ISSN 2395-8618

IN-DEDUCTIVE and DAG-Tree Approaches for Large-Scale Extreme Multi-label Hierarchical Text Classification

Mohammad Golam Sohrab, Makoto Miwa, and Yutaka Sasaki

***Abstract*—This paper presents a large-scale extreme multi-label hierarchical text classification method that employs a large-scale hierarchical inductive learning and deductive classification (IN-DEDUCTIVE) approach using different efficient classifiers, and a DAG-Tree that refines the given hierarchy by eliminating nodes and edges to generate a new hierarchy. We evaluate our method on the standard hierarchical text classification datasets prepared for the PASCAL Challenge on Large-Scale Hierarchical Text Classification (LSHTC). We compare several classification algorithms on LSHTC including DCD-SVM, SVM^{perf}, Pegasos, SGD-SVM, and Passive Aggressive, etc. Experimental results show that IN-DEDUCTIVE approach based systems with DCD-SVM, SGD-SVM, and Pegasos are promising and outperformed other learners as well as the top systems participated in the LSHTC3 challenge on Wikipedia medium dataset. Furthermore, DAG-Tree based hierarchy is effective especially for very large datasets since DAG-Tree exponentially reduce the amount of computation necessary for classification. Our system with IN-DEDUCIVE and DAG-Tree approaches outperformed the top systems participated in the LSHTC4 challenge on Wikipedia large dataset.**

***Index Terms*—Hierarchical text classification, multi-label learning, indexing, extreme classification, tree-structured class hierarchy, DAG- or DG-structured class hierarchy.**

I. Introduction

STATISTICAL Natural Language Processing (NLP) is now facing various "Big Data" challenges. In machine learning (ML)-based text classification (TC), the current front-line of "Big Data" deals with millions of training and test documents as well as hundreds of thousands, or even millions of labels. Although strong ML methods such as Support Vector Machines (SVMs) [1], [2], [3] have been successfully applied to TC, such large-scale datasets are often handled with a light-weight classifiers, such as k-nearest neighbors [4], or by information retrieval-based approaches [5].

In general, ML-based TC can be categorized into two classification tasks: a flat classification (FC) by referring to standard binary or multi-class classification problems and a hierarchical classification (HC)– typically a tree, a directed acyclic graph (DAG), or a directed graph (DG) is incorporated, where the classes to be predicted are organized into a class hierarchy. A very large amount of research in TC, data mining (DM), and related researches have focused on FC problems. In contrast, many important real-world classification problems are naturally cast as HC problems. In large-scale hierarchical text classification (LSHTC) tasks, the size of data is too large to analyze the suitable classifiers. Therefore, it is still an open and more challenging problem to design a model that classifies large-scale documents into large-scale hierarchically-structured categories that correspond to classes accurately and efficiently. The benefit of hierarchical text classification (HTC) approach is the efficiency. In the training stage, deeper in the hierarchy the category is located, less data need to be handled by classifiers on average. Because of this, total training time can be drastically reduced. In the classification stage, the complexity to decide a category as the assignment for a sample will be $O(\log n)$ with n leaf categories.

In this direction, this paper present an approach that refines the DG and generates a DAG-Tree hierarchy where DAG and tree are incorporated together, in order not only to drastically reduce training and test time but also to significantly improve the classification performance; especially when the hierarchy is too large and the training data for each class is sparse. We built an accurate and efficient LSHTC system and applied it to Wikipedia medium dataset (WMD) and Wikipedia large dataset (WLD), which treat a typical multi-label TC problem to automatically assigning Wikipedia hierarchical categories to a document. These tasks of assigning Wikipedia categories to documents are included in the third (LSHTC3[1]) and forth (LSHTC4[2]) editions of PASCAL challenge.

The remainder of the study is organized as follows. Section 2 describes the base ML algorithms. In Section 3, we elaborate the proposed inductive learning and deductive classification approaches for hierarchical learning. Section 4 presents a hierarchy refinement approach for very large-scale hierarchical classification. Section 5 shows the experiment results with discussions. Finally, we conclude the study in Section 6.

Manuscript received on February 09, 2016, accepted for publication on June 16, 2016, published on October 30, 2016.

Mohammad Golam Sohrab (corresponding author), Makoto Miwa, and Yutaka Sasaki are with the Faculty of Engineering, Toyota Technological Institute, 2-12-1 Hisakata Tempaku-ku Nagoya 468-8511, Japan (e-mails: {sohrab, makoto-miwa, yutaka.sasaki}@toyota-ti.ac.jp).

[1]http://lshtc.iit.demokritos.gr/LSHTC3_CALL
[2]http://lshtc.iit.demokritos.gr/

ISSN 2395-8618

II. Base ML Algorithms

The section briefly describes the efficient base learners: SGD-SVM, Pegasos, and DCD-SVM.

A. SGD-SVM

Stochastic Gradient Descent SVM (SGD-SVM) [6] is an incremental training algorithm for SVMs. It randomly selects a sample and adjusts the weight vector. Given a loss function $\ell(\vec{w};(x,y))$, SGD-SVM solves the following primary SVM optimization problem directly:

$$\min_{w} \frac{\lambda}{2}\|\vec{w}\| + \frac{1}{N}\sum_{(\vec{x},y)\in D} \ell(\vec{w};(\vec{x},y)), \tag{1}$$

where $\vec{w}$ is an weight vector, D is a set of N pairs of samples and their labels, and λ is a regularization parameter. The weight vector at a time step t is updated as:

$$\vec{w}_{t+1} = \vec{w}_t - \eta_t S^{-1}(\lambda\vec{w}_t + \ell'(\vec{w}_t;(\vec{x}_{t+1},y_{t+1}))\vec{x}_{t+1}) \tag{2}$$

where S is a symmetric positive definite matrix, regarded as a pre-conditioner.

