An adaptive probabilistic classification method for dynamic class hierarchies

César Ferri[1], María José Ramírez-Quintana[1], Meelis Kull[2], Adolfo Martínez-Usó[1], and Peter Flach[2]

[1] DSIC, Universitat Politècnica de València
Camí de Vera s/n, 46022, València, Spain
{cferri,mramirez,admarus}@dsic.upv.es
[2] Intelligent Systems Laboratory
University of Bristol, United Kingdom
{Meelis.Kull,Peter.Flach}@bristol.ac.uk

Abstract. Many classification tasks involve a large number of categories which are ordered in a hierarchy. In hierarchical classification a model learns to distinguish between the classes by using the structure of the hierarchy. This paper investigates a hierarchical classification framework in that the class hierarchies can dynamically change from learning to deployment time. We propose a new probabilistic method that uses the hierarchy information in order to determine the predicted class. This fact makes our approach able to handle changes in the hierarchy by adapting the decisions of the model to the new class structure. To evaluate our method in this dynamic scenario, we define an evaluation metric based on the class hierarchy. We experimentally analyse the performance of our approach over a collection of datasets using different classification techniques for learning the model. The dynamic scenario defined for the experiments simulate the change of hierarchy by a process that allows us to extract different taxonomies from data. The results show that the performance of the hierarchical probabilistic model over the new hierarchy is competitive.

Keywords: Classification, Hierarchical Classification, Probability estimation, Class Hierarchies as Contexts

1 Introduction

Traditional classification problems (binary or multi-class) assume the independence between classes. However, in many real problems the set of classes are structured in a hierarchy according to a *is-a* relationship. Examples can be found in a wide variety of domains such as bioinformatic, text categorization, film or music classification, etc. The field of hierarchical classification aims to predict the class label of unseen instances when the classes are organised in a hierarchy [25].

Hierarchical classification methods can be structured in different manners. Following the nomenclature introduced in [25], methods that only predict leaf classes are called "mandatory leaf-node", whereas methods that can predict any class in the hierarchy are referred as "non-mandatory leaf-node". On the other hand, regarding the way in that the hierarchical structure is explored, hierarchical classification methods can also be grouped into three categories [25]: the flat approach (when the hierarchy is ignored), the local approach (when a classifier for each internal node is used, it is commonly denoted as top-down in the literature) and the big-bang (global) approach (when there is only one classifier that deals with the entire hierarchy).

The *flat* method consists in transforming the hierarchical problem into a multi-class problem that only considers the set of leaf nodes. Hence, a (traditional) classifier is learnt by completely ignoring the internal nodes of the class tree. It is the simplest method to deal with hierarchical problems.

Usually, when a hierarchical problem is addressed the class hierarchy is either provided a priori (by the user or an expert) or it can be inferred from data ([18], [5]). But independently of how the hierarchy has been obtained, it is clear that hierarchies are always arbitrarily determined (note that, even for the same set of classes, different people might organize it into different hierarchies). As a consequence, for a given hierarchical problem, there is not a unique class taxonomy that can be used but a collection of taxonomies depending on the criteria applied to organise the classes. This is a well-known problem in document classification, where the concept hierarchy changes very often. The following example inspired in [8] illustrates this point.

Example 1. Consider the problem of classifying open-air sports from the GPS tracks registered during the user activities. The list of sports considered are: Cycling, Hiking, Kayaking, Motorcycling, Mountain Biking, Mountaineering, Running, Sailing, Trial Bike and Trial Running. It is not difficult to induce a hierarchical structure by using the similarities among the ten sports. Figure 1 shows the sport hierarchy obtained by grouping the sports by locomotion form, surface type and speed. Note that the original classes are the leaf nodes.

There are other possible criteria for grouping the sports. For instance, Figure 2 shows a different class hierarchy tree (based on [15]) obtained by simply grouping the sports according to the level of intensity (low, medium, high), in terms of cardiovascular demands (cardiac output and blood pressure) generally required to perform that sport.

