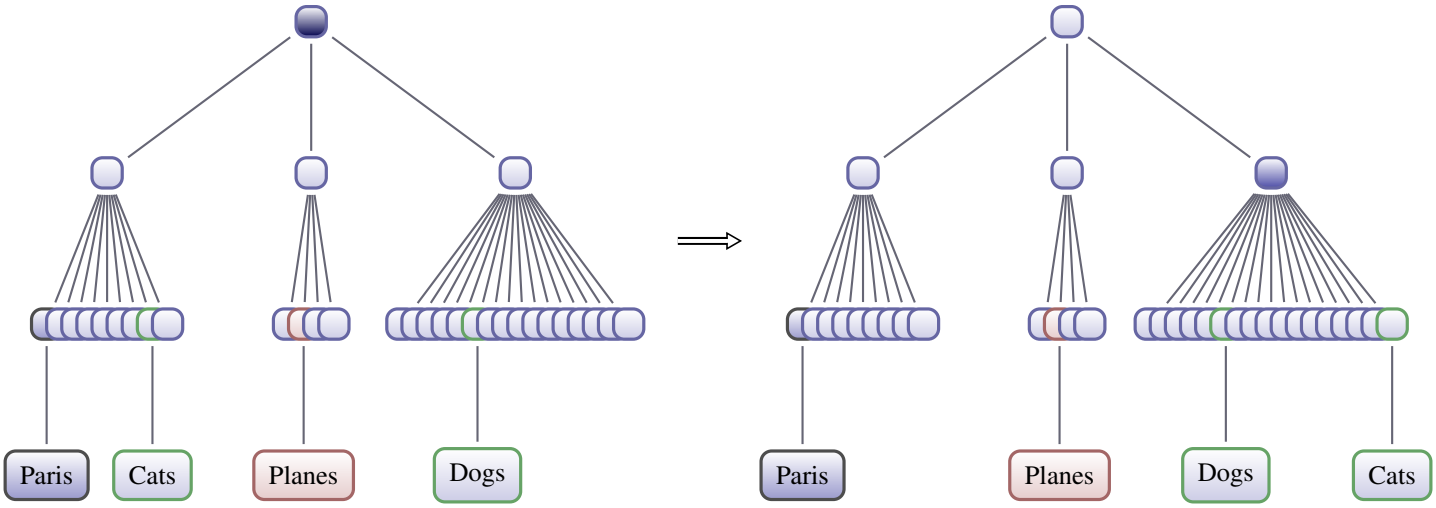

Fig. 1: Partitioning visual example. Let us consider two classes: {Cats} and {Dogs}. At the beginning, the classifying process at the root is hard (dark node) because clusters are not really homogeneous. By computing separability with each cluster, it appears that {Cats} *confused more* with the cluster with {Dogs} inside. Then, {Cats} class is relocated in the same cluster as {Dogs}. This will improve the global {Cats} vs {Dogs} separability accuracy and it will also ease the top level separation as the clusters are more homogeneous.

Algorithm 1 Tree Label Classification

Require: {example : x — $rootNode$ of a trained tree $T = (N, E, F, L)$}
$a \leftarrow rootNode$
while a is not a leaf **do**
$a \leftarrow \arg\max_{\{c\,:\,(a,c)\in E\}} f_c(x)$
end while
return s_a

[21], [22]. These methods achieved generally high accuracy performances as the hierarchical label information provided as input is often correlated with the learning problem. Works on refinement of an existing label tree has been proposed in [2], [23], [24] where accurate modifications of the tree structure allowed an improvement of the performances.

However, sometimes the hierarchy provided is either non existing or not useful for the given classification task. For instances, a hierarchy of medicinal plants based on their therapeutic property would be of no use if the task is to differentiate them using only image input.

In this work, we consider the general case where no hierarchical information on labels is provided with the learning problem and a suitable hierarchy has to be discover ex-nihilo. The proposed work focuses on building N, E and L. N and E can be seen as the skeleton of the classifying tree whereas L corresponds to the label assignment to each node of the tree (which labels are associated to which nodes). The classifiers F are learnt independantly once the structure and the partitioning are established. Those three components (N,E,L) can be learnt altogether but it is often difficult to manage such degree of freedom in the learning of the hierarchy. Usually, the structure (N,E) is locked by specifying a parameter controlling the number of children per node. Then, using a top-down approach, the label assignment is learnt at each non-leaf node of the tree.

For instance, spectral clustering on interclass confusion matrices [25], [12], [15] has been proposed for learning L on a pre-built skeleton N-E in a top-down fashion. Specifically, these approaches aim to compute homogeneous clusters of classes using the confusion between classes. A parameter controls the number of children per node globally for every node and it has to be specified by the user. This is actually the only parameter which control the trade-off accuracy versus classification computational complexity. The density class balance is not controlled and this can lead to unbalanced tree. Other work proposed to learn a linear discriminant projection in order to automatically build the hierarchy [26]

In theory, the *fastest* tree is a binary label tree with perfect balance of the classes among the nodes (weighted or not by the size of each class). By decreasing the number of children per node, the classification tasks are generally harder but the computational complexity decreases if we consider that we are using one-versus-all scheme multi-class classification inside nodes. At the opposite, if the number of children per node is equal to the number of classes, then the tree is almost flat and equivalent to a standard one-versus-all scheme, with high accuracy but poor classification time complexity.

