The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

Optimizing Hierarchical Classification with Adaptive Node Collapses

Sujan Perera,[1] **Orna Raz,**[2]
Ramani Routray,[1] **Shenghua Bao,**[1] **Marcel Zalmanovici**[2]
[1]IBM Watson Health, USA
[2]IBM Research, Israel

Abstract

Data intensive solutions, such as solutions that include machine learning components, are becoming more and more prevalent. The standard way of developing such solutions is to train machine learning models with manually annotated or labeled data for a given task. This methodology assumes the existence of ample human annotated data. Unfortunately, this is often not the case, due to imbalanced distribution of classes and lack of human annotation resources. This challenge is exasperated when thousands of hierarchical classes are introduced. Therefore, it is critical to quantify the sufficiency of the data for a given task before applying standard machine learning algorithms. Moreover, it may be the case that there is ample labeled training data to only solve a sub-problem. In particular, in the hierarchical classification problem, the sufficiency level of training data could vary significantly depending on the granularity level of hierarchy we use for classification. We identify a need to decompose the given problem to sub-problems for which there is ample training data. In this paper we propose a methodology to decompose a hierarchical classification problem considering the characteristics of a given dataset. We define an optimization problem of adaptive node collapse that identifies an appropriate hierarchy decomposition based on a trade-off between multiple goals. In our experiments, we consider the trade-off between the learning accuracy and the hierarchy abstraction level.

Introduction

Data intensive solutions rely on data and on machine learning models that generalize from this data. In many of the real-world problems, the quantity and the quality of the data challenge the machine learning algorithms. In particular, the large-scale hierarchical classification problems are challenged by the data distribution and complexity of the hierarchy. This is prevalent in the medical domain where hierarchical classification tasks like coding the medical conditions to ICD hierarchy, coding adverse events to MedDRA hierarchy and coding drug and compounds to WHODD hierarchy are essential for various purposes like diagnosing, billing, monitoring drug safety, secondary analysis tasks and regulatory processes. The main challenges in solving these problems include:

- Long tail distributions, making it infeasible to learn all the data classes (labels), because there is insufficient training data for classes appearing in the long tail.
- Hierarchy of labels. There is a trade-off between the level of label details (there are more details lower in the hierarchy) and the amount of training data available (there is less data lower in the hierarchy).
- Often, the goal is to learn the most specific concept, i.e., cover the lowest level hierarchy classes. Of course, the requirement is to do so with sufficiently high accuracy.

Given these challenges, it is important to understand the limitations of the machine learning models with respect to the quality and quantity of the available dataset. In a hierarchical classification problem, it may not be possible to train a classifier that always classifies instances to a leaf node. For example, if a particular leaf node has only a few training examples or the existing examples are significantly diverse in terms of their semantics, the classification algorithm may find it difficult to learn to classify to this node. In such cases, it may make sense to train a classifier that is capable of classifying to the parent of that node. In this paper, we formulate this problem as an optimization problem. We are unaware of prior work that formulates this problem. The goal of our optimization problem is to find a *surface* in the hierarchy that maximizes a combination of:

1. The classification accuracy and,
2. Coverage of leaf nodes in the hierarchy.

We define a *surface* over a hierarchy to be a set of hierarchy tree nodes that intersect once each path from the root of the hierarchy (highest level node) to the leaves of the hierarchy (lowest level nodes).

For example, consider the classification problem depicted in figure 1. The hierarchy has three levels. The ideal scenario would be that the training data set contains sufficient data with high quality to train a classifier that can classify new instances to one of the seven leaf nodes. However, when the training data does not satisfy the above requirement, it is important to decompose the problem and solve sub-problems. In our example, the idea is to find the segment of the hierarchy where we can train the model to classify at leaf node level and the segment where we have to restrict the classification problem to higher level in the hierarchy. This segmentation is indicated with a line drawn on the hierarchy

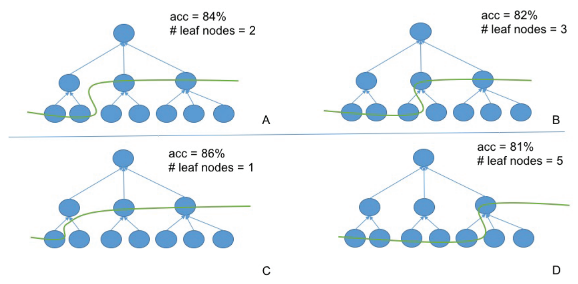

Figure 1: Trade-off between accuracy and leaf node coverage

which we call the *surface*. However, we should keep a balance when we draw the surface on the hierarchy. As shown in the figure 1, as we move the surface up we can potentially get a higher accuracy but we are loosing the specificity of the classification results. Hence, it is important to cover as many leaf nodes as possible while attaining the desired accuracy level.