Let us imagine that we have learnt a classifier using the hierarchy in Figure 1. Now, let us suppose that we are interested in analysing the GPS tracks for medical reasons. So, a hierarchy such as the one depicted in Figure 2 would be more appropriated to this end. How should we proceed? Obviously we can discard the actual model and learn a new model using the new hierarchy. In general, this option could be costly depending on the number of classes and how often the hierarchy changes. Nevertheless, given that we already have a model able to predict the sport activity, another alternative would be to use the classifier in hand and to adapt its predictions to cope with the new hierarchical situation. This is the approach we study in this paper.

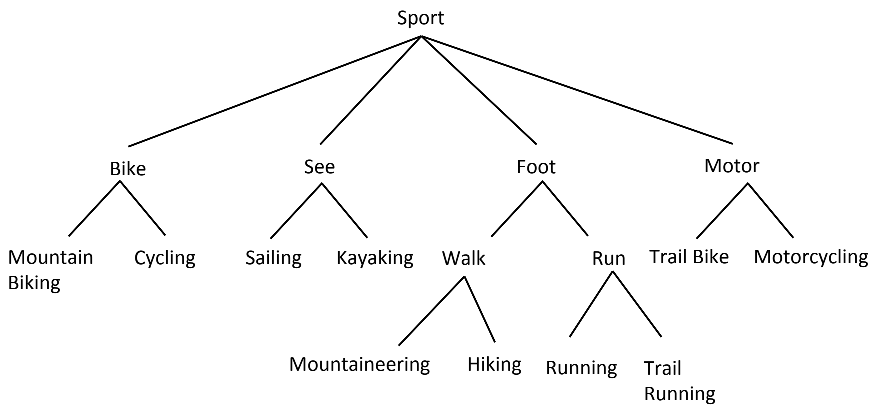

Fig. 1. Class hierarchy of sports based on locomotion form, surface type and speed.

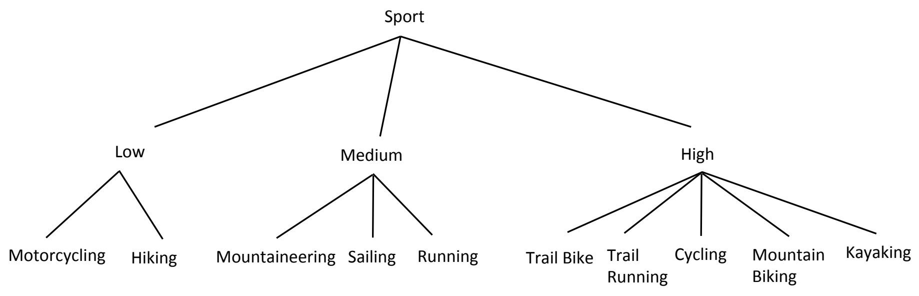

Fig. 2. Class hierarchy of sports based on the level of intensity required to perform the activity.

In order to do this, we propose a new hierarchical classification method based on probability estimations. We focus on hierarchical problems that have the objective of predicting leaf classes. The idea is to learn a classifier that first calculates the probability of belonging to any leaf class and then uses the class hierarchy in order to return the class that minimizes the classification error. The error function implicitly takes into account the hierarchy by using a suitable distance function defined between leaf classes. So, our method can be considered as a 'mandatory leaf global hierarchical method' because it generates a unique classifier that takes the class hierarchy into account to determine the predicted leaf class.

Moreover, we define an evaluation metric based on the class hierarchy which allows us to study how our method performs when the hierarchy changes between training and test. We consider that the set of initial classes (the leaves of the hierarchy) remain unchanged during all the process. This means that the training and test hierarchies only differ in the internal structure of the tree (as happens in Example 1 where the ten sports are the leaves in both taxonomies).