With the addition of redundancy in the label assignment [16], a more precise control of the trade-off accuracy versus computational complexity is possible while learning L and F simultaneously in a top down fashion. In this case, a same label can be found at different nodes in the same level of the hierarchy and in multiple leaves. Such a redundancy allows a better error recovery as one single error of one node does not anymore mean an error at the end of the classification process

but increases slightly the classification time complexity.

Other works proposes to learn F before the node labeling L. In [27], a clustering of the example is learnt at the root node. Then a label assignment L is learnt for each node using pre-computed one-versus-all scorer models that will be used finally as F component in each node to make the final prediction. The tree obtained with this model is always of depth 3 with the mid level being the result of the label assignment. Recursive non-negative matrix factorization has been proposed [28] to learn L by solving minimization of an information-theoretic loss function at each step. [29] tried to globally learn L and F by masking label in the nodes. At each step, confusing classes are not taken into account in the training of the node classifier. Another way around is to build in a bottom-up fashion the label tree. [30] developed a metric between classes based on features frequency for each class. [31] used a one-class SVM in order to compute its proximity metric between classes. The problem of reducing the training time complexity has been explored with the conditional probability tree [32] model where the label tree is built in an online fashion, and more recently in [33].

C. Discussion

The aforementioned approaches output generally a fixed tree structure depending on control parameters (number of children per node for instances) without exploiting actively the different shapes that a real hierarchy may require. Most of suitable approaches for large scale problem use a greedy exploration algorithm in a top-down partitioning scheme or bottom-up aggregative scheme which brings long term control issues for the future steps.

We propose in this work a greedy algorithm which aims to learn a suitable structure N and E, and the partitioning of the label L simultaneously. Beginning from a flat tree structure with a root node and only leaves, each step of the proposed algorithm iteratively expands node in the label tree by partitioning the labels present into that node in a number of children. Two contributions are proposed in the following : 1) a global iterative learning algorithm that will build successively the tree structure N and E by selecting most favorable node to *expand*; 2) a partitioning algorithm for the learning of L that takes into account the family class of classifiers F that will be used inside each node in the final tree of classifiers. The global iterative learning algorithm is learning the middle stage of the hierarchy each time in order to avoid the problem of top-down and bottom-up greedy learning. The optimal number of children is computed each time in order to equally allocate computation charge above and under the building node level.

III. Hierarchical Label Partitioning

This section presents the proposed algorithm. Firstly we introduce the global scheme of our algorithm; next we present the algorithm responsible for the optimization of the hierarchical structure; finally we present our partitioning algorithm.

A. Overview of the approach

The main step of the proposed algorithm is to select an existing node in the tree in order to expand it into smaller clusters. Choosing the node to expand allows to control the structural evolution of the label tree and to control the speed-up in terms of classification time complexity. Optimizing the assignment of labels among the new created nodes is highly correlated to the final accuracy of the inferred model. Unlike tree partitioning label in recent state of the art methods that try to build successively the tree by beginning at the top (top-down approaches) or at the bottom (bottom-up approaches), our model begins by inferring the label clusters that will be located in the middle level of the final label tree. The Fig. 2 shows the functioning of the global algorithm.

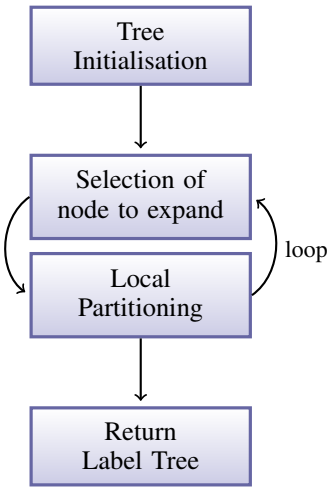

Fig. 2: Functional diagram of the overall algorithm. At the tree initialization, the tree is flat: it has one root and K leaves with the K different labels. At each step of the loop, one node is selected and a partitioning of its children is computed. When the aimed complexity is reached, the algorithm end. Classifiers inside each node are then trained.

Once the node to expand has been selected, the label partitioning at this node begin. To grasp the idea behind what the algorithm does, let us discuss about the location of the closest ancestor node for two classes. This closest ancestor node is the lowest node in the hierarchy containing the two classes. The ancestor node of two classes that make mistakes is better situated deep-down in the hierarchy (at the lowest level possible) because the difficult task of separating these two classes won't affect the classifiers accuracy of nodes on top (i.e. higher) of their ancestor node. Furthermore, a classifier trained to separate two classes is more likely to be more accurate in his task compared to the classifier trained to separate the same two classes augmented with other external classes. Thus, the motivation of globally lowering the closest ancestor node is beneficial to the final hierarchy by helping difficult classes separation by making it happened at lower levels and by preventing to impact negatively the accuracy of top level classifiers with difficult class separation problems.

The proposed algorithm iteratively builds clusters in order to ensure that the closest ancestor node for all the hardly separable pairs of class subsets is the lowest possible.

The learning of the label tree partitioning can be divided into two parts: 1/ the overall algorithm that select which node to expand (Section III-B) and 2/ the aforementioned partitioning algorithm that details how the partitioning is done

given a node and its set of labels to separate (Section III-C).

B. Iterative Expansion Algorithm

The iterative expansion algorithm aims at selecting the nodes to expand in order to build successively the label tree. Moreover, it dictates also the number of cluster that should be aimed at during the partitioning process. There are then two problems to solve. The first one is to select the node to expand and the second one is to select the size of the intermediary level that is beeing created. Fig. 3 shows the expansion of one node of the hierarchy by creating an intermediary level.