The defined optimization problem balances the need for accuracy and the coverage of leaf nodes. The method starts with a surface and calculates the accuracy and the number of leaf nodes covered, and optimizes over these characteristics. The task of classifying to the leaves that are not included in the surface should be handled separately. A possible approach is to implement a non learning solution for these leaf classes (e.g., a dictionary). Another possible approach is to implement a learning solution, where there is only a need to classify the nodes under the chosen surface node. For those leaf nodes of the hierarchy not included in the surface, the method determines the correct parent class for the data instance. This allows to train a separate classifier for leaf nodes under this parent to determine their leaf node. Note that this classifier is easier to train relative to the large classifier and would have less impact from the long-tailed data distribution due to a significantly smaller number of classes.

A user may specify a number of leaf nodes to be included in the training data set and the accuracy desired for the classification operation. From these inputs, it is possible to determine the surface that provides the desired number of leaf nodes and accuracy for use in selecting training data. The method optimizes the accuracy plus the highest number of nodes. Additional optimization goals may be added, such as a requirement for high diversity in the content (e.g., textual description) of the leaf nodes.

Methodology

We develop a mechanism for determining a combination of nodes over which to train a machine learning model from an ontology or a labeling hierarchy. Any proposed algorithm needs to balance learning accuracy with leaf nodes coverage. Various considerations exist. For example, some classes in the labeling hierarchy of the training data set do not have enough training data or contain particularly difficult semantics while instances of another class have homogeneous semantic in the training data. In this case, it might make sense to draw the surface covering latter class and the parent of the former class.

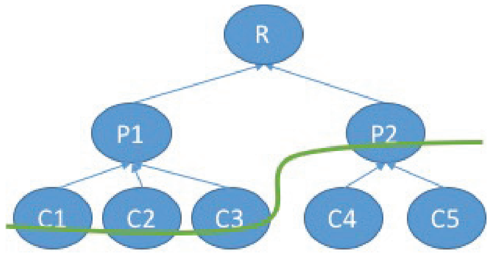

Figure 2: Label Tree example. The line depicts one possible surface or hierarchy decomposition.

We define the following search problem. Given a set of training data for a classification task, our goal is to find a surface in the labeling hierarchy, that maximizes the accuracy of predictions and the coverage of specific classes in the hierarchy.

We are given a training set $(X_1, Y_1), \ldots (X_n, Y_n)$. We are also given a labeling hierarchy T. Each Y_i corresponds to a path in T from root to leaf. A labeling surface, s, is a set of tree nodes that intersect each path from the root to the leaf once. For example, Figure 2 depicts a three level labeling tree:

$$T = \{(R, P1), (R, P2),$$
$$(P1, C1), (P1, C2), (P1, C3),$$
$$(P2, C4), (P2, C5)\}$$

$$s = (C1, C2, C3, P2)$$

Where T is a label tree and s is a possible surface as depicted by the line over the tree nodes.

Given a labeling surface s, we re-label the training set using the surface s as follows. The new label of X_i is defined as the intersection between the surface s and Y_i, $\overset{s_i}{Y} = s \cap Y_i$. We call the learning problem associated with surface s, L_s.

We assume that we apply some learning algorithm to L_s and get an accuracy function $f(L_s)$.

For a given surface s, an adjacent surface s' is a surface that differs from s by only one node. The set of all such adjacent surfaces of s will be denoted as $adj(s)$.

Given a surface s we define its weight to be the sum of depth of all its nodes, denoted by $w(s)$. For example, in our running example the weight is $w(s) = 2 + 2 + 2 + 1$.

Now the following discrete gradient decent greedy search algorithm is defined. Set m to be a weighted hyper parameter of the algorithm.