On the other hand, when working with probabilities, it has been proved that not all classifiers provide well-calibrated probabilities, some being over-confident while others being under-confident [4]. Thus, a calibration of predicted probabilities is often recommended. In this paper we calibrate the predicted probabilities by applying one of the most commonly used calibration methods, the PAV calibration technique [2], also known as isotonic regression [22].

Additionally, to evaluate the performance of our method, we conducted some experiments using different classification techniques, different datasets and several taxonomies inferred from the data.

The paper is organised as follows. In Section 2 we define the hierarchical classification method we use in the paper and introduce an evaluation metric based on a distance function between classes directly derived from the hierarchy tree. Section 3 describes the experiments carried out to evaluate our method in a scenario in that hierarchy in the test stage is different than in the training one. Section 4 summarises some related works. Finally, conclusions and future works are outlined in Section 5.

2 Hierarchical Classification Measure and Method

Several performance metrics have been specifically defined for the hierarchical classification task (a review of them can be found in [6]). In this work we use a distance-based metric, that is a measure that considers the distance between the predicted and the actual class in the tree of classes. The measure is based on the rationale that classes that are close to each other in the hierarchy tend to be more similar than other classes that are further away in the hierarchy. Formally, let C be the set of classes that correspond to the leaves of a class tree T. We define the *distance* between two classes $c_i, c_j \in C$ in T as

$$dis_T(c_i, c_j) = \frac{d(c_i, c_j)}{dis_{max}(T)} \tag{1}$$

where $d(c_i, c_j)$ denotes the number of edges that traverses the shortest path in T between classes c_i and c_j. Note that, as the denominator $dis_{max}(T) = max_{\forall c_i, c_j \in C, i<j} d(c_i, c_j)$ represents the number of edges of the longest path in the tree, the values given by dis_T are always between 0 and 1. Then, given an example e and a hierarchy tree T, the *hierarchical loss* measure is defined as

$$hloss_T(e) = dis_T(c_p, c_a) \tag{2}$$

where c_p and c_a are the predicted and actual classes of example e, respectively. A similar evaluation metric was used in [26] and [30]. In what follows, we will omit the subindex T when the tree is clear from the context.

For instance, given the class tree in Figure 1, if the predicted class for an example e is Mountain Biking (mb) and the actual class is Trail Bike (tb), we have that the shortest path between these two classes traverses 4 edges, hence $d(mb, tb) = 4$. Given that the longest path in the tree has 5 edges, we have $hloss(e) = 4/5$.

Now, we introduce a new hierarchical classification technique, that we name *hierarchical probabilistic* method, that returns leaf classes by using a flat classifier and the class hierarchy. The idea is using the scores computed by a flat classifier M (learnt by using the set of classes C) as probability class estimations. With these estimations and the distance dt derived from the class hierarchy T, we can compute the expected loss for all classes and then select the class with the lowest expected loss. This is similar to labelling instances that minimise the expected missclassification cost in cost-sensitive classification. Let $p(c|e)$ the estimated probability that example e belongs to a class $c \in C$ given by M. We define the function *hprb* for classifying examples with respect to hierarchy T as:

$$hprb(e) = argmin_{c \in C}\left(\sum_{\forall k \in C} p(k|e) * hloss_T(k, c)\right) \tag{3}$$

Therefore, our probabilistic hierarchical method makes its predictions based on two completely independent components: a flat model (that estimates the scores) and a hierarchy (from which the distance is calculated). That makes our method able to be adapted easily to work with other hierarchies without the needed of retraining the model M. More concretely, the adaptation of the *hprb* method to deal with a new hierarchy NT consists in replacing in Equation 3 the distance term that is now calculated with respect to NT, that is, $hloss_{NT}(k, c)$ is used instead of $hloss_T(k, c)$.

3 Experiments

In this section we evaluate the performance of the *hprb* method in contexts where the hierarchical structure is variable. For that reason we are going to use ten multi-class datasets. A description of the datasets is included in Table 1.