B. Pegasos

Primal Estimated sub GrAdient SOlver for SVM (Pegasos) [7] is an efficient training algorithm for SVMs. Pegasos alternates stochastic gradient decent steps and projection steps. In the projection step, the current weight vector is re-scaled to fit the L_2-ball of radius $1/\sqrt{\lambda}$. For a randomly chosen sample $(\vec{x}_t, y_t)$ and a weight vector $\vec{w}_t$ at a time step t, if $y_t\vec{w}_t\cdot\vec{x}_t < 1$, Pegasos updates the weight vector as follows:

$$\vec{w}_{t+\frac{1}{2}} = (1-\eta_t\lambda)\vec{w}_t + \eta_t y_t \vec{x}_t. \tag{3}$$

$$\vec{w}_{t+1} = min\left(1, \frac{\frac{1}{\lambda}}{\|\vec{w}_{t+\frac{1}{2}}\|}\right)\vec{w}_{t+\frac{1}{2}}. \tag{4}$$

C. DCD-SVM

Dual coordinate decent support vector machine (DCD-SVM) [8] randomly selects a weight vector $\vec{w}$ and updates the weight vector as:

$$\vec{w} \leftarrow \vec{w} + (\alpha_i - \alpha_i')y_i\vec{x}_i, \tag{5}$$

where α_i' is the current value and α_i is the target value. The optimization process starts from an initial point $\vec{\alpha} \in \mathbb{R}^l$ and generates a sequence of vectors $\{\vec{\alpha}^k\}_k^\infty$. We refer to the process from $\vec{\alpha}^k$ to $\vec{\alpha}^{k+1}$ as an outer iteration. In each outer iteration, we have l inner iterations, so that sequentially updates $\alpha_1, \alpha_2, ..., \alpha_l$. In updating $\vec{\alpha}^{k,i}$ to $\vec{\alpha}^{k,i+1}$, the process must find the optimal solution as:

$$\alpha_i^{k,i+1} = min\left(max\left(\alpha_i^{k,i} - \frac{\nabla_i f\left(\vec{\alpha}^{k,i}\right)}{\vec{x}_i^T\cdot\vec{x}_i}, 0\right), C\right), \tag{6}$$

where $C > 0$ is a penalty parameter and set to 0.5 based on our pilot study. $\nabla_i f$ is the i-th component of the gradient ∇f, and $\nabla_i f\left(\vec{\alpha}^{k,i}\right)$ is set as:

$$\nabla_i f\left(\vec{\alpha}\right) = y_i\vec{w}^T\cdot\vec{x}_i - 1. \tag{7}$$

The process move to index $i+1$ with updating $\alpha_i^{k,i}$, if and only if the projected gradient $\nabla_i^P f\left(\vec{\alpha}^{k,i}\right) \neq 0$ and satisfy the following conditions,

$$\nabla_i^P f\left(\vec{\alpha}\right) = \begin{cases} \nabla_i f\left(\vec{\alpha}\right) & \text{if } 0 < \alpha_i < C, \\ \min\left(0, \nabla_i f\left(\vec{\alpha}\right)\right) & \text{if } \alpha_i = 0, \\ \max\left(0, \nabla_i f\left(\vec{\alpha}\right)\right) & \text{if } \alpha_i = C. \end{cases} \tag{8}$$

III. IN-DEDUCTIVE Approach

IN-DEDUCTIVE (inductive learning and deductive classification) is a hierarchical learning and classification approach for classifying large-scale documents into a large-scale hierarchically structured categories (category hierarchy) accurately and efficiently. The IN-DEDUCTIVE approach follows the bottom-up propagation with edge-based training, top-down classification approach with global adjustments, and global pruning.

A. Inductive Learning on Category Hierarchy

The inductive learning induces a set of observed instances or samples from specific bottom categories to general top categories in the category hierarchy.

1) Bottom-up Propagation: Since only leaf categories are assigned to data, first we propagate training samples from the leaf level to the root in the category hierarchy. Fig. 1 illustrates propagation of documents in a hierarchy consisting of six categories A-F. In this figure, sample x_1 is assigned to categories D and E, x_2 to D, and x_3 to F. Let us look at the case of x_1 assigned to E. x_1 of E is propagated to both categories B and C. Then, x_1 of B is propagated to A. When x_1 is propagated from C to A afterwards, to avoid redundant propagation, the propagation of x_1 (originally from E via C) terminates at A, even if A had a parent category. To perform the propagation of the sample, we employ a recursive algorithm. Steps 1-13 for bottom-up propagation in Algorithm 1 are described in pseudo-code. Samples are propagated in a DAG in the bottom-up manner.

2) Edge-based Training: Based on the propagation of training samples in the hierarchy, we train classifiers for each edge of the hierarchy to estimate how strong the samples of the parent node are related to the child nodes. Each edge is coupled with a binary classifier using the one-against-the-rest approach. In Fig. 2 at node B, x_1 and x_2 are assigned to node B during the bottom-up propagation. Since edge-based learning is in concern, the model M_{BD} is trained in the hierarchy as to classify both x_1 and x_2 to D; whereas the model M_{BE} is trained as to classify x_1 to E but not x_2 to E. M_{BD} is trained without negatives. Training models on local branches is beneficial in restricting the number of samples. It is also

ISSN 2395-8618

effective to reduce positive-negative data imbalance. Another benefit of this edge-oriented classification is that a classifier can capture local characteristics of data distribution. Naturally, x_1 classified into E from node B and x_1 from node C should need different considerations, since the former is classification based on x_1 and x_2 and the later is based on x_1 and x_3, and thus M_{BE} and M_{CE} will be different models. Edge-based training is described in the steps 14-34 in Algorithm 1.

B. Deductive Classification on Category Hierarchy

The deductive classification deduces a set of unlabeled samples from general top categories to more specific bottom categories in the hierarchy.