1. Choose some surface s
2. Calculate $C(s) = f(L_s) + m * w(s)$ For s and any $s \in adj(s)$. Pick s that maximizes C(s).
3. Repeat until C(s) does not improve much.

$$TreeDepthCost = \sum_{j=1}^{n} \sum_{i=1}^{m} w_{ij}$$

where n = number of chosen surface nodes, and
m = number of nodes in path from tree root to node$_j$

Figure 3: Formulating the challenge of learning as much of the lower hierarchy level as possible by maximizing the depth of the chosen surface (all hierarchy trees that are classified); adding a weighing factor w_i such that different cost may be incurred by stopping at different hierarchy levels.

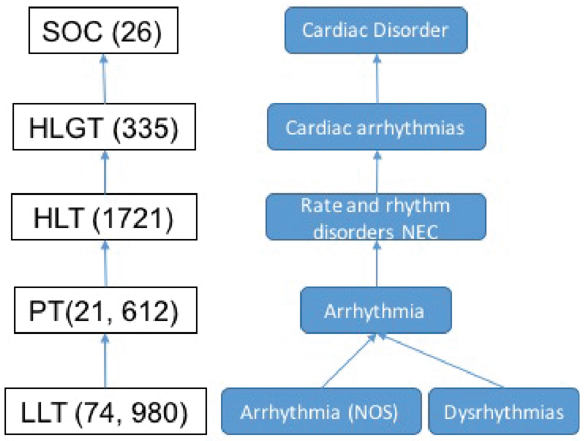

Figure 4: MedDRA hierarchy v 18.1 (left) with an example (right)

Initial results

In our experiments, we examine an aspect of the method suggested in Section Methodology. We show that it is possible to change the accuracy of a learning model by moving up the classification hierarchy while incurring the trade-off of a shorter path from the root. We formulate the challenge mentioned in Section Introduction as an optimization problem: maximize the weighted depth of all hierarchy trees (see Formula 3) and a measure of the learning model accuracy. We chose a weight of 1 for all hierarchy levels, but it is possible to tune the optimization by adjusting the weights. We chose F1-score (F1 score) as the measure of accuracy. Other measures, such as only precision or G-measure, are possible. Our experimental results demonstrate one of the major strengths of our approach: it is generic and can be applied independently of the learning model. We chose three different learning models: Support Vector Classifier (SVC), Multinomial Naive Bayes classifier (NB), and Convolutional Neural Network (CNN).

Dataset

We used a publicly available MedDRA coded drug label dataset for our experiments. A drug label describes how a drug should be used, on what populations and any safety concerns there may exist. One of the elements in a drug label is the expected adverse events. The drug label dataset released for TAC 2017 challenge (TAC 2017) specifies the MedDRA code for the adverse events in the label. MedDRA is a hierarchical vocabulary used to uniquely identify a particular adverse event. The MedDRA hierarchy consists of five levels that are called: System Organ Class (SOC), High Level Group Term (HGLT), High Level Term (HLT), Preferred Term (PT), and Lowest Level Term (LLT) respectively from the root. The adverse events in TAC dataset are coded with MedDRA version 18.1. This version of the MedDRA has 26 SOCs, 335 HGLTs, 1721 HLTs, 21612 PTs, 74980 LLTs (see Figure 4). The TAC dataset consists of 7034 adverse event mentions and they are coded for 1404 unique PT codes. As depicted in Figure 5, the adverse event mentions has a long tailed distribution over the PT codes.

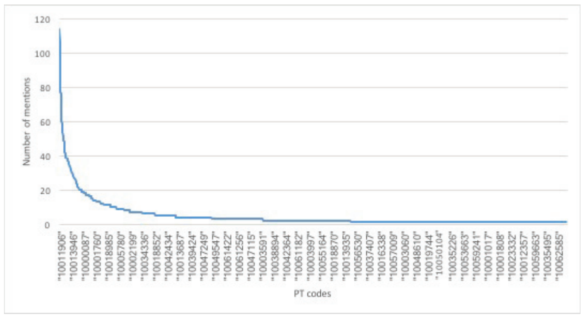

Figure 5: Adverse event distribution over PTs

Experiments and results

To demonstrate that the hierarchy tree level across which we draw the surface affects the accuracy of the trained model, we conducted a number of experiments over three surfaces:

1. The surface was drawn over the PTs (bottom of hierarchy or leaves).
2. The surface was drawn over the PTs when their HLT has five or more adverse events in the dataset and over the HLT otherwise. I.e, we re-label the instances to correspond to the HLT when their PTs has less than five instances (moving them one step up the hierarchy).

$$LabelClass = \begin{cases} HLT & \text{if} \sum data\_points \in HLT < 5 \\ PT & \text{otherwise} \end{cases}$$

3. The surface was drawn over the SOCs (top of the hierarchy, below 'root').

To show that the accuracy variation over different surfaces is consistent across various machine learning algorithms, we chose the simple Multinomial Naive-Bayes (NB), the more sophisticated Support Vector Classifier (SVC), and the Convolutional Neural Network (CNN). Table 2 shows the results of our experiments.