These datasets are not hierarchical originally. However we can use techniques that are able to induce hierarchies in the classes automatically. Different techniques have been proposed to build a hierarchy from the set of classes. Methods based on clustering consists inferring a distance or similarity function between classes and then applying a hierarchical clustering algorithm. From the dendrogram, the taxonomy of classes is derived by considering that the clusters created between the leaves (the set of original classes) and the root constitute the internal nodes of the hierarchy. Those internal nodes are virtual classes created during the learning process. This approach has been more frequently applied for learning taxonomies from texts ([18], [5]). The learning of taxonomies is an emergent topic given its connection with the construction of ontologies from data [24]. In the fields of the semantic web and natural language processing, other methods based on lexical-syntactic patterns that define the relations between the concepts have been proposed ([17], [23]). Here, we build a confusion matrix running a ten fold cross validation methodology with a classification technique. This matrix

dataset	NumInst	NumAtt	Numclass
1 anneal	898	39	5
2 glass	214	10	6
3 zoo	101	18	7
4 autos	205	26	6
5 grub-damage	155	9	4
6 soybean	683	36	19
7 eucalyptus	736	20	5
8 vowel	990	14	11
9 pendigits	10992	17	10
10 segment	2310	20	7

Table 1. Information about the multiclass datasets used in the experiments: number of instances, attributes and classes.

is converted into a distance matrix between classes, and from the distances a dendrogram is built. This hierarchical clustering reflects which classes present more common errors and therefore are more similar. The hierarchy of classes is directly derived from this dendrogram.

In the experiments we use two class hierarchies. The first one, old context T, is used in the training step while the second one is considered as the new context NT and is employed in the test step. In order to obtain two different (and diverse) trees of hierarchies we use ZeroR classification technique (it always predict the majority class) from Weka [29] for the old context and a decision tree ”J48” for the new context.

Weka [29] is a suite of java packages that contains a wide variety of machine learning techniques, and concretely, a huge amount of classification methods. In the experiments we apply twelve classification techniques of Weka in a R [21] script by means of the library RWeka [12] and caret [11]. Specifically, we use the following twelve classification methods: a decision tree ”J48”, an unpruned decision tree with Laplace correction “J48Unp”, a propositional rule learner ”JRip”, logistic regression ”Logist”, Naive Bayes ”NB”, K-nearest neighbours with ten neighbours ”IBK”, Random Forest ”RF”, a combination of ten J48 models by the Bagging technique ”Bagging”, a decision list ”PART”, a combination of ten J48 models by the Boosting technique ”Boosting”, a decision stump technique “Stump” , and a Boosted Logistic Regression ”LB” [28].

We employ the following methodology. Each dataset is randomly divided into train (50%) and test (50%). Using the methodology introduced previously, we build an old context hierarchy and a new context hierarchy. We learn a model classifier from the train data and the model is evaluated with the test data. We use *hloss* as a performance measure. We also include accuracy error and Mean Square Error to know the quality of the probabilities predicted by the classifier. In order to obtain significant results we repeat the experiments 300 times.

As an example of the results , Table 2 includes the results for the ten datasets and “Naive Bayes” learning technique (averages of 300 iterations). The first column includes the *flat_error* classification error for each dataset. *flat* represents the performance of the flat classifier for the hierarchical loss measure considering the new context NT. *hpnt* is the performance of the hierarchical probabilistic method *hprb* considering the new context NT in the expected loss estimation, while *hp* considers the old context hierarchy T in the expected hierarchical loss estimation. Since the *hprb* method is mainly based on the quality of probabilities, we analyse the effect of calibration over the estimated probabilities. For this purpose, we employ PAVCal function from the code introduced in [3] based on the PAV calibration method [7]. We use 20% of the train data as a validation data for the calibration process. We include in the table of results the Mean Squared Error of the classifier before calibration *MSE* and after calibration *MSEcal*. MSE can be considered as a measure that indirectly can estimate the calibration of the estimated probabilities [9]. Columns *hpntcal* and *hpcal* contain the results of the *hprb* method after the PAV calibration phase with respect to the new context hierarchy and the old context hierarchy (in the expected loss estimation), respectively.

