Adaptive Classifier Selection in Large-Scale Hierarchical Classification

Ioannis Partalas, Rohit Babbar, Éric Gaussier, Cécile Amblard

Adaptive Classifier Selection in Large-Scale Hierarchical Classification

Ioannis Partalas*, Rohit Babbar, Eric Gaussier, and Cecile Amblard

LIG, Université Joseph Fourier, Grenoble 1
Grenoble, cedex 9, 38041, France
firstname.lastname@imag.fr

Abstract. Going beyond the traditional text classification, involving a few tens of classes, there has been a surge of interest in automatic document categorization in large taxonomies where the number of classes range from hundreds of thousands to millions. Due to the complex nature of the learning problem posed in such scenarios, one needs to adapt the conventional classification schemes to suit this domain. This paper presents a novel approach for classifier selection in large hierarchies, which is based on exploiting training data heterogeneity across the hierarchy. We also present a meta-learning framework for further flexibility in classifier selection. The experimental results demonstrate the applicability of our approach, which achieves accuracy comparable to the state-of-the-art and is also significantly faster for prediction.

Keywords: Hierarchical classification, classifier selection, meta-learning

1 Introduction

Many recent practical applications of text classification have the number of target classes as an added dimension to the complexity of the underlying learning problem. Directory Mozilla and Wikipedia exemplify this research challenge in the domain of text classification, where the number of target classes range from hundreds of thousands to over a million. Due to the enormous effort involved in manually classifying unseen data in such scenarios, automatic classification has assumed significant importance. For large scale classification, an underlying semantic structure, a rooted tree for instance, typically exists among the classes. The taxonomy structure serves as useful prior information aimed at improving classification accuracy and speed. If no semantic structure exists among the classes or is ignored if it exists, one needs to evaluate $O(K)$ one-vs-all classifiers, one for each of the K classes. This technique, also referred to as flat classification, consequently leads to a significant slowdown in prediction performance.

In hierarchical classification, major challenges in obtaining higher classification accuracy include: (i) error propagation, since the overall classification mechanism cannot recover from classification error at top levels of the hierarchy,

* I. Partalas and R. Babbar equally contributed to this work

(ii) data heterogeneity across levels in the hierarchy, and (iii) class size imbalance between positive and negative examples for one-vs-all classification. Another important aspect for large scale hierarchical classification, which is also the main focus of this work, is prediction speed. This factor has largely been ignored while designing classification mechanisms in this domain, but is crucial for acceptable behavior in many applications, such as large scale Question-Answering systems.

2 Related Work and Our Contributions

Various approaches have been proposed for hierarchical classification, which can be broadly divided into one of the two techniques: (i) Big-bang approaches, which train a single classifier for the entire hierarchy and hence are more suited for relatively small-scale problems as in [3] where the number of classes are limited to 1172, and (ii) Top-down approaches, in which the test document is classified at the root and then iteratively following the most confident child class, till leaf node is reached. In [5], an SVM classifier is deployed at each node of the hierarchy, which is relatively slow for training and testing. Deep classification technique [9] is slightly better in terms of accuracy but suffers from the limitation that it needs to train a new classifier for every test document. Refined Experts [2] employs top-down SVM classifiers which are augmented with a bottom-up pass using validation set to correct false negatives. Classifier selection for hierarchical classification on a smaller scale in protein function prediction has been studied in [8]. A related study, with focus on empirical trade-offs of large scale hierarchical classification, has been explored in [1].

This work presents a classification technique which exploits the properties of the data, such as feature to example ratio and data imbalance between the target class and rest of the classes, in a large scale hierarchy. Based on such properties, our algorithm performs automatic classifier selection by choosing either an SVM or a Naive Bayes classifier at the classification nodes of the hierarchy. Additionally, this work formulates and solves a meta-learning problem for dynamic classifier selection. Section 4 presents in detail the criteria which determine the choice of classifiers.