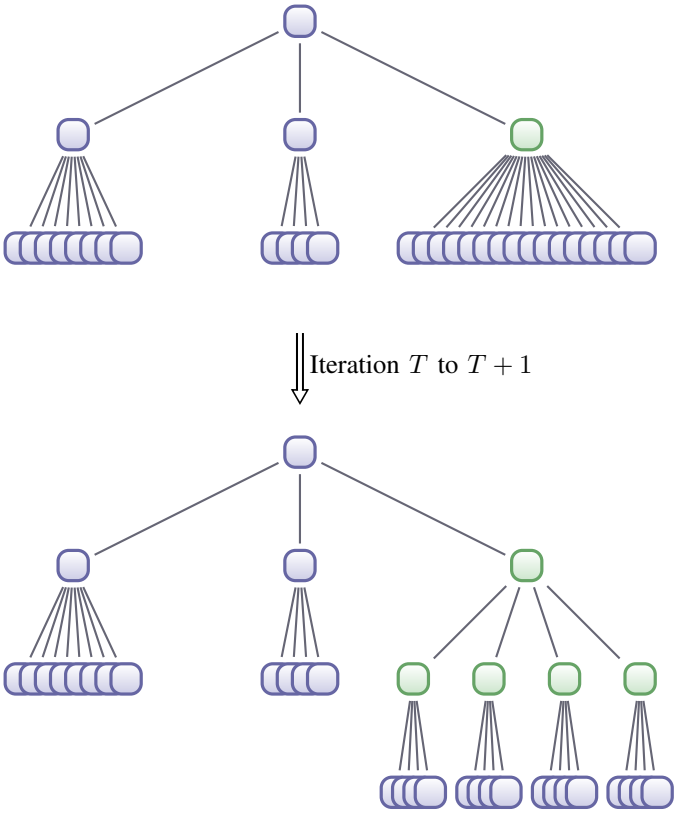

Fig. 3: Effect of one label partitioning iteration. The green node is the one that will benefit the most from an additional level of label partitioning.

For the first problematic which is the selection of the node to expand, our goal is to reduce the classification time computation. In order to measure the speed-up of our model, we introduce the complexity ratio. The complexity ratio for a model is the ratio between the equivalent number of classifier evaluations used by the model and the number of evaluations that would use a one-versus-all method. The number of classifier needed by a one-versus-all strategy is equal to the total number of labels. The complexity ratio of a given model is superior to one if it is slower than a one-versus-all strategy and inferior to one if it is quicker to classify the class of a new example. This complexity ratio take into account the example density specific for each classes.

Given a complexity ratio to reach, the algorithm will iterate successively the partitioning process of nodes until the aimed complexity ratio is reached. To do so, we can define a local complexity ratio regret for each node of the hierarchy. This is simply the difference between the optimal local complexity ratio possible and the current local complexity ratio (which is always one):

$$\mathcal{R}_{comp}(n) = \frac{1}{|s_n|}\left(|s_n| - \log_2(|s_n|)\right)$$

The algorithm selects the node which has the highest complexity regret in order to add a hierarchy level under this node. Algorithm 2 describes the overall iterative learning process to build the label tree. At the beginning, the tree is flat: one root and K leaves directly under the root. At each iteration, the node with the maximum cost is selected to be *expanded* by using the partitioning algorithm that we will detail further. When the overall tree is able to classify as fast as required, the algorithm stops and the classifiers inside each node are trained.

The complexity ratio that control when to stop the expansion of the tree T is estimated by the function f_{comp}:

$$f_{comp}(T) = \sum_{n \in \mathcal{A}} w_n \frac{|s_n|}{K} |\mathcal{C}_n|$$
$$\text{where } \mathcal{A} = \{n \,|\, |\mathcal{C}_n| > 0\}$$

with $\mathcal{C}_n$ the set of children of the node n and w_n the weighting that consider the example density of the class n normalized over the whole training set.

That function computes the expected sum over the computational cost of each node by example divided by the number of class. The cost per node is proportional to the size of the node and to the number of children of this node.

Algorithm 2 Learning the label tree

Require: Tree $T = (N, E, F, L)$
Require: m the aimed complexity ratio.
  while $f_{comp}(T) > m$ **do**
    $\hat{n} = \arg\max_n \mathcal{R}_{comp}(n)$
    Compute partitioning : $\mathcal{P}_{\hat{n}}$
    Update Tree T
  end while
  return T

Once a node has been selected to be expanded, we need to select the size of the intermediary level that is created. The goal is the same as previously : minimizing the complexity ratio. Let us define as d_0 the out-degree of the top node and d_1 the out-degree of all the nodes of the intermediary level. Then, the resolution of the following system : $\text{minimize}\, d_0 + d_1 \quad \text{s.t} \quad d_0\ d_1 = K, d_0 > 0, d_1 > 0, K > 0$ corresponds to the optimal structure parameters to reach the lowest possible complexity ratio. The system is solved when $d_0 = d_1$ which means that $d_0 = \sqrt{K}$.

This solution has an interesting property : the intermediary level that is created at one node n will be, if the tree is fully expanded, equally distant from the node n and from the leaves. This means that the algorithm is building the hierarchy by

infering the middle level of it first. In fact, the usuals top-down approaches suffer from the lack of visibility of several steps ahead. That means that the partitioning at the root node is not easy as the clusters found may not be suitable for the full label tree organization. The bottom-up approaches have to deal with the aggregating task where every pairs of classes behave similarly as the space dimensionality is very large. With our method which infer the mid level clusters between two existing levels of the label tree, we do not have the same issues.