1) Efficient Top-down Classification: Fig. 3 illustrates top-down classification of a test sample $\vec{x}$. First, $\vec{x}$ is classified to B and C, based on the decision by $M_{AB}(\vec{x})$ and $M_{AC}(\vec{x})$, respectively. The decision is made by:

$$G_{pc}(\vec{x}) = \vec{w}_{pc} \cdot \vec{x} + b_{pc}. \tag{9}$$

To adjust the effect of positive-negative sample imbalance, we set a bias β. When $G_{pc}(\vec{x}) > \beta$, $\vec{x}$ is classified from parent category p to child category c. When both $G_{AB}(\vec{x}) > \beta$ and $G_{AC}(\vec{x}) > \beta$ are satisfied, $\vec{x}$ is classified into both B and C. Note that the standard bias term b_{pc} is automatically tuned for each edge in the training stage. Usually, the dot product can be calculated efficiently with the dot product of sparse vectors [9]. For the first calculation of the dot product in the classification stage, the dot product can be calculated as follows:

$$\vec{w} \cdot \vec{x} = \sum_{i:1..size[x_{index}]} w[x_{index}[i]] * x_{value}[i], \tag{10}$$

where w is an weight vector array and x_{index} and x_{value} represent a sparse vector of sample x. Top-down classification is described in the steps 35-53 Algorithm 1. The classification procedure is initiated with TOP-DOWN-CLASSIFY(x,root,1).

C. Global Pruning

After the classification of a test sample x throughout the hierarchy, we prune unlikely classes for x. We define a confidence score and set the global threshold θ for it. When x reaches a leaf node n, the confidence score $c_\alpha(x, n)$ is calculated as follows:

$$c_\alpha(x, n) = \prod_{(n_1,n_2)\in E} \sigma_\alpha(G_{n_1 n_2}(x))), \tag{11}$$

where E is a set of edges that x has followed in the path from the root to the leaf n. The output value of a classifier is converted to $[0, 1]$ range by:

$$\sigma_\alpha(x) = \frac{1}{1 + \exp(-\alpha x)}, \tag{12}$$

α is set to 2 from our pilot study. When there are multiple nodes assigned to x, if $c(x, n) < \theta$, the assignment of x to n is removed. Fig. 4 illustrates the global pruning. In Fig. 4, x is classified into D and F. Here the confidence scores of x in D and F is 0.21 and 0.09 respectively. When θ = 0.3, both of the confidence scores are below the threshold. However, we need at least one category for every test sample. Therefore, only assignment of x to F is removed.

IV. Hierarchy Refinement: DAG-Tree

Since IN-DEDUCTIVE is a edge-oriented approach that increases the computational cost of training and test by building hundreds of thousands, or even millions of classification models. To reduce the computational cost in training and test as well as to improve the classification performances, we introduce a hierarchy refinement approach and generate a DAG- and tree-based hierarchy; especially from very large directed graph which contains cycles and multiples roots in the hierarchy.

DAG-Tree incorporates DAG and tree substructures together into a same graph to refine the given hierarchy. In the DAG-Tree approach, lead nodes contain multiple parents where from intermediate nodes to root nodes in the hierarchy contain single parent in the hierarchy. To generate the DAG-Tree hierarchy, we split the given hierarchy into three indexing system. In the LEAF-INDEX, all the leaf categories (leaf contains only parent) in the hierarchy are associated. In the INTER-INDEX, all the intermediate categories (Intermediate nodes contain both parent and child) in the hierarchy are associated. Finally, in the ROOT-INDEX, root categories (Root contains only children) in the hierarchy are associated. From leaf level to intermediate level in the DAG-Tree hierarchy, a leaf node can have more than one parents. From intermediate levels to top level, a node contains only one parent in the DAG-Tree hierarchy.

To handle large data using hierarchy, we introduce a bottom-up based DAG-Tree. The primary motivation of creating DAG-Tree is to remove all cycles by ignoring nodes and edges that have been already visited in the paths from leaf to root in the intermediate levels. Any parent-child relations that would lead to cycle are omitted. Fig. 6 shows an example DAG-Tree categories hierarchy.

DAG-Tree is a visited-paths based approach. In the visited-paths, DAG-Tree keeps record all the parents list of a certain leaf or intermediate node. If any parent of a certain leaf or intermediate node appears in the visited-paths list, DAG-Tree immediately stores the last parent record by terminating the process and accesses the next parent of a certain leaf in the hierarchy. In Figures 5 and 6, the bottom and top gray circles indicate the leaf and root nodes respectively; as well as circles in the dotted rectangle indicate the intermediate nodes. First we generate a leaf to root paths using bottom-up manner and invoke the paths as a reference-path for every parents of a certain leaf to reach root node. The procedures in Fig. 7 show a DAG-Tree generation from the hierarchy in Fig. 5.

Algorithm 2 shows the visited-paths based DAG-Tree generation algorithm. Compare to Fig. 5, Fig. 6 shows many nodes and edges, including intermediate node G and root node

ISSN 2395-8618

Algorithm 1. IN–DEDUCTIVE Approach for LSHTC

procedure BOTTOM-UP-PROPAGATE(NODE, SAMPLES)
  if sample is already assigned to node **then**
    return;
  else
    Assign sample to node;
  end if
  if node is the root **then**
    return;
  end if
  for all p in the parents of node **do**
    BOTTOM-UP-PROPAGATE(p, sample)
  end for
end procedure
procedure TRAIN(NODE)
  if sample is already explored **then**
    return;
  end if
  if node is a leaf **then**
    return;
  end if
  Let X be the set of samples assigned to node;
  Let Y be the set of labels for X
  for all n in the children of node **do**
    for all x_i in X **do**
      if x_i is assigned to n **then**
        $y_i \leftarrow +1$;
      else
        $y_i \leftarrow -1$;
      end if
    end for
    Train classification model $M_{node,n}$ on (X,Y)
    TRAIN(n)
  end for
end procedure
procedure TOP-DOWN-CLASSIFY(x, $node$, $conf$)
  if $node$ is a leaf node **then**
    assign x to node;
  end if
  if node is a leaf **then**
    c_α(x,$node$) $\leftarrow$ max($conf$,old(c_α(x,$node$));
  else
    c_α(x,$node$) $\leftarrow$ $conf$;
  end if
  for all n in the children of $node$ **do**
    if $G_{node,n} > \beta$ **then**
      TOP-DOWN-CLASSIFY($x, n, conf * \sigma(G_{node,n}(x))$);
    end if
  end for
  if None of child nodes satisfies $M_{node}, n(x) > \beta$ **then**
    n' = $\underset{n}{\text{argmax}} G_{node,n}(x)$;
    TOP-DOWN-CLASSIFY($x, n', conf * \sigma(G_{node,n'}(x))$);
  end if
end procedure

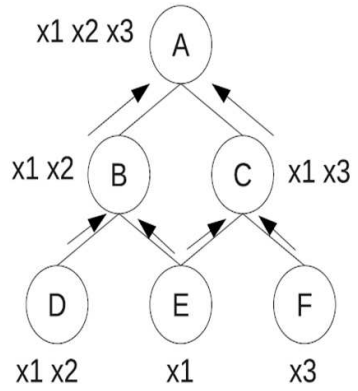

Fig. 1. Bottom–up propagation of training data.