3 Problem Setup

In single-label multi-class hierarchical classification, the training set can be represented by $S = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{N}$. In the context of text classification, $\mathbf{x}^{(i)} \in \mathcal{X}$ denotes the vector representation of document i in the input space $\mathcal{X} \subseteq \mathbb{R}^d$. Assuming that there are K classes denoted by the set $\mathcal{Y} = \{1 \dots K\}$, the label $y^{(i)} \in \mathcal{Y}$ represents the class associated with the instance $\mathbf{x}^{(i)}$. The hierarchy in the form of rooted tree is given by $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $\mathcal{V} \supseteq \mathcal{Y}$ denotes the set of nodes of $\mathcal{G}$, and $\mathcal{E}$ denotes the set of edges with parent-to-child orientation. The leaves of the tree which usually forms the set of target classes is given by $\mathcal{Y} = \{u \in \mathcal{V} : \nexists v \in \mathcal{V}, (u, v) \in \mathcal{E}\}$.

In the above setup, given a new test instance $\mathbf{x}$, the goal is to predict the class $\widehat{y}$. This is typically done by making a sequence of predictions iteratively in a top-down fashion starting from the root until a leaf node is reached. At each non-leaf node $v \in \mathcal{V}$, a score $f_c(\mathbf{x}) \in \mathbb{R}$ is computed for each child c and the child $\widehat{c}$ with the maximum score is predicted i.e. $\widehat{c} = \underset{c:(v,c)\in\mathcal{E}}{\text{argmax}} f_c(\mathbf{x})$.
In addition to requiring *accurate* predictions, we also focus on *prediction speed*, two seemingly contradicting design requirements for a machine learning algorithm. To address the above conflicting requirements of high prediction accuracy and faster prediction and training time, we focus on Support Vector Machine (SVM) and Naive Bayes (NB) classifiers as base classifiers. SVM classifier is known to have better accuracy but is slower to train and deploy for prediction. NB classifier, on the other hand, is faster for training and prediction but is on the lower side of the accuracy spectrum. In the next section, we present conditions in large hierarchies, which determine the selection of classifiers to achieve better run-time performance without sacrificing accuracy.

4 Classifier Selection in LSHC

In this section, we present two approaches to classifier selection that exploit the data heterogeneity in large scale hierarchical classification. The first one, referred to as static approach [1], is based on using the relation between number of features and number of training examples for the classification problem at a given node in the hierarchy. The second approach to classifier selection is based on solving a meta-learning problem which includes further meta-features such as test instance size and class imbalance in one-vs-all classification.

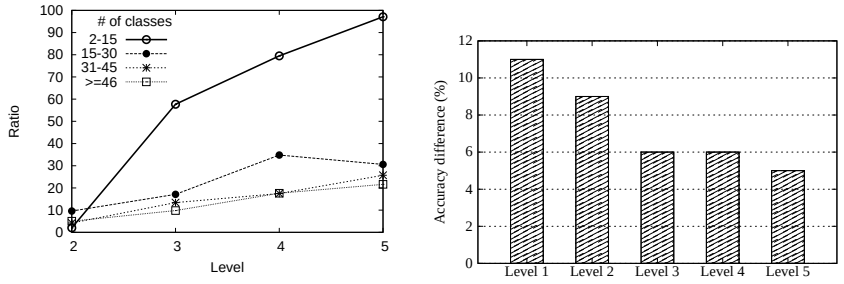
4.1 Static Approach to Classifier Selection

For a multi-class classification problem at node v of the hierarchy, let d_v denote the dimensionality of the feature space and n_v denote the number of training documents for which the root-to-leaf path goes through node v. Let their ratio for node v in the hierarchy be denoted by r_v, i.e. $r_v = \frac{d_v}{n_v}$.

As one traverses down from the root node towards the leaves, in large scale hierarchies such as DMOZ, r_v varies over a wide range of values. Due to large number of classes corresponding to the leaf nodes, the number of training documents (n_v) decrease much faster than the number of features (d_v) for classification nodes on root to leaf path. As a result, this ratio is much higher at hierarchy levels close to the root as compared to its value for nodes at lower levels. Figure 1(a) shows the variation of average value of r_v for DMOZ dataset when plotted against the hierarchy levels. Each piece-wise linear curve in the plot corresponds to the class size range of the multi-class problem. Two important properties of the dataset, one of which follows from Figure 1(a), are:

[1] called Adaptive Classifier Selection in our previous work [1]

1. The ratio r_v increases towards the leaves;
2. Almost 97% of the multi-class classification problems involve 2-15 classes.