For instance, if our label tree has only two levels with the root node at the first one and 10,000 leaves at the second one, the size and number of middle clusters wanted is 100 as it will evenly distribute the workload of the hierarchy: the root node will spread into 100 nodes and each of these node will spread into 100 leaves. By building this mid-level of the label tree, each example will go through 200 classifiers ($100 + 100$) in place of the 10000 before the addition of the new mid-level. By taking the square root of the number of children, we provide the greatest speed-up possible.

C. Label Partitioning Optimization

The goal of this algorithm is to partition the set of children nodes of a particular node of the tree. Given a node n, its set of children $\mathcal{C}_n$ and a number Q of clusters to build , the label partitioning process aims at learning an optimal partitioning $\mathcal{P}_n = \{p_1, \ldots, p_Q\}$ with $p_i \subset \mathcal{C}_n$ and $\bigcup_i p_i = \mathcal{C}_n$ and $\forall (p_i, p_j), p_i \cap p_j = \emptyset$. To avoid to converge to unbalanced clusters, a maximum p_{max} number of node per cluster can be set. Experimentally, it can be automatically set to force perfect balance of nodes in the clusters.

The principle goal of the partitioning process here is to force nodes that are not easily separable to be in the same cluster. Thus, for a given class, our main focus is to guide it to the cluster that will hardly dissociate it from the other cluster classes.

Let us define $f_{y,s}$ the classifier trained to separate class y from a set of classes s that returns $+1$ if x is inferred as being from class y, and -1 if the example is inferred as being from the classes set s. We define $e(y, s)$ the sum of the classification errors of the examples of the label y when the label y belong to the cluster s:

$$e(y, s) = \sum_{i|y_i=y} \mathbb{1}(f_{y_i,s}(x_i) \neq 1)$$

Then, we want to maximize this *intra* errors over all the clusters:

$$\begin{aligned}
\underset{\mathcal{P}}{\text{maximize}} \quad & \sum_{p \in \mathcal{P}} \sum_{y \in s} e(y, s_p) && \text{(2a)} \\
\text{subject to:} \quad & \forall (p_i, p_j),\ p_i \cap p_j = \emptyset && \text{(2b)} \\
& \bigcup_i p_i = \mathcal{C}_n && \text{(2c)} \\
& \forall p,\ \|p\| \leq p_{max} && \text{(2d)}
\end{aligned}$$

As usual with integer programming, the optimization problem is intractable as is. We propose the following relaxation consisting in a two steps iterative algorithm : at first, a random classes assignation is drawn for each cluster. At each iteration, the algorithm computes classifiers for each class $\hat{y}$ against all the classes containing in each cluster. It updates next the partitioning by assigning for each class $\hat{y}$ the cluster with the highest summed loss over the examples of the class $\hat{y}$. The algorithm is given in Algorithm 3. The algorithm terminates when the loss is stable, i.e. when the new loss is inside an ϵ-ball distance from the last one. Practically, if the training of classifiers is costly, it is possible to use one-class classifiers in order to factorize the workload and greatly reduce the number of classifiers to train.

Algorithm 3 Tree Label Classification

Require: { node n, nb of clusters Q }
  Initialise randomly Q clusters : $\mathcal{P}$
  while Computed loss not stable **do**
    for each class y **do**
      for each cluster p **do**
        Compute classifiers : f_{y,s_p}
      end for
    end for
    Assign label y to cluster $\arg\max_p e(y, s_p)$
  end while
  return $\mathbb{P}$

IV. Experiments

A. Datasets

Two public datasets were used to assess our model. The first one Sector is a dataset of web page texts with 105 classes [34]. The second one is the DMOZ dataset (or ODP for Open Mozilla Directory) that was used during the recent challenge LSHTC [35]. It is a large text documents classification task with more than $12,000$ different classes. These classes are organized in an pre-existing hierarchy. This hierarchical information were not used by the models tested here but it is valuable to compare the hierarchy that our model is able to build with the original one. However, a study of the hierarchy produced by our method were performed in order to compare it to the original hierarchy in Section IV-E. The test and validation sets were the same as the ones of the challenge. A word count and a TFIDF feature transformation have been applied on the raw documents by the challenge organizers. Statistics of these datasets are available in the Table I.

	DMOZ	Sector
# training instances	93805	6992
# validation instances	34905	1469
# test instances	34880	1158
# features	347255	55198
# classes	12294	105

TABLE I: Statistics of the datasets used.

B. Protocol

We first compared our methods with two state of the art label tree building methods. The first one is using spectral clustering as described in [12]. A confusion matrix is computed based on the validation set. Then, spectral clustering is applied

recursively in a top down fashion in order to build the label tree. The second one is building a single partition level label tree [27] by using k-means on examples of interest, and then performing a label assignment to these cluster based on the examples inside each cluster. We also computed classic ECOC methods as described in [4], [36] (with two different decoding processes) in order to compare our method to flat methods. Coding process has been done by randomizing 5000 thousands coding matrices. The coding matrix maximizing hamming distance between the code of each class is kept. Two different decoding are used: a classic hamming distance decoding, and a much more sophisticated loss based decoding using classifiers margins[6]. We computed one-versus-all scheme to have an idea of one of the best performers on these problems of large scale classification. Finally, we computed a label tree based only on the original hierarchy (when available).

For all these methods, the sub classifiers that were aggregated are all SVM linear models which proved to be very robust for these tasks. They have been regularized over the same validation set for each model[1].