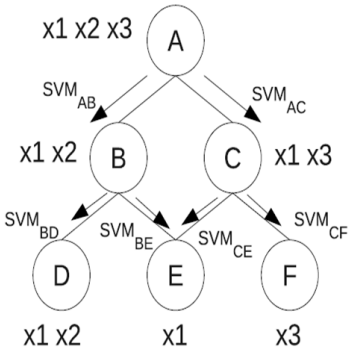

Fig. 2. Edge–based training.

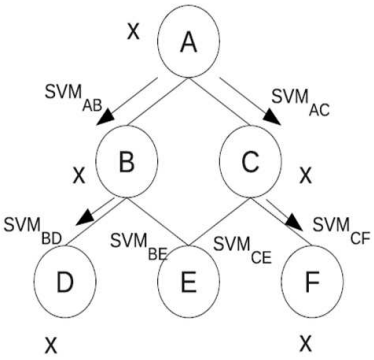

Fig. 3. Top–down classification.

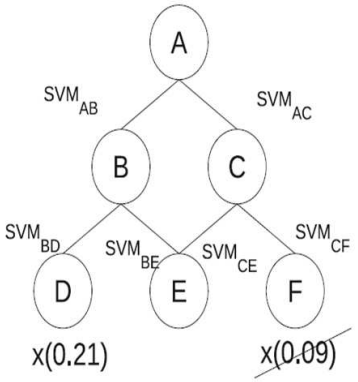

Fig. 4. Global pruning.

ISSN 2395-8618

C, are eliminated during the DAG-Tree generation process. We then extend the intermediate nodes by including the root nodes and connect all the root nodes to a single root Z.

V. Experimental Settings

In this section, we provide empirical evidence for the effectiveness of our proposed IN-DEDUCTIVE and DAG-Tree approaches.

A. Base ML Algorithms

We employ sofia-ml[3] for the experiments with Pagasos, SGD-SVM, Passive Aggressive (PA) [10], Relaxed Online Margin Algorithm (ROMMA) [11], and Logistic regression with Pegasos projection (logreg-pegasos). For Pegasos and SGD-SVM, we use the SVM regularization parameter C where $\lambda = \frac{1}{CN}$ and the number of iteration is set to $\max(10000, 100N)$. Moreover, SVM^{perf} [12], [13] is adopted, and the SVM parameters C for SVM^{perf} is given as $C^{perf} = NC/100$, where N is the number of samples of each edge. Note that C^{perf} varies in each node during the top-down training since C^{perf} depends on N.

B. Term Weighting Approaches

We compare several term weighting methods: term frequency (TF), TF.IDF, and four class-oriented indexing-based weighting methods.

TF is a very simple, conventional, and baseline term weighting method in the information retrieval task and it is defined as:

$$W_{TF}(t_i, d) = tf_{(t_i,d)}, \tag{13}$$

where $tf(t_i, d)$ is the number of occurrences of term t_i in document d.

The common document-indexing-based TF.IDF is defined as:

$$W_{TF.IDF}(t_i, d) = tf_{(t_i,d)} \times \left(1 + \log \frac{D}{\#(t_i)}\right), \tag{14}$$

where D denotes the total number of documents in the training corpus, $\#(t_i)$ is the number of documents in the training corpus in which term t_i occurs at least once, $D/\#(t_i)$ is the inverse document frequency (IDF) of term t_i.

As for the class-oriented indexing-based methods [14], [15], we employed four following methods defined as:

$$W_{TF.ICF}(t_i, d, c_k) = tf_{(t_i,d)} \times \left(1 + \log \frac{C}{c(t_i)}\right), \tag{15}$$

$$W_{TF.ICS_\delta F}(t_i, d, c_k) = tf_{(t_i,d)} \times \left(1 + \log \frac{C}{CS_\delta(t_i)}\right), \tag{16}$$

$$W_{TF.IDF.ICF}(t_i, d, c_k) = tf_{(t_i,d)} \times \left(1 + \log \frac{D}{\#(t_i)}\right) \times \left(1 + \log \frac{C}{c(t_i)}\right), and \tag{17}$$

$$W_{TF.IDF.ICS_\delta F}(t_i, d, c_k) = tf_{(t_i,d)} \times \left(1 + \log \frac{D}{\#(t_i)}\right) \times \left(1 + \log \frac{C}{CS_\delta(t_i)}\right), \tag{18}$$

where C denotes the total number of predefined categories in the training corpus, $c(t_i)$ is the number of categories in the training corpus in which term t_i occurs at least once, $\frac{C}{c(t_i)}$ is the ICF of the term t_i, and $\frac{C}{CS_\delta(t_i)}$ is the (ICS$_\delta$F) of term t_i. Please refer to [14], [16] for more details.

C. Datasets

To evaluate the performance of our proposed IN-DEDUCTIVE and DAG-Tree approaches, we compare our results with WMD and WLD which considering two standard datasets for LSHTC.

1) Wikipedia Medium Dataset: The Dataset[4] consists of 456,866 training documents with 346,299 distinct features and 81,262 test documents with 132,296 distinct features. It contains 36,504 leaf categories and 50,312 categories in the hierarchy with maximum depth 12. The number of edges in the hierarchy are 65,333. The category hierarchies of WMD is in the form of DAG.