(a) Ratio variation (b) Accuracy differences

Fig. 1. 1(a)Variation in ratio of feature set size to training sample size with the hierarchy level, and 1(b) Difference of SVM and NB accuracy, (SVM - NB), in % for each hierarchy level. Level 1 corresponds to the root and level 5 to level leading to leaves.

The above observations imply that in order to achieve the goal of high accuracy and faster run-time, one needs to exploit the wide variation in training data properties. In this context, we next present the relevant results from statistical learning theory which deal with accuracy measure of discriminative classifiers (such as SVM) and generative classifiers (such as NB).

Let f_G and f_D represent the classifiers learned by fitting generative and discriminative model respectively and $f_{G,\infty}$ and $f_{D,\infty}$ be their corresponding asymptotic versions, i.e. functions learned when the sample size approaches infinity. Let $\varepsilon(.)$ be the function representing the generalization error of its argument. For a classification problem in d-dimensional feature space with n training examples, these results can be summarized as follows [6]:

1. $\varepsilon(f_{D,\infty}) \leq \varepsilon(f_{G,\infty})$;
2. $\varepsilon(f_G) \leq \varepsilon(f_{G,\infty}) + \delta_0$ if $n = \Omega(\ln(d))$;
3. $\varepsilon(f_D) \leq \varepsilon(f_{D,\infty}) + \delta_0'$ if $n = \Omega(d)$;
   for arbitrary but fixed $\delta, \delta_0' > 0$; $\Omega(.)$ denotes the big Omega notation.

Informally, the above inequalities can be interpreted as follows:

1. The generalization performance of discriminative classifiers is better than that of generative classifiers under asymptotic regime of operation.
2. The number of training examples required for discriminative classifier to reach its asymptotic performance is at least *linear* in the number of features.
3. The number of training examples required for generative classifier to reach its asymptotic performance is at least *logarithmic* in the number of features.

The above results show that even though SVM is a better classifier for nodes close to the root, NB can be used for nodes in the lower levels of the hierarchy

due to its faster training and prediction time. Figure 1(b) confirms this intuition further, showing that NB accuracy is much closer to that of SVM for lower levels (4 and 5) than at higher levels (1 and 2). Consistently lower accuracy of NB for all levels in the hierarchy can be attributed to the argument indexed 1 above.

In order to allow further flexibility in classifier selection and instead of fixing NB classifier for the *entire* lower levels, we can use a threshold value τ_v for ratio r_v to choose the classifier at node v, such that

$$\text{Classifier at node } v = \begin{cases} \text{Naive Bayes} & \text{if } r_v \geq \tau_v \\ \text{SVM} & \text{otherwise} \end{cases}$$

The thresholding strategy, even though a simplification of the three arguments presented above, works well in practice, as shown in the experimental section 6.

4.2 Adaptive Hierarchical Classifier Selection

Using the classifier selection strategy introduced in the previous section, the choice of a classifier at every node in the hierarchy becomes fixed for all test instances. Further adaptivity can be achieved by enabling classifier selection for each test instance separately. The rationale of incorporating an adaptive selection mechanism is that classification models have different levels of expertise in different parts of the feature space. For example, for a specific node in the hierarchy, a NB classifier may be complementary to an SVM classifier. Thus, it would be desirable to select NB in some test cases targeting to both predictive accuracy and computational performance. The proposed method for hierarchical classifier selection is based on the concept of meta-learning. One of the objectives of meta-learning methods is to produce general and robust learning algorithms [7].