	dataset	flat_error	flat	hpnt	hp	hpntcal	hpcal	MSE	MSEcal
1	anneal	0.1216	0.0929	0.0929	0.0929	0.0748	0.0748	0.0449	0.0321
2	glass	0.5159	0.2917	0.2899	0.2902	0.3206	0.3215	0.1342	0.1228
3	zoo	0.0461	0.0261	0.0262	0.0265	0.0284	0.0300	0.0110	0.0145
4	autos	0.4613	0.3128	0.3123	0.3130	0.3203	0.3224	0.1347	0.1161
5	grub-damage	0.5189	0.3776	0.3757	0.3760	0.3742	0.3748	0.1806	0.1768
6	soybean	0.1627	0.0720	0.0673	0.0718	0.0593	0.0652	0.0129	0.0131
7	eucalyptus	0.5254	0.3709	0.3708	0.3707	0.3686	0.3695	0.1672	0.1380
8	vowel	0.4006	0.2184	0.2153	0.2170	0.2164	0.2194	0.0493	0.0498
9	pendigits	0.1426	0.0956	0.0957	0.0956	0.0937	0.0929	0.0257	0.0213
10	segment	0.1974	0.1048	0.1048	0.1048	0.0873	0.0879	0.0518	0.0320
	Average	0.3092	0.1963	0.1951	0.1959	0.1944	0.1958	0.0812	0.0716

Table 2. Performance results with Naive Bayes averaged after 300 iterations. From left to right: classification error for each dataset (*flat_error*), performance in *hloss* of the flat classifier for the hierarchical loss measure considering the new context NT (*flat*), hierarchical probabilistic methods considering the new context NT in the expected hierarchical loss estimation (*hpnt*) , and considering the old context hierarchy T (*hp*). *hpntcal* and *hpcal* present the result of these methods after calibration. Finally, the Mean Squared Error of the models before calibration *MSE* and after calibration *MSEcal* .

For the new hierarchy and considering *hloss*, we observe that our method is adaptative with respect to the new context. We can see this, if we examine the differences between *hpnt* and *hp* columns. In *hp* we show the performance of the *hprb* method adapted to the old context hierarchy T (the expected loss is computed using T), while in *hprb* the labels are assigned considering the new context NT . In all cases except from two, *hpnt* is better than *hp* We can also find that *hprb* method performs better than the flat approach ($flat$ column) in average. Only in two datasets (3 and 9) the flat classifier outperforms the *hprb* method although the differences are very small ($\simeq 0.001$).

When comparing the results with and without applying calibration (*hpntcal* and *hpnt* columns) a slight improvement is observed in average when the probabilities are calibrated. This is consistent with the results observing the MSE measure where in 8 datasets the calibrated *hprb* method exhibits lower MSE values than the uncalibrated version. However, for datasets 2 and 4 a better value in MSE does not mean a better hierarchical loss.

Table 3 summarises the different methodologies analysed. In this table we include the averages of the twelve classification methods analysed for the ten datasets. Here we consider classification error ($flat\_error$), the performance of the flat classifier for the hierarchical loss measure, ($flat$), the performance of the hierarchical probabilistic method *hprb* considering the new context NT in the expected hierarchical loss estimation (*hpnt*) and the results after a PAV calibration (*hpntcal*). We also include a pairwise comparison (Wins-Ties-Losses) of *hpntcal* versus $flat$ for the 300 executions for each of the ten datasets. Finally, we include MSE of estimated probabilities before (MSE) and after calibration ($MSEcal$) and a pairwise comparison between $MSEcal$ and MSE. In the last row of the table we can see the total results for the pairwise comparisons.