2) Wikipedia Large Dataset: The Dataset[5] consists of 2,365,436 training data with 1,617,899 distinct features and 452,167 test data with 627,935 distinct features. It contains 325,055 leaf categories and 478,020 categories in the hierarchy with maximum depth 15. The number of edges in the hierarchy are 863,261. The category hierarchies of WLD are in the form of DG.

D. Performance Measures

We employ official LSHTC3 and LSHTC4 evaluation metrics [17]. Given documents D, correct labels Y_i, and predicted labels Z_i, the metrics are as follows:

- Accuracy(Acc): $1/|D| \sum_{i \in D} |Y_i \cap Z_i| / (|Y_i \cup Z_i|)$
- Example-based F1 measure (EBF): $1/|D| \sum_{i \in D} 2|Y_i \cap Z_i| / (|Y_i| + |Z_i|)$
- Label-based Macro-average F1 measure (LBMaF): Standard multi-label Macro-F1 score
- Label-based Micro-average F1 measure (LBMiF): Standard multi-label Micro-F1 score
- Hierarchical F1 measure (HF): The example-based F1-measure counting ancestors of true and predicted categories

We evaluated our systems on LSHTC evaluation site[6] because the gold standard labels for the test data of WMD and WLD are not publicly available.

[3] http://code.google.com/p/sofia-ml/

[4] http://lshtc.iit.demokritos.gr/LSHTC3_DATASETS

[5] http://lshtc.iit.demokritos.gr/LSHTC4_GUIDELINES

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

ISSN 2395-8618

Algorithm 2. Bottom–up DG to DAG-Tree Generation

```
1: procedure DAG–TREE(LEAF-INDEX, INTER-INDEX, ROOT-INDEX)
2:     while LEAF-INDEX.leaf is not empty do
3:         parentLeaf ← leaf.parentList
4:         LEAF–ROOT(parentLeaf)
5:     end while
6:     return path
7: end procedure
8: procedure LEAF–ROOT(PINDEX)
9:     while pIndex.curParent is not empty do
10:        if curParent is in ROOT-INDEX then
11:            VISITED–PATH(leaf, curParent)
12:        else
13:            childInter ← curParent
14:            INTER–NODES(childInter)
15:        end if
16:    end while
17: end procedure
18: procedure VISITED–PATH(leaf, p)
19:    path.leaf.pathList ← p
20:    if all parents of a leaf are explored then
21:        goto step2: for next leaf
22:    else
23:        explore parent of LEAF- or INTER-INDEX
24:    end if
25: end procedure
26: procedure INTER–NODES(inter)
27:    while INTER-INDEX.inter is not empty do
28:        parentInter ← inter.parentList
29:        LEAF–ROOT(parentInter)
30:    end while
31: end procedure
```

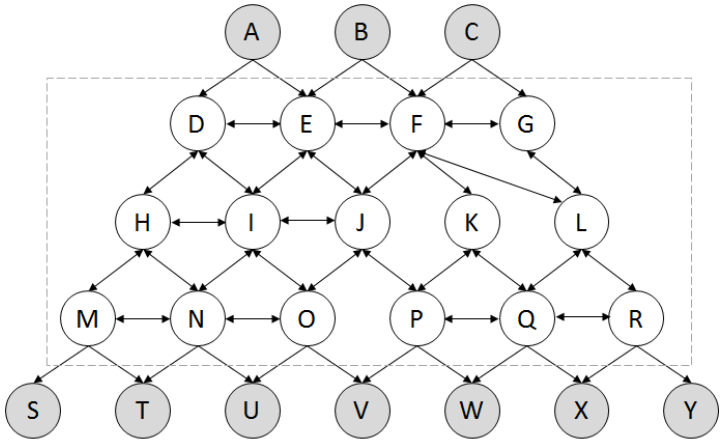

Fig. 5. Directed Graph (DG).

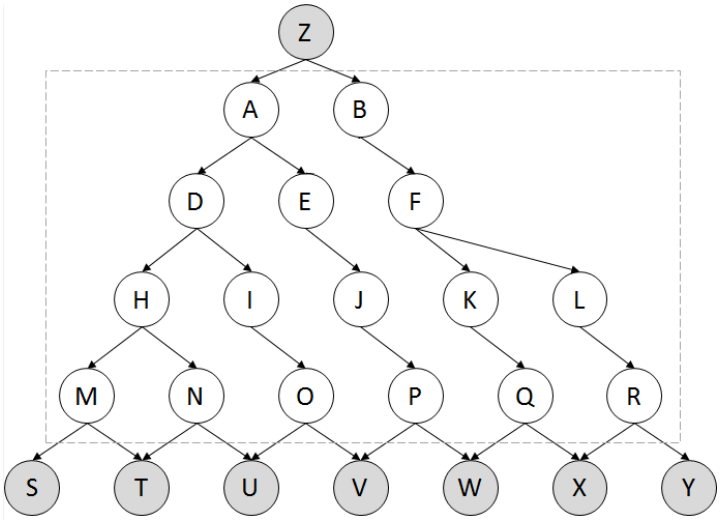

Fig. 6. DAG-Tree.

E. Experimental Environments

We assessed the training and classification time using a single Xeon 3.0GHz core with 96GB memory for WMD and 396GB memory for WLD. The feature values in the LSHTC3 and LSHTC4 datasets represent the number of occurrences of each unigram. From Eqns. 13–18, we scaled the feature value with the function $v/(v+1)$ where v is the original feature value.

VI. Results and Discussion

In this section, we provide empirical evidence of the INDEDUCTIVE approach and the effectiveness of the DAG-Tree approach in very large-scale datasets.

A. DG to DAG-Tree using Wikipedia Large Dataset

The hierarchy of WLD is a directed graph (DG), where a node can have more than one parents. It contains cycles, where a set of nodes are connected by edges and the cycles only appear in the intermediate or hidden nodes. Fig. 5 shows an example of DG, where the double arrow between two nodes, is the parent of one another. In the WLD, 10,246 cyclic nodes appear in the intermediate levels with a maximum depth of 13. It contains multiple roots with 11,405 in the hierarchy.