In the context of hierarchical classification, the purpose of the meta-learning framework is to select the best classifier for each node in the hierarchy as we traverse it in a top-down manner. In order to do so, we need to define an appropriate meta-learning problem. Let $S_V = \{(\mathbf{x}_V^{(i)}, y_V^{(i)})\}_{i=1}^{N_V}$ be a validation set containing examples of the initial classification problem and C_a, C_b be two classifiers that have been trained for each multi-class problem in the hierarchy. For every example in the validation set, we create meta-learning examples by traversing the hierarchy top-down. For a specific node v and a validation example i, a meta-learning example is defined by the tuple: $\mathbf{x}_m^{(i,v)} =< \gamma_v, r_v, ubr_v, sz_i >$ such that the elements of the tuple are the meta-features of the meta-learning problem:

- γ_v : number of children of node v.
- r_v : ratio of the number of features to number of training examples for node v.
- ubr_v : the unbalanced ratio of the classification problem at node v, calculated as $\frac{\#\text{examples of minority class}}{\#\text{examples of majority class}}$
- sz_i : the size of instance i. Different classification algorithms behave differently according to the size of the instance and thus it would be desirable to adapt to this phenomenon.

Next, we need to define the label space of the meta-learning problem. Different learning problems can be defined according to the characteristics of the classifiers (prediction accuracy, training and testing time) and the objectives of the hierarchical classification problem (accuracy, computational cost). For example, if we assume that $C_a \prec C_b$ in terms of prediction accuracy and $C_a \succ\succ C_b$ in terms of computational cost, where $\prec$ denotes the natural preference relation, then we set the following learning problem:

$$y_m^{(i,v)} = \begin{cases} C_a & \text{if } C_a \text{ is correct} \\ C_b & \text{otherwise} \end{cases}$$

In this problem the objective is to identify regions where C_a complements C_b while reducing the computational cost in cases where both classifiers have good or bad prediction accuracy. Note that the definition of the meta-learning problem is flexible in order to allow different instantiations by considering different and multiple classification schemes. For example, one may consider multiple classifiers and form a multi-label problem where more than one classifier could be correct for an instance. In this case the method could be coupled with ensemble techniques for acquiring a decision.

5 Experimental Setup

For the experiments we use the publicly available DMOZ data set from LSHTC2 [2]. The dataset, after preprocessing by stemming and stopword removal, appears in the LibSVM format. Table 1 presents the important properties of the dataset. Since the average number of labels per document is 1.02, we consider it as single-label classification for our purpose. We use Liblinear [4] to train the models

Property Name	**Value**
Total number of training examples	394,756
Size of the Overall Feature Space	594,158
Number of Target Classes ($\|\mathcal{Y}\|$)	27,875
Number of Nodes in the Hierarchy ($\|\mathcal{V}\|$)	35,449
Total number of multi-class classifiers	7,574

Table 1. Training Data Properties

for L2-regularized L2-loss support vector classification. The models are trained for all 7,574 non-leaf nodes for One-Vs-All classification. For NB classifier, we implement the standard multinomial NB using Laplace smoothing.

For producing the meta-learning training set we use a validation set (separate from the training set for building the classifiers) containing 3000 examples. As

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

meta-learner, we use a decision tree based on C4.5 algorithm as it can provide interpretable rules. The Weka machine learning library was used in this case setting the confidence factor to 0.25, with reduced error pruning and without Laplace smoothing, using 10-fold cross-validation.

6 Results and Analysis

Table 2 shows the different classification mechanisms and the metrics of interest for the overall classifier, which include, (i) SVM classifier for the entire hierarchy (SVM-TD), (ii) Static classifier selection strategy based on threshold value (SCS-τ), (iii) adaptive hierarchical classifier selection method (AH-CS) for C_a =NB and C_b=SVM, and (iv) NB classifier for the entire hierarchy (NB-TD). We first

Method	**Accuracy (%)**	**Tr. Time (hours)**	**Test Time (secs)**
SVM-TD	35.58	35	20
SCS-τ, $\tau = 60$	35.19	22	12
SCS-τ, $\tau = 30$	34.68	12	5
AH-CS	**35.66**	35.25	**4**
NB-TD	22.22	0.25	0.5

Table 2. Trade-off between Prediction Accuracy in %, Total Training for entire dataset in hours, and Average Test Time per Instance in seconds

notice that the best performing algorithm in terms of accuracy is AH-CS with a slight difference over SVM-TD, while the gain in prediction speed is about 5 times. Note that in the case of AH-CS the training time of the models is the sum of the training times of SVM-TD and NB-TD as we need to retain all the models due to the instance-based nature of the method.