	Method	flat_error	flat	hpnt	hpntcal	hpntcal vs flat	MSE	MSEcal	MSEcal vs MSE
1	J48	0.2558	0.1655	0.1655	0.1618	905-1284-811	0.0679	0.0609	1745-406-849
2	JRip	0.2844	0.1831	0.1830	0.1846	96-2607-297	0.0689	0.0664	1785-520-695
3	logist	0.2738	0.1760	0.1759	0.1798	876-750-1374	0.0861	0.0771	1884-161-955
4	NB	0.3092	0.1963	0.1951	0.1944	1993-139-868	0.0812	0.0716	2245-48-707
5	IBK	0.3862	0.2484	0.2383	0.2333	2102-52-846	0.0728	0.0731	1811-40-1149
6	RF	0.2098	0.1379	0.1393	0.1352	1362-176-1462	0.0542	0.0523	1999-38-963
7	bagging	0.2301	0.1497	0.1497	0.1475	1199-309-1492	0.0546	0.0555	1310-99-1591
8	PART	0.2609	0.1679	0.1684	0.1641	770-1689-541	0.0693	0.0621	1963-376-661
9	boosting	0.2198	0.1427	0.1426	0.1450	1006-799-1195	0.0638	0.0571	2340-326-334
10	lb	0.2446	0.1578	0.1566	0.1564	1269-294-1437	0.0568	0.0573	1298-88-1614
11	J48Unp	0.2542	0.1642	0.1692	0.1672	801-624-1575	0.0643	0.0619	2227-44-729
12	stump	0.6145	0.3953	0.3752	0.3761	1567-1259-174	0.0987	0.1001	613-1145-1242
Total						13946-9982-12072			21220-3291-11489

Table 3. Average results of the classification methods. Columns *hpntcal vs flat* and *MSEcal vs MSE* shows a pairwise comparison (wins-ties-losses) between *hpntcal* and *flat* classifiers and *MSEcal* and *MSE* error measures respectively.

Without calibration, if we consider averages, the summary is that the flat classifier and our hierarchical method obtain similar results (except from IBK, J48Unp and stump). When we apply calibration, *hprb* method improves generally the results although there are four exceptions (JRip, logist boosting and stump). For the last case stump, *hprb* is not able to enhance the results since the calibration process is not useful as the results in MSE show. We need to analyse the other three cases in order to understand why the improvement in MSE is not reflected in *hpntcal*. To have more insight into the causes of these results, the decomposition of MSE in calibration and refinement loss [10] could be useful to analyse better the effect of calibration on the *hprb* method . On the other hand, the small differences observed for the hierarchical loss between methods may be due to the fact that the class hierarchies are similar. Hence, it should be convenient to expand the analysis with a more diverse range of hierarchies or different performance measures for hierarchical classification. This is confirmed looking at the number of ties, wins and loss of each method ($hpntcal\ vs\ flat$ column).

4 Related Work

In hierarchical classification, probabilistic methods have been mainly applied for text categorisation. One of the first proposal was [16] which followed the top-down philosophy. In that paper, document classification is solved by decomposing the problem according to the class hierarchy and constructing a hierarchical set of classifiers (one at each node of the hierarchy). The idea is that each classifier is built by only considering the set of relevant features for this particular (local) problem. The authors used the Naive-Bayes method with a double purpose: for deciding the appropriate feature set at each node, and for constructing the set of classifiers. Other related approaches based on a collection of Naive-Bayes classifiers for exploiting the hierarchical structure of classes have also been proposed for the classification of images [1] or web content [20]. Authors in [19] improved the performance of a Naive-Bayes text classifier by applying a statistical technique (called shrinkage). The idea is to smooth the parameter estimates of a child node by interpolation with all its ancestors in the hierarchy. An alternative approach based on a hierarchical mixture model is presented in [27]. In this paper, the hierarchy is used to improve the estimates of the class-conditional term probabilities by obtaining a differentiation of words in the hierarchy according to their level of generality/specificity. In any case, unlike our method, all of the mentioned approaches use the same hierarchy for both training and test. Moreover, since the hierarchy is exploited for determining the learning parameters, the adaptability of the model to another hierarchy is very limited and would probably require learning the model again.