Using this DAG-Tree approach, we refine the hierarchy by reducing the edges, intermediate nodes, root nodes, and cyclic nodes from 863,261 to 683,359, 141,559 to 140,703, 11,405 to 10,902, and 10,246 to 0 respectively.

B. LSHTC Evaluation

Table I shows the results with Pegasos on WMD. We showed the results with $\beta \in (0.0, -0.5)$ and varied $\theta \in (0.00, 0.25, 0.27, 0.32)$. $\beta = -0.5$ means that data classified into negative side to some extent are passed to the child node. This means that some incorrect assignments are kept in the candidate sets. However, most of the incorrect classification are removed after-ward during the global pruning stage. SVM hyper-parameter C has been set to 0.5 based on our pilot study. Table II shows the scores of different weighting approaches with DCD-SVM. As for DCD-SVM results on WMD in

ISSN 2395-8618

1: S→M→H→D→A ▷ First Reference Visited-Paths from leaf to root
2: T→M ▷ M parent of T, is in the Visited-paths list in step 1:
3: T→N→H ▷ H parent of N, is in the Visited-paths list in step 1:
4: U→N ▷ N parent of U, is in the Visited-paths list in step 3:
5: U→O→I→D ▷ D parent of I, is in the Visited-paths list in step 1:
6: V→O ▷ O parent of V, is in the Visited-paths list in step 5:
7: V→P→J→E→A ▷ A parent of E, is in the Visited-paths list in step 1:
7: W→P ▷ P parent of W, is in the Visited-paths list in step 7:
8: W→Q→K→F→B ▷ B parent of F, is a root
9: X→Q ▷ Q parent of X, is in the Visited-paths list in step 8:
10: X→R→L→F ▷ F parent of L, is in the Visited-paths list in step 8:
11: Y→R ▷ R parent of Y, is in the Visited-paths list in step 10:

Fig. 7. Procedures to generate DAG-Tree hierarchy with a visited-paths list

TABLE I
EXPERIMENTAL RESULTS WITH PEGASOS ON WMD

C	α	β	θ	Acc	EBF	LBMaF	LBMiF	HF
0.5	2	0.0	0.00	0.3955	0.4585	0.2753	0.4393	0.6633
0.5	2	0.0	0.25	0.4334	0.4840	0.2666	0.4870	0.7015
0.5	2	0.0	0.27	0.4335	0.4834	0.2633	0.4870	0.7028
0.5	2	0.0	0.32	0.4328	0.4816	0.2552	0.4853	0.7014
0.5	2	–0.5	0.00	0.2966	0.3749	0.2542	0.2772	0.5469
0.5	2	–0.5	0.25	0.4388	**0.4958**	**0.2832**	0.4951	0.7058
0.5	2	–0.5	0.27	0.4406	**0.4958**	0.2773	0.4966	0.7069
0.5	2	–0.5	0.32	**0.4423**	0.4948	0.2669	0.4966	**0.7076**

TABLE II
DIFFERENT WEIGHTING APPROACHES WITH DCD-SVM ON WMD

Weighting Approach	β	θ	Acc	EBF	LBMaF	LBMiF	HF
IN-DEDUCTIVE + TF	–0.5	0.39	**0.4452**	**0.4968**	**0.2664**	**0.4978**	**0.7086**
IN-DEDUCTIVE + TF.IDF	–0.5	0.42	0.4284	0.4764	0.2537	0.4800	0.6943
IN-DEDUCTIVE + TF.ICF	–0.5	0.42	0.4346	0.4831	0.2592	0.4863	0.6984
IN-DEDUCTIVE + TF.IDF.ICF	–0.5	0.42	0.4219	0.4695	0.2481	0.4733	0.6900
IN-DEDUCTIVE + TF.ICS$_\delta$F	–0.5	0.42	0.4297	0.4779	0.2544	0.4812	0.6953
IN-DEDUCTIVE + TF.IDF.ICS$_\delta$F	–0.5	0.42	0.4221	0.4697	0.2481	0.4735	0.6899

Table III, when β = -0.5 and θ = 0.39, we obtained the best accuracy 0.4452. Since the local weight TF in Table II outperformed other weighting approaches, only this weighting approach is taken into account for WLD. In addition, the parameter β = -0.5 performed consistently better in different learning algorithms, thus β = -0.5 is taken into account for the WLD. Table V shows the results with Pegasos on the WLD, where we obtained the best result 0.3496 using our system with DAG-Tree hierarchy. The result shows that our system using DAG-Tree hierarchy outperformed with the given hierarchy.

Tables IV and VI, summarizes our results with compare to the top four systems using WMD and WLD respectively. The result shows that the IN-DEDUCTIVE approach based system outperformed the other systems participated in the LSHTC3 challenge as well as in LSHTC4 challenge. Table VII illustrates the training and test time spent for the WMD and WLD. In Table II, Table III, Table V, Table VI, and Table VII we use $C = 0.5$ and $\alpha = 2$.

C. Discussion

Ioannou [20] summarizes thresholding methods in multi-label classification. Basically bias β is a threshold that adjusts PCut [21]. Note that the training phase automatically set bias b of the decision function $G_{pc}(x) = \vec{w}_{pc}\vec{x} + b_{pc}$. Setting β means the classification threshold adjustment, $i.e., G_{pc}(x) = \vec{w}_{pc}\vec{x} + b_{pc} > \beta$, where p and c are parent and child categories, respectively. It is noticeable in Table II that the local term weighting approach TF outperformed other weighting approaches that incorporate global weightings into local weights for LSHTC.