As one of the main purpose of such hierarchy is to speed-up the classification process, we defined a measure to compare fairly the speed-up of each methods: the *complexity ratio* is simply the ratio between the number of classifiers used by the model and the number of classifiers that would have been used by a one-versus-all standard flat scheme (which actually uses a number of classifiers equal to the total number of classes). Thus, a complexity ratio of 1 means no speed-up compared to one-versus-all scheme. And a complexity ratio inferior to 1 means that the model use less classifiers than the reference flat model.

We did not add speed-up techniques using feature dimensionality reduction as we only aimed here at a structural reduction of the classification time by limiting the number of classifiers used for the classification of a new document.

We recorded the standard multi-class classification accuracy on test set for each models for different computational complexity ratio. For each model, we used parameters like the number of children per node, or the length of the code for the ECOC models, in order to achieve comparable complexity ratio.

C. Results

Tables II and III show accuracy results for different complexity ratio (speed) on the DMOZ dataset and on the sector dataset. Our model is able to produce slightly better results than standard one-versus-all scheme while performing the classification task significantly faster ($\times 55$). The other methods able to produce comparable speed-up are not able to achieve similar performances. With the increase of the speed-up, the performances of all the methods decrease rapidly. Still, our approach outperforms others with a significant margin. The results from the single partition level model were not satisfying as the k-means lead to highly non balance clusters on these datasets, and thus this model results are not reported in the tables. We put for reference only the performance of the label tree using the original hierarchy information (*Hierarch-Ontology*). As this information was not used by all the compared methods, the performance obtained by this model should not be compared directly. Each column indicates performances of the models for one specific speed-up. When the model is not able to produce a model adapted for a given speed-up, a dash ($-$) is inserted in the corresponding cell. For instance, one-versus-all scheme results appear only in the first column with no speed-up as it is impossible for it to reduce the number of classifiers to use to get a faster prediction. The graph in Fig. 4 shows the accuracy evolutions of the different models with the variation of the complexity ratio. For our approach, the *worst* complexity ratio is obtained after one iteration of the algorithm. However, we designed the process to optimally reduce de classification complexity. Thus, our *slowest* point is not that slow compared to other methods which explains why we do not have slower points.

D. Discussion

It is interesting to note that ECOC type model suffers more of extreme speed-up compared to hierarchical model: at the first complexity ratio, the Ecoc-LossBased shows better performance than the hierarchical clustering, while at the most extreme complexity ratio, the hierarchical clustering model performs better. This is explained by the inherent property of hierarchical model. It is very easy to achieve strong computational classification time reduction with a hierarchical model. With such model, the different classifiers are optimally chosen : there is one at the root which deal with all the classes. Then, for each lower levels, classifiers become more specific as the number of labels is strongly reduced. With ECOC models, classifiers are much similar with one another. Thus, by reducing drastically the number of classifiers, it appears rapidly a decrease of the performance as it lacks specific information to separate every class to every other classes. Our model is able to produce better results than slower flat model with the same type of classifiers. Even a label tree built directly from the classes hierarchy given with the DMOZ dataset produce slightly less accurate predictions. It has to be noted that the task asked during the challenge allowed the use of the hierarchical information which explains the better results obtained by the top participants.

The sector dataset is quite different from the DMOZ one as it comes without any prior hierarchical organization of its classes. This does not mean that there are no hierarchical organization possible for this dataset. However, it appears that finding a hierarchy for the classes of this dataset were not as successful as finding one on the DMOZ dataset. Our approach still performs better than state of the art methods for similar speed-up but it is slightly less accurate than the one-versus-all model (which use far more computational power to do the classification of documents).

A study more in depth of the level specific errors during the iterations of the algorithm is helping to get the intuition of the heuristic used in our approach. Fig. 5 shows an example of the evolution of the accuracy of different level of the hierarchy during the partitioning of the 12294 classes of the DMOZ dataset. It can be noted that the algorithm successfully degraded the error of the second level by putting class into the

[1]The implementation comes from *scikit learn library* (`LinearSVC`). Similar parameters were used for all the compared methods : (*l2* loss, automatic class weighting and *l2* penalty).

Models	Type	Acc $\mathcal{C}$@1	$\mathcal{C}$@0.018 ($\times$55)	$\mathcal{C}$@0.0012 ($\times$83)	$\mathcal{C}$@0.0081 ($\times$125)
One against all - OAA	Flat	38.28%	-	-	-
Hierarch-Ontology	Tree	-	-	-	*38.05%*
Our	Tree	-	**38.58%**	**36.45%**	**33.71%**
Hierarch-Clustering	Tree	-	32.83%	32.78%	31.37%
Ecoc-Hamming	Flat	-	29.61%	25.06%	21.19%
Ecoc-LossBased	Flat	-	34.17%	32.08%	30.55%

TABLE II: Accuracy results for 13K classes on DMOZ dataset. We reported in bold font the best significant results for each speed-up. We put in italic the results using additional information. Each column is associated with a specific speed-up. The dash (−) character is used when the model can not be computed for the specified speed-up.

Models	Type	Acc $\mathcal{C}$@1	$\mathcal{C}$@0.2 ($\times$5)	$\mathcal{C}$@0.14 ($\times$7)
One against all - OAA	Flat	**94.1%**	-	-
Our	Tree	-	**93.7%**	**90.4%**
Hierarch-Clustering	Tree	-	88.5%	88.2%
Ecoc-Hamming	Flat	-	27.7%	18.3%
Ecoc-LossBased	Flat	-	89.6%	73.5%

TABLE III: Accuracy results on sector dataset with 105 classes. We reported in bold font the best significant results for each speed-up.