As expected, the results show that accuracy improves as we go up in the hierarchy by re-labeling the data points. The second table column shows how many distinct classes were

Table 1: Experiment Results

Surface	# Classes	Algorithm	F1-score
1(PT)	1494	SVC	0.81
	1494	NB	0.58
	1494	CNN	0.77
2(PT-HLT)	1394	SVC	0.82
	1394	NB	0.59
	1394	CNN	0.78
3(SOC)	25	SVC	0.91
	25	NB	0.90
	25	CNN	0.92

Table 2: Results of measuring accuracy using F1-score for three different surfaces: 1(PT) lowest level leaves (PT), 2(PT-HLT) in-between level that includes mostly lowest level nodes (PT) and some higher level nodes (HLT), and 3(SOC) highest level of the hierarchy (SOC). The accuracy is best at the top of the hierarchy, poorest at the lowest level, and in-between for the in-between level. Consistent results using three different learning models are shown to emphasize that our approach is generic.

chosen for each surface. The accuracy gap between each surface is significant; it should be noted that we have not penalized the accuracy reported here based on the level of the code in the hierarchy. The results are very consistent among multiple algorithms. This shows that the long-tailed distribution of the data impacts all learning methods in the same way. Hence, a systematic solution to find the optimal surface on a hierarchical vocabulary would help to improve the results of a learning problem regardless of the algorithm being used.

Related work

To the best of our knowledge the problem considered in this paper is not addressed in the current literature. However, there exist work related to clustering and classification techniques when a long-tailed distribution of data is present. Zhu et. al present an idea of sharing the training examples of rare categories in order to overcome this challenge (Zhu, Anguelov, and Ramanan 2014). The method proposed in (Jiang 2013) suggests to determine the number of classes for a classification problem based on their distribution. It suggests to group the data values into two parts around the arithmetic mean and continues the partitioning for values above the mean iteratively until the head part values are no longer long-tailed distributed. This way the number of classes is naturally determined by the data. None of these methods addresses the problem of classifying long-tailed data distributions with respect to a given hierarchical vocabulary where there is no flexibility in determining the number of classes. We propose a novel approach that determines the abstractness level, expressed as a surface in the hierarchy, that can be used to train the main classification algorithm, and allows to apply different classification algorithms to the classes where the main algorithm determines only the higher level category.

Discussion and conclusions

We formulate the challenge of training sufficiently accurate classifiers on long tail dataset distributions when a hierarchy or ontology of classes exists. We propose a generic method for addressing this challenge. Our method defines an optimization problem, balancing learning accuracy with coverage of the lowest level hierarchy nodes in the label tree.

Our experiments indicate that indeed there is a trade-off between the learning accuracy and the hierarchy abstraction level. This supports our suggested method which is to treat the challenge as a multi-goal optimization problem.

However, we have yet to experiment with the many potential optimization algorithms that could apply. Different cost functions can be explored, as well as different accuracy measures, as briefly discussed in Section Methodology.

Acknowledgements

We thank Eitan Farchi for sharing ideas and providing feedback that greatly contributed to this work.

References

Wikipedia Convolutional Neural Network. https://en.wikipedia.org/wiki/Convolutional_neural_network.

Wikipedia F1 score. https://en.wikipedia.org/wiki/F1_score.

Jiang, B. 2013. Head/tail breaks: A new classification scheme for data with a heavy-tailed distribution. *The Professional Geographer* 65(3):482–494.

Wikipedia Naive Bayes Classifier. https://en.wikipedia.org/wiki/Naive_Bayes_classifier.

Scholarpedia Support Vector Classifier. http://www.scholarpedia.org/article/Support_vector_clustering.

TAC 2017. https://bionlp.nlm.nih.gov/tac2017adversereactions/.

Zhu, X.; Anguelov, D.; and Ramanan, D. 2014. Capturing long-tail distributions of object subcategories. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 915–922.