ISSN 2395-8618

TABLE III
COMPARISON OF EFFICIENT ML METHODS ON WMD

Learning Algorithm	β	θ	Acc	EBF	LBMaF	LBMiF	HF
IN-DEDUCTIVE + DCD-SVM	–0.5	0.39	**0.4452**	**0.4968**	0.2664	**0.4978**	**0.7086**
IN-DEDUCTIVE + Pegasos	–0.5	0.32	0.4423	0.4948	**0.2669**	0.4966	0.7076
IN-DEDUCTIVE + SGD-SVM	–0.5	0.32	0.4419	0.4938	0.2641	0.4957	0.7072
IN-DEDUCTIVE + SVM^{perf}	–0.5	0.32	0.4405	0.4919	0.2623	0.4947	0.7071
IN-DEDUCTIVE + PA	–0.5	0.49	0.4005	0.4512	0.2550	0.4527	0.6673
IN-DEDUCTIVE + ROMMA	–0.5	0.15	0.3827	0.4324	0.2296	0.4362	0.5610
IN-DEDUCTIVE + logreg	–0.3	0.14	0.3690	0.4235	0.1544	0.4271	0.6688
IN-DEDUCTIVE + logreg-pegasos	–0.5	0.14	0.3689	0.4255	0.1644	0.4296	0.6682

TABLE IV
EXPERIMENTAL RESULTS WITH PEGASOS ON WLD

Name	C	α	β	θ	Acc	EBF	LBMaF	LBMiF	HF
IN-DEDUCTIVE + DG	0.5	2	−0.5	0.37	0.3183	0.3861	0.1918	0.3641	0.4291
IN-DEDUCTIVE + DAG-Tree	0.5	2	−0.5	0.38	0.3495	**0.4239**	**0.1968**	0.3946	**0.4790**
	0.5	2	−0.5	0.39	**0.3496**	0.4235	0.1937	0.3951	0.4773
	0.5	2	−0.5	0.40	0.3494	0.4229	0.1907	**0.3952**	0.4751

TABLE V
COMPARISON WITH TOP FOUR LSHTC3 PARTICIPANTS ON WMD

Name	Acc	EBF	LBMaF	LBMiF	HF
IN-DEDUCTIVE + DCD-SVM	**0.4452**	**0.4968**	0.2664	**0.4978**	**0.7086**
IN-DEDUCTIVE + Pegasos	0.4423	0.4948	0.2669	0.4966	0.7076
IN-DEDUCTIVE + SGD-SVM	0.4419	0.4938	0.2641	0.4957	0.7072
IN-DEDUCTIVE + SVM^{perf}	0.4405	0.4919	0.2623	0.4947	0.7071
arthur (Ist)	0.4382	0.4937	0.2674	0.4939	0.7092
coolvegpuff (2nd)	0.4291	0.4824	0.2507	0.4779	0.6892
TTI (3rd)	0.4200	0.4771	**0.2835**	0.4725	0.6922
chrishan (4th)	0.4117	0.4768	0.2454	0.4187	0.6766

TABLE VI
COMPARISON WITH TOP FOUR LSHTC4 PARTICIPANTS ON WLD

Name	Acc	EBF	LBMaF	LBMiF	HF
IN-DEDUCTIVE + Pegasos + DAG-Tree	**0.3495**	**0.4239**	**0.1968**	**0.3946**	**0.4790**
TTI (1st)	0.3185	0.3866	0.1920	0.3644	0.4295
anttip (2nd)	0.3152	0.3682	0.1919	0.3038	0.4546
knn baseline (3rd)	0.2724	0.3472	0.1486	0.3016	0.4616
kensk8er (4th)	0.2714	0.3462	0.1519	0.3002	0.4594

TABLE VII
EFFICIENCY WITH IN-DEDUCTIVE APPROACH

Data	Training	CPU time	Test	CPU time
WMD	65,333 models		36,506 leaf categories	
	SVM^{perf}	13.85 hrs	SVM^{perf}	7.9m
	Pegasos	3.8 hrs	Pegasos	10.2m
	TTI [18]	16 hrs	dhlee [5]	12.5m
	anttip [19]	15 days	TTI [18]	10.2m
WLD	863,261 models		478,020 leaf categories	
	Pegasos + DG	16 days	Pegasos + DG	2 days
	683,359 models		478,020 leaf categories	
	Pegasos + DAG-Tree	2 days	Pegasos + DAG-Tree	18 hrs

 https://doi.org/10.17562/PB-54-8

ISSN 2395-8618

It is also noticeable that the DAG-Tree approach not only drastically decrease the computational cost but also can significantly improve the system performances. It decreases the computational cost by reducing the given hierarchy to a new one. Even though we get less data from intermediate to top nodes but the IN-DEDUCTIVE approach based system gets higher accuracy using the proposed DAG-Tree. Since we first perform the DAG-based approach to train each edge from leaf to immediate intermediate nodes in the hierarchy, the original training information remains. Moreover, from bottom to top intermediate levels we perform the Tree-based approach to train each edge in the hierarchy based on number of descendants are associate with a certain parent. Since large-scale data-set is in concern and for each outer iteration, we randomly select samples and update weight vectors from block size– referring to inner iteration, even a less information in the intermediate nodes can significantly improve the system performance. Thus, the DAG-Tree is useful to enhance the HTC for very large-scale hierarchy.

VII. Related Work

TC is a typical multi-class single- and multi-label classification problem. To efficiently solve, Platt [9] proposed a faster training of SVM using sequential minimal optimization (SMO) that breaking a very large quadratic programming optimization problem into a series of smallest possible problems as an inner loop in each outer iteration. The approach is generally 1200 and 15 times faster for linear and non-linear SVMs respectively. Studies to solve multi-class multi-label classification have been summarized in [22], in three smaller data sets with maximum labels of 27 in compare to current front-line of multi-label classification task.

There have been many studies that use local context in HTC [23], [24], [25]. Chakrabarti et al. [23] proposed a Naive-Bayes document classification system that follows hierarchy edges from the root node. Koller et al. [24] applied Bayesian networks to a hierarchical document classification. In our approach, we applied efficient learners which have shown good classification performance with fast training as well as improved the pruning method.

In LSHTC3, the arthur system [26] successfully applied meta-classifiers to the LSHTC task. Meta-classifiers can also be regarded as a sort of pruning. The system employed Liblinear, Max Entropy classifier, and SVM^{light}. The meta-classifier with SVM^{light} achieved 0.4381 on the aspect of accuracy; however relatively slow in compare to Liblinear and Max Entropy on the aspect of efficiency.