In fact, up to our knowledge, the use of several taxonomies has been only considered during the training step in order to improve the performance of the classifier. In [13] the authors propose a framework, called *class adaptation*, in that the learning data is augmented with auxiliary samples labelled with different taxonomies. After that, the method computes the weight that represents the similarity between the target and the auxiliary classes, and then a classifier for target classes that minimises the weighted error is built. [14] presented a method to modify a given taxonomy automatically to achieve a better hierarchy according to the data. The algorithm performs a search in the hierarchy space trying to maximise the likelihood of data given a hierarchy. An approximated solution is found by assuming that the optimal hierarchy should reside within the vicinity of the original hierarchy and guiding the search by a criterion that always chooses in each step the neighbour node with largest likelihood improvement.

5 Conclusions

This paper has addressed the problem of hierarchical classification in dynamic contexts where the hierarchical structure of classes can be modified. We propose a method based on the probabilities estimated by multiclass methods that predicts the label that minimises the expected loss with respect to the current context expressed (class hierarchy) and in that way is able to adapt the predictions to the novel context. We have analysed the performance of the proposal by comparing the results over ten multiclass datasets and twelve learning methods. Hierarchies are artificially induced by computing similarities between classes considering mutual missclassifcations. Probabilistic methods are mainly based on the quality of the estimated scores of classifiers. We have studied the effect of applying multiclass calibration over probabilities (PAV Calibration). As future work, we propose to consider other scenarios such as "non-mandatory leaf-node" problem, where classifiers can predict any class in the hierarchy (not only in the leaves), or more drastic changes in the hierarchy of classes such as fusion or deletion of classes. We also want to explore the adaption to this dynamic contexts of hierarchical methods such as the Top Down approach.

Acknowledgements

This work was supported by the Spanish MINECO under grant TIN 2013-45732-C4-1-P, and the REFRAME project, granted by the European Coordinated Research on Long-term Challenges in Information and Communication Sciences Technologies ERA-Net (CHIST-ERA), and funded by MINECO in Spain (PCIN-2013-037) and by Generalitat Valenciana PROMETEOII2015/013.

References

1. A. Jain A. Vailaya, M. Figueiredo and H. Zhang. Image classification for content-based indexing. *IEEE Transactions on Image Processing*, 10(1):117–130, 2001.
2. M. Ayer, H. Brunk, G. Ewing, W. Reid, and E. Silverman. An empirical distribution function for sampling with incomplete information. *Annals of Mathematical Statistics*, 5:641–647, 1955.
3. Antonio Bella, César Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. On the effect of calibration in classifier combination. *Applied Intelligence*, pages 1–20, 2013.
4. Antonio Bella, Csar Ferri, José Hernández-Orallo, and María José Ramírez-Quintana. Calibration of machine learning models. In E. S. Olivas, J. D. Guerrero, M. Martinez-Sober, J. R. Magdalena-Benedito, and A. J. Serrano López, editors, *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pages 128–146. IGI Global, 2010.
5. P. Cimiano, A. Hotho, and S. Staab. Comparing conceptual, divisive and agglomerativeclustering for learning taxonomies from text. In *16th Eureopean Conference on Artificial Intelligence, ECAI'2004*, pages 1–5, 2004.