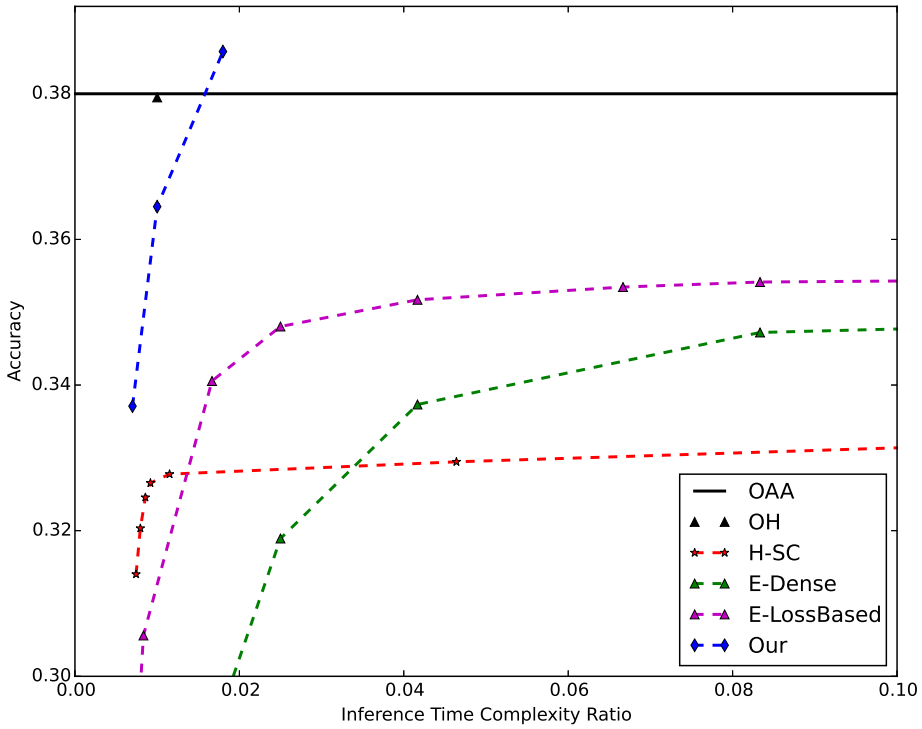

Fig. 4: Accuracy results on DMOZ dataset function of the complexity ratio. The black line is actually just the extension of the point corresponding to the one-versus-all which would be situated at 1 on the x axis if the full x axis was displayed here.

cluster which had the most issue with the classification of this class examples. By doing so, the algorithm refined the partition by computing homogeneous classes together. This leads to an improvement of the classification quality at the top level as the separability at the top is eased by the homogeneous partition.

E. Hierarchy Study

Although the aim of this work was not to produce an pertinent hierarchy, we compared the obtained hierarchies (our model and the spectral clustering) with the original one. One way to evaluate the pertinence of the hierarchy extracted is to simply compared it to a gold standard one as described in [37]. Thus, we compared the hierarchy obtained with the

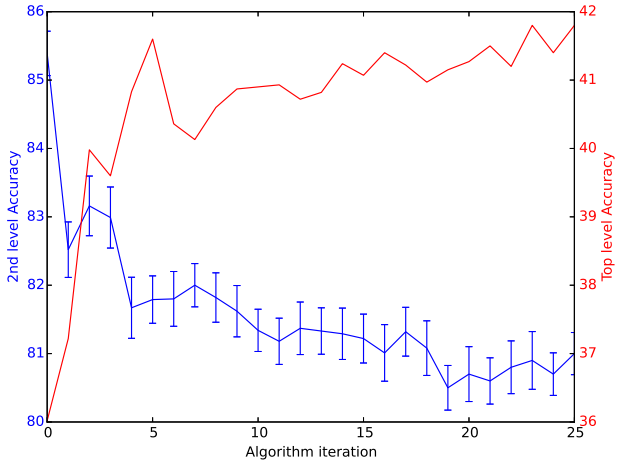

Fig. 5: Example of the behavior of the errors during partitioning algorithm. These two accuracy curves are the accuracies of the classification process at specific level of the hierarchy during the partitioning of the 12,294 classes into ~100 nodes. The top level is the one between the root and the ~100 nodes. The second level is the one between the ~100 nodes and all the leaves.

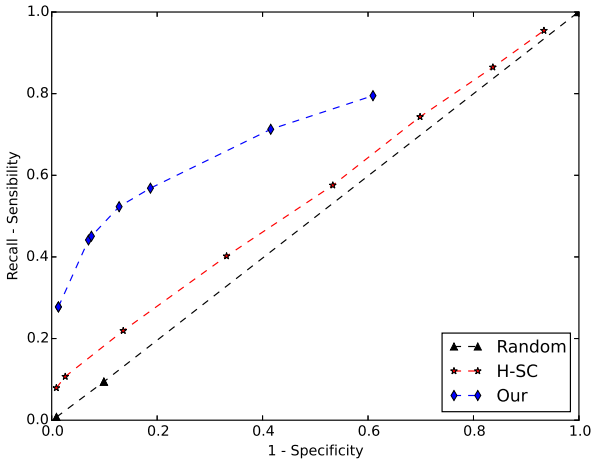

Fig. 6: Comparison of classes hierarchies computed with the different compared models versus the original hierarchy for the DMOZ dataset. The higher the precision and the recall are, the *closer* the model is from the original hierarchy.

hierarchical clustering baseline model and our partitioning model with the original class hierarchy given with the DMOZ dataset.