The anttip system [19] employed the ensemble of different classifiers by introducing Feature-Weighted Linear Regression. The system also used greedy pruning of the base-classifiers by ranking hypothesis that iteratively removing a classier from the ensemble in the development stage. In LSHTC, the system achieved 0.3152 over the WLD on the aspect of accuracy.

Lee [5] proposed a Multi-Stage Rocchio classification (MSRC) based on similarity between test documents and label's centroids for large-scale datasets. The system used greedy search algorithm in the predicted label set and then compare similarities between test documents and two centroid to check whether more labels are needed or not. The MSRC achieved 0.3974, 0.4326, and 0.6783 in terms of accuracy, LBMiF, and HF respectively for WMD. On the aspect of efficiency the system is much faster than baseline such as K-Nearest Neighbor when the expected number of labels per document are less.

VIII. Conclusions

The IN-DEDUCTIVE approach based system outperformed top-group systems in LSHTC3, w.r.t the most of evaluation metrics in different learning algorithms. This can be attributed to the bias adjustment $\beta = -0.5$ and post pruning. Moreover, the SVM-based systems with the IN-DEDUCTIVE and the DAG-Tree approaches also outperformed the LSHTC4's top-group systems. We believe that, to handle extreme multi-label LSHTC problems, the results will make a useful contribution as an useful performance reference. Our future work includes the development of much more efficient algorithms for large-scale IN-DEDUCTIVE approach based system in HTC.

Acknowledgment

This work has been partially supported by JSPS KAKENHI Grant Number 449001.

References

[1] C. Cortes and V. Vapnik, "Support vector networks," *Journal of Machine Learning*, vol. 20, pp. 273–297, 1995.

[2] S. Dumais, J. Platt, and D. Heckerman, "Inductive learning algorithms and representations for text categorization," in *Proceedings of the 1998 CIKM*, 1998, pp. 148–155.

[3] V. Vapnik, *The Nature of Statistical learning Theory*. Springer, 1995.

[4] X. Han, S. Li, and Z. Shen, "k-NN method for large scale hierarchical text classification for LSHTC3," in *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*, Bristol, 2012.

[5] D. H. Lee, "Multi-stage rocchio classification for large-scale multi-labeled text data," in *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*, Bristol, 2012.

[6] T. Zhang, "Solving large scale linear prediction problems using stochastic gradient decent algorithms," in *Proceedings of the the 21th Conference on Machine Learning*, 2004.

[7] S. Shlev-Shwartz, Y. Singer, and N. Srebro, "Primal estimated sub-gradient solver for SVM," in *Proceedings of the 24th International Conference on machine Learning*, 2007.

[8] C. Hsieh, K. Chang, C. Lin, S. S. Keerthi, and S. Sundarara-jan, "A dual coordinate descent method for large-scale linear SVM," in *Proceedings of the ICML-08*, 2008, pp. 408–415.

[9] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*. MIT Press, 1998.

[10] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithm," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[11] Y. Li and P. Long, "The relaxed online maximum margin algorithm," *Journal of Machine Learning*, vol. 46, pp. 1–3, 2002.

[12] T. Joachims, "A support vector method for multivariate performance measures," in *Proceedings of the 2005 International Conference on Machine Learning*, 2005, pp. 377–384.

ISSN 2395-8618

[13] ——, "Training linear SVMs in linear time," in *Proceedings of the ACM conference on Knowledge Discovery and Data Mining*, 2006, pp. 217–226.

[14] F. Ren and M. G. Sohrab, "Class-indexing-based term weighting for automatic text classification," *Information Sciences*, vol. 236, pp. 109–125, 2013.

[15] M. G. Sohrab and F. Ren, "The effectiveness of class-space-density in high and low-dimensional vector space for text classification," in *Proceedings of the 2nd IEEE International Conference of CCIS*, China, 2012, pp. 2034–2042.

[16] M. G. Sohrab, M. Miwa, and Y. Sasaki, "Centroid-means-embedding: An approach to infusing word embeddings into features for text classification," in *Proceedings of Advance in Knowledge Discovery and Data Mining, LNCS*, vol. 9077, 2015, pp. 289–300.

[17] G. Tsoumakes, I. Katakis, and I. Vlahavas, "Random k-labelsets for multi-label classification," in *Proceeding of the Knowledge Discovery and Data Engineering*, 2010.

[18] Y. Sasaki and D. Weissenbacher, "TTI'S system for the LSHTC3 challenge," in *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*, Bristol, 2012.

[19] A. Puurula and A. Bifet, "Ensembles of sparse multinomial classifiers for scalable text classification," in *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*, Bristol, 2012.

[20] M. Ioannou, G. Sakkas, G. Tsoumakas, and L. Vlahavas, "A dual coordinate descent method for large-scale linear SVM," in *Proceedings of the 2010 IEEE International Conference on Tools with artificial Intelligence*, 2010, pp. 409–419.

[21] Y. Yang, "A study on threshold strategies for text classification," in *Proceedings of the 24th annual international ACM SIGIR conference on research and development in Information Retrieval*, NY, 2001, pp. 137–145.

[22] G. Tsoumakes and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[23] S. Chakrabarti, B. Dom, R. Agrawal, and P. Raghavan, "Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies," *International Journal on Very Large data Bases*, vol. 7, no. 3, pp. 163–178, 1998.

[24] D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," in *Proceedings of the 14th Conference on Machine Learning*, 1997, pp. 170–178.

[25] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng, "Improving text classification by shrinkage in a hierarchy of classes," in *Proceedings of the Fifteenth International Conference on Machine Learning*, 1998, pp. 359–367.

[26] K. L. Wang, H. Zhao, and B. L. Lu, "A meta-top down method for large scale hierarchical classification," in *Proceedings of the 2012 ECML/PKDD Discovery Challenge Workshop on Large-Scale Hierarchical Text Classification*, Bristol, 2012.