AH-CS selected 45.58% times NB models during the testing procedure. To better understand the results of the proposed method we consider the contingency matrix of the two classifiers for each level of the hierarchy. Noticeably, the NB classifier corrects the errors of SVM in 6.8%, 6.4%, 6.6% and 6.7% of the examples for the first 4 levels respectively. This shows that being able to identify these cases can lead us to better performance both in accuracy and speed. We also calculated the κ-statistic for each level which shows how diverse the classifiers are, getting 0.42, 0.45, 0.44 and 0.45 for the 4 levels respectively. $\kappa = 1$ means that the two classifiers are identical while $\kappa = 0$ indicates independent classifiers. The results show that the classifiers are diverse across the hierarchy and supports the rationale of using different classifiers for different cases. Note that the achieved performance of our method is comparable to the best participant (38.8%) in LSHTC for the DMOZ track. However, the objective of our work does not coincide with the participants' in the challenge since their major focus is on accuracy related metrics. As a result, some of them may employ some pre-processing steps to boost accuracy.

From rows 1 and 3 of table 2, the accuracy of the hierarchical classifier by using static classifier selection is comparable to top-down SVM, while being four times faster in prediction and three times faster to train. SCS-τ was computed based on a uniform threshold value of $\tau_v = 60$ and $\tau_v = 30,\ \forall v \in \mathcal{V}$. Increasing the threshold value selects more SVM classifiers, leading to better accuracy but slower training and test time, while decreasing it would have the opposite effect.

The gain in speed-up for test time is achieved as a result of more compact models built by NB as compared to SVM from same the training data. All NB models can be loaded in the physical memory for predictions. For SVM, the models for only the top two levels can be loaded in physical memory.

7 Conclusions and Future Work

We presented a static and an adaptive classifier combination technique to build large scale hierarchical classification systems. As a result, we not only achieve accuracy comparable to state-of-the-art but also reduce the prediction time significantly compared to the top-down SVM classifier. Further work includes the consideration of more complex topologies such as directed acyclic graphs and also allowing multiple labels.

8 Acknowledgments

This work was supported by the French National Research Agency (ANR) as part of the project Class-Y (ANR-10-BLAN-02).

References

1. R. Babbar, I. Partalas, E. Gaussier, and C. Amblard. On empirical tradeoffs in large scale hierarchical classification. In *ACM CIKM*, 2012.
2. P. N. Bennett and N. Nguyen. Refined experts: improving classification in large taxonomies. In *Int. ACM SIGIR Conference*, pages 11–18, 2009.
3. L. Cai and T. Hofmann. Hierarchical document categorization with support vector machines. In *CIKM*, pages 78–87. ACM, 2004.
4. R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *JMLR*, 9:1871–1874, 2008.
5. T.-Y. Liu, Y. Yang, H. Wan, H.-J. Zeng, Z. Chen, and W.-Y. Ma. Support vector machines classification with a very large-scale taxonomy. *SIGKDD Explor. Newsl.*, pages 36–43, 2005.
6. A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *NIPS*, pages 841–848, 2001.
7. T. Schaul and J. Schmidhuber. Metalearning. *Scholarpedia*, 5(6):4650, 2010.
8. A. Secker, M. N. Davies, A. A. Freitas, E. B. Clark, J. Timmis, and D. R. Flower. Hierarchical classification of g-protein-coupled receptors with data-driven selection of attributes and classifiers. *Int. J. Data Min. Bioinformatics*, pages 191–210, 2010.
9. G.-R. Xue, D. Xing, Q. Yang, and Y. Yu. Deep classification in large-scale text hierarchies. In *Int. ACM SIGIR Conference*, pages 619–626, 2008.