To do so, we define a neighborhood function $\mathcal{N}$ that take a class ℓ and a label tree T. Given a tree distance metric Δ, we can compute for that class the set of classes that are closer to a given radius θ in the label tree:

$$\mathcal{N}_\theta(\ell^*, T) = \bigcup_{\{\ell | \Delta_T(\ell^*, \ell) \leq \theta\}} \ell$$

.

Then, we compare the neighborhood obtained in the computed label tree with the neighborhood observed in the original hierarchy. Let us define T_o the label tree built using the original hierarchy. We can compute a *precision* of a computed label tree T:

$$\mathcal{P}_{\theta'}(T) = \frac{1}{K} \sum_\ell \frac{\mathcal{N}_{\theta'}(\ell, T) \cap \mathcal{N}_\theta(\ell, T_o)}{\mathcal{N}_{\theta'}(\ell, T)}$$

and a *recall*:

$$\mathcal{R}_{\theta'}(T) = \frac{1}{K} \sum_\ell \frac{\mathcal{N}_{\theta'}(\ell, T) \cap \mathcal{N}_\theta(\ell, T_o)}{\mathcal{N}_\theta(\ell, T_o)}$$

By varying the threshold θ', we obtain the curve Fig. 6. By doing similarly with the *specificity*, we can compute ROC curve seen Fig. 7. These were plot with an original threshold θ of 3 and by randomizing 1000 sub sampled classes to reduce the quadratic number of comparisons needed to compute the neighborhood.

Those graphics show the ability of our model to partially recover the original hierarchical information that has never been used by the model. The standard hierarchical clustering is able to recover some of that information compared to a random

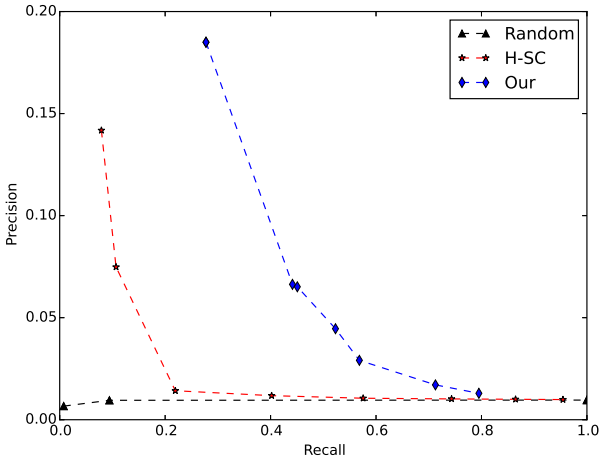

Fig. 7: Comparison of classes hierarchies computed with the different compared models versus the original hierarchy for the DMOZ dataset.

label tree. Our model is nonetheless far better at recovering this information.

V. Conclusion

We proposed in this paper a new algorithm for the hierarchical large scale multi-class classification problem. Starting from a flat tree with a large number of leaves - the entire set of classes - our algorithm selects iteratively a node and expands it in order to optimize the classification time complexity by creating a new level of nodes in the tree between the selected node and its children. We propose a partitioning algorithm based on the maximization of the error in each new node in order to ensure that difficult classes separation problems are rejected at the low levels of the tree and thus reduces globally

the error of the model. The proposed approach is generic and does not depend on a particular family of classifiers as the partitioning phase uses for the error estimation any classifiers family that will be used at the end by the model. This fact can explain the excellent experimental results obtained on real datasets compared to other hierarchical approaches which treat the partitioning problem and the problem of learning the decision function predicting the next node as two distinct learning problems. The other benefit of the proposed algorithm is that it explores a more diversified structure space as the tree structure is not fixed from the beginning of the process but rather optimized at each step.

Future work will focus mainly on two aspects. The first one is to extend the algorithm to redundant label trees, where a label can correspond to multiple leaves, which allows to achieve better classification performances. How to select which label to duplicate, at which level without hindering the classification complexity are questions of interest. The second one is to study the adaptation of this algorithm to the multi-label classification task where an instance can belong to multiple classes.

REFERENCES

[1] C. S. Jr and A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery*, vol. 44, no. 0, 2011.

[2] R. Babbar and I. Partalas, "On Flat versus Hierarchical Classification in Large-Scale Taxonomies," *Neural Information Processsing Systems*, pp. 1–9, 2013.

[3] F. Perronnin, Z. Akata, Z. Harchaoui, and C. Schmid, "Towards good practice in large-scale learning for image classification," *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3482–3489, 2012.

[4] T. Dietterich and G. Bakiri, "Solving multiclass learning problems via error-correcting output codes," *Journal of Artificial Intelligence Research*, 1995.

[5] R. Schapire, "Using output codes to boost multiclass learning problems," *International Conference on Machine Learning*, no. 1, pp. 1–9, 1997.

[6] A. Passerini, M. Pontil, and P. Frasconi, "New results on error correcting output codes of kernel machines." *IEEE transactions on neural networks*, vol. 15, no. 1, pp. 45–54, 2004.

[7] D. Hsu, S. Kakade, J. Langford, and T. Zhang, "Multi-Label Prediction via Compressed Sensing." *Neural Information Processsing Systems*, pp. 1–16, 2009.

[8] S. Escalera, O. Pujol, and P. Radeva, "On the decoding process in ternary error-correcting output codes." *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 1, pp. 120–134, 2010.

[9] G. Zhong and M. Cheriet, "Adaptive error-correcting output codes," *International Joint Conferences on Artificial Intelligence*, pp. 1932–1938, 2013.