6. E.P. Costa, A.C. Lorena, A.C.P.L.F. Carvalho, and A.A. Freitas. A review of performance evaluation measures for hierarchical classifiers. In *Evaluation Methods for Machine Learning II: papers from the AAAI-2007 Workshop, AAAI Technical Report WS-07-05*, pages 182–196. AAAI Press, 2007.
7. Tom Fawcett and Alexandru Niculescu-Mizil. PAV and the ROC convex hull. *Machine Learning*, 68(1):97–106, 2007.
8. Cèsar Ferri. Hierarchical classification of sport activities from gps tracks. Technical report, Universitat Politcènica de València, 2015.
9. César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern Recognition Letters*, 30(1):27–38, 2009.
10. Peter Flach and Edson Takashi Matsubara. A simple lexicographic ranker and probability estimator. In *ECML*, pages 575–582. Springer, 2007.
11. Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, and the R Core Team. *caret: Classification and Regression Training*, 2014. R package version 6.0-30.
12. Kurt Hornik, Christian Buchta, and Achim Zeileis. Open-source machine learning: R meets Weka. *Computational Statistics*, 24(2):225–232, 2009.
13. Tomoharu Iwata, Toshiyuki Tanaka, Takeshi Yamada, and Naonori Ueda. Improving classifier performance using data with different taxonomies. *IEEE Transactions on Knowledge and Data Engineering*, 23:1668–1677, 2011.
14. L. Tang J. Zhang and H. Liu. Automatically adjusting content taxonomies for hierarchical classification. In *Proceedings of the 4th Workshop on Text Mining*, pages 105–113, 2006.
15. Peter Snell Jere H. Mitchell, William Haskell and Steven P. Van Camp. Task force 8: Classification of sports. *American College of Cardiology*, 45(8):1364–1367, 2005.
16. Daphne Koller and Mehran Sahami. Hierarchically classifying documents using very few words. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, pages 170–178, 1997.
17. Zornitsa Kozareva, Ellen Riloff, and Eduard Hovy. Semantic class learning from the web with hyponym pattern linkage graphs. In *ACL-08: HLT*, pages 1048–1056. Association for Computational Linguistics, 2008.
18. T. Li, S. Zhu, and M. Ogihara. Hierarchical document classification using automatically generated hierarchy. *Intelligent Information Systems*, 29:211–230, 2007.
19. Andrew K. McCallum, Ronald Rosenfeld, Tom M. Mitchell, and Andrew Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '98, pages 359–367, 1998.
20. Neetu. Hierarchical classification of web content using naive bayes approach. *Computer Science and Engineering*, 5(5):402408, 2013.
21. R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2015.
22. T. Robertson, F.T. Wright, and R.L. Dykstra. *Order Restricted Statistical Inference.* John Wiley and Sons, 1988.
23. D. Sanchez and A. Moreno. Pattern-based automatic taxonomy learning from the web. *AI Communications*, 21:27–48, 2008.
24. V. Schickel-Zuber and B. Faltings. Using hierarchical clustering for learning the ontologies used in recommendation systems. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'2007*, pages 599–608, 2007.

25. Jr. Silla, N. Carlos, and Alex A. Freitas. A survey of hierarchical classification across different application domains. *Data Min. Knowl. Discov.*, 22(1-2):31–72, 2011.
26. Aixin Sun and Ee-Peng Lim. Hierarchical text classification and evaluation. In *Proceedings of the 2001 IEEE International Conference on Data Mining*, pages 521–528, 2001.
27. Kristina Toutanova, Francine Chen, Kris Popat, and Thomas Hofmann. Text classification in a hierarchical mixture model for small training sets. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 105–113, 2001.
28. Jarek Tuszynski. *caTools: Tools: moving window statistics, GIF, Base64, ROC AUC, etc.*, 2014. R package version 1.17.1.
29. Ian H Witten, Eibe Frank, Leonard E Trigg, Mark A Hall, Geoffrey Holmes, and Sally Jo Cunningham. Weka: Practical machine learning tools and techniques with java implementations. 1999.
30. Hui Yang and Jamie Callan. A metric-based framework for automatic taxonomy induction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1 - Volume 1*, pages 271–279, 2009.