[10] M. Cissé, T. Artières, and P. Gallinari, "Learning compact class codes for fast inference in large multi class classification," *European Conference on Machine Learninga*, 2012.

[11] K. Weinberger and O. Chapelle, "Large margin taxonomy embedding with an application to document categorization," *Neural Information Processsing Systems*, pp. 1–8, 2008.

[12] S. Bengio, J. Weston, and D. Grangier, "Label embedding trees for large multi-class tasks," *Advances in Neural Information Processing Systems*, vol. 23, no. 1, pp. 163–171, 2010.

[13] J. Platt, N. Cristianini, and J. Shawe-taylor, "Large margin DAGs for multiclass classification," *Neural Information Processsing Systems*, pp. 547–553, 2000.

[14] R. Puget, N. Baskiotis, and P. Gallinari, "Scalable Learnability Measure for Hierarchical Learning in Large Scale Multi-Class Classification," *Web Search and Data Mining Workshop Web-Scale Classification: Classifying Big Data from the Web*, vol. 6, 2014.

[15] G. Griffin and P. Perona, "Learning and using taxonomies for fast visual categorization," *Computer Vision and Pattern Recognition*, pp. 1–8, Jun. 2008.

[16] J. Deng, S. Satheesh, A. Berg, and L. Fei-Fei, "Fast and Balanced: Efficient Label Tree Learning for Large Scale Object Recognition," in *Neural Information Processsing Systems*, no. 1, 2011, pp. 1–9.

[17] D. Koller and M. Sahami, "Hierarchically Classifying Documents Using Very Few Words," *International Conference on Machine Learning*, pp. 170–178, 1997.

[18] S. D'Alessio, K. Murray, R. Schiaffino, and A. Kershenbaum, "The Effect of Using Hierarchical Classifiers in Text Categorization." *Riao*, 2000.

[19] P. Hao, J. Chiang, and Y. Tu, "Hierarchically SVM classification based on support vector clustering method and its application to document categorization," *Expert Systems with Applications*, vol. 33, pp. 627–635, 2007.

[20] K. Wang, "Building Hierarchical Classifiers Using Class Proximity," *Very Large Data Bases*, pp. 363–374, 1999.

[21] L. Cai and T. Hofmann, "Hierarchical document categorization with support vector machines," *Conference on Information and knowledge management*, p. 78, 2004.

[22] X. Qiu, W. Gao, and X. Huang, "Hierarchical Multi-Class Text Categorization with Global Margin Maximization," *Association for Computational Linguistics and International Joint Conference of the Asian Federation of Natural Language Processing*, 2009.

[23] L. Tang, H. Liu, J. Zhang, N. Agarwal, and J. J. Salerno, "Topic taxonomy adaptation for group profiling," *ACM Transactions on Knowledge Discovery from Data*, vol. 1, no. 4, pp. 1–28, 2008.

[24] K. Nitta, "Improving taxonomies for large-scale hierarchical classifiers of web documents," *Proceedings of the 19th ACM international conference on Information and knowledge management - CIKM '10*, p. 1649, 2010.

[25] T. Li, S. Zhu, and M. Ogihara, "Topic Hierarchy Generation via Linear Discriminant Projection," *Sigir*, pp. 421–422, 2003.

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

[27] J. Weston, A. Makadia, and H. Yee, "Label partitioning for sublinear ranking," *International Conference on Machine Learning*, vol. 28, 2013.

[28] B. Liu, F. Sadeghi, M. Tappen, O. Shamir, and C. Liu, "Probabilistic label trees for efficient large scale image classification," *Computer Vision and Pattern Recognition*, pp. 843–850, 2013.

[29] T. Gao and D. Koller, "Discriminative learning of relaxed hierarchy for large-scale visual recognition," *International Conference on Computer Vision*, 2011.

[30] S. Liu, H. Yi, L.-t. Chia, and D. Rajan, "Adaptive Hierarchical Multi-class SVM Classifier for Texture-based Image Classification," *International Conference on Multimedia and Expo*, pp. 3–6, 2005.

[31] L. Zhigang, S. Wenzhong, Q. Qianqing, L. Xiaowen, and X. Donghui, "Hierarchical support vector machines," *Geoscience and Remote Sensing Symposium IGARSS '05*, 2005.

[32] A. Beygelzimer, J. Langford, Y. Lifshits, G. Sorkin, and A. Strehl, "Conditional Probability Tree Estimation Analysis and Algorithms," *Uncertainty in Artificial Intelligence*, 2009.

[33] A. Choromanska and J. Langford, "Logarithmic Time Online Multiclass prediction," pp. 1–13, 2014.

[34] A. McCallum and K. Nigam, "A Comparison of Event Models for Naive Bayes Text Classification," *AAAI/ICML-98 Workshop on Learning for Text Categorization*, pp. 41–48, 1998.

[35] I. Partalas, A. Kosmopoulos, N. Baskiotis, T. Artieres, G. Paliouras, E. Gaussier, I. Androutsopoulos, M.-r. Amini, and P. Galinari, "LSHTC : A Benchmark for Large-Scale Text Classification," pp. 1–9, 2015.

[36] E. Allwein, R. Schapire, and Y. Singer, "Reducing multiclass to binary: A unifying approach for margin classifiers," *The Journal of Machine Learning*, vol. 1, pp. 113–141, 2001.

[37] K. Dellschaft and S. Staab, "On How to Perform a Gold Standard Based Evaluation of Ontology Learning," *International Semantic Web Conference*, 2006.