Data Min Knowl Disc (2011) 22:31–72
DOI 10.1007/s10618-010-0175-9

A survey of hierarchical classification across different application domains

Carlos N. Silla Jr. · Alex A. Freitas

Received: 24 February 2009 / Accepted: 11 March 2010 / Published online: 7 April 2010

Abstract In this survey we discuss the task of hierarchical classification. The literature about this field is scattered across very different application domains and for that reason research in one domain is often done unaware of methods developed in other domains. We define what is the task of hierarchical classification and discuss why some related tasks should not be considered hierarchical classification. We also present a new perspective about some existing hierarchical classification approaches, and based on that perspective we propose a new unifying framework to classify the existing approaches. We also present a review of empirical comparisons of the existing methods reported in the literature as well as a conceptual comparison of those methods at a high level of abstraction, discussing their advantages and disadvantages.

Keywords Hierarchical classification · Tree-structured class hierarchies · DAG-structured class hierarchies

1 Introduction

A very large amount of research in the data mining, machine learning, statistical pattern recognition and related research communities has focused on flat classification problems. By flat classification problem we are referring to standard binary or multi-class classification problems. On the other hand, many important real-world classification problems are naturally cast as hierarchical classification problems, where the classes to be predicted are organized into a class hierarchy—typically a tree or a DAG

C. N. Silla Jr. (✉) · A. A. Freitas
School of Computing, University of Kent, Canterbury, UK
e-mail: cns2@kent.ac.uk

A. A. Freitas
e-mail: A.A.Freitas@kent.ac.uk

(Direct Acyclic Graph). The task of hierarchical classification, however, needs to be better defined, as it can be overlooked or confused with other tasks, which are often wrongly referred to by the same name. Moreover, the existing literature that deals with hierarchical classification problems is usually scattered across different application domains which are not strongly connected with each other. As a result, researchers in one application domain are often unaware of methods developed by researchers in another domain. Also, there seems to be no standards on how to evaluate hierarchical classification systems or even how to setup the experiments in a standard way.

The contributions of this paper are:

- To clarify what the task of hierarchical classification is, and what it is not.
- To propose a unifying framework to classify existing and novel hierarchical classification methods, as well as different types of hierarchical classification problems.
- To perform a cross-domain critical survey, in order to create a taxonomy of hierarchical classification systems, by identifying important similarities and differences between the different approaches, which are currently scattered across different application domains.
- To suggest some experimental protocols to be undertaken when performing hierarchical classification experiments, in order to have a better understanding of the results. For instance, many authors claim that some hierarchical classification methods are better than others, but they often use standard flat classification evaluation measures instead of using hierarchical evaluation measures. Also, in some cases, it is possible to overlook what would be interesting to compare, and authors often compare their hierarchical classification methods only against flat classification methods, although the use of a baseline hierarchical method is not hard to implement and would offer a more interesting experimental comparison.

This survey seems timely as different fields of research are more and more using an automated approach to deal with hierarchical information, as hierarchies (or taxonomies) are a good way to help organize vast amounts of information. The first issue that will be discussed in this paper (Sect. 2) is precisely the definition of the hierarchical classification task. After clearly defining the task, we classify the existing approaches in the literature according to three different broad types of approach, based on the underlying methods. These approaches can be classified as: flat, i.e., ignoring the class hierarchy (Sect. 3); local (Sect. 4) or global (Sect. 5). Based on the new understanding about these approaches we present a unifying framework to classify hierarchical classification methods and problems (Sect. 6). A summary, a conceptual comparison and a review of empirical comparisons reported in the literature about these three approaches is presented in Sect. 7. Section 8 presents some major applications of hierarchical classification methods; and finally in Sect. 9 we present the conclusions of this work.

2 What is hierarchical classification?

In order to learn about hierarchical classification, one might start searching for papers with the keywords “hierarchical” and “classification”; however, this might be misleading. One of the reasons for this is that, due to the popularity of SVM (Support Vector Machine) methods in the machine learning community (which were originally

developed for binary classification problems), different researchers have developed different methods to deal with multi-class classification problems. The most common are the One-Against-One and the One-Against-All schemes (Lorena and Carvalho 2004). A less known approach consists of dividing the problem in a hierarchical way where classes which are more similar to one another are grouped together into meta-classes, resulting in a Binary Hierarchical Classifier (BHC) (Kumar et al. 2002). For instance, in Chen et al. (2004) the authors modified the standard SVM, creating what they called a H-SVM (Hierarchical SVM), based on this hierarchical problem decomposition approach.

When we consider the use of meta-classes in the pattern recognition field, they are usually manually assigned, like in Koerich and Kalva (2005), where handwritten letters with the same curves in uppercase and lowercase format (e.g. "o" and "O" will be represented by the same meta-class). An automated method for the generation of meta-classes was recently proposed by Freitas et al. (2008). At first glance the use of meta-classes (and their automatic generation) seems to be related to the hierarchical problem decomposition approach, as one can view the use of meta-classes as a two-level hierarchy where leaf classes are grouped together by similarity into intermediate classes (the meta-classes). This issue is interesting and deserves further investigation, but is beyond the scope of this paper. In this paper we take the perspective that this kind of approach is not considered to be a hierarchical classification approach, because it creates new (meta-)classes on the fly, instead of using a pre-established taxonomy. In principle a classification algorithm is not supposed to create new classes, which is related to clustering.

In this paper we are interested in approaches that cope with a pre-defined class hierarchy, instead of creating one from the similarity of classes within data (which would lead to higher-level classes that could be meaningless to the user). Let us elaborate on this point. There are application domains where the internal (non-leaf) nodes of the class hierarchy can be chosen based on data (usually in the text mining application domain), like in Sasaki and Kita (1998), Punera et al. (2005), Li et al. (2007), Hao et al. (2007), where they build the hierarchy during training by using some sort of hierarchical clustering method, and then classify new test examples by using a hierarchical approach. However, in other domains, like protein function prediction in bioinformatics, just knowing that classes A and B are similar can be misleading, as proteins with similar characteristics (sequences of amino acids) can have very different functions and vice-versa (Gerlt and Babbitt 2000). Therefore, in this work, we are interested only in hierarchical classification (a type of supervised learning). Hierarchical clustering (a type of unsupervised learning) is out of the scope of the paper.

Hierarchical classification can also appear under the name of Structured Classification (Seeger 2008; Astikainen et al. 2008). However, the research field of structured classification involves many different types of problems which are not hierarchical classification problems, e.g. Label Sequence Learning (Altun and Hofmann 2003; Tsochantaridis et al. 2005). Therefore, hierarchical classification can be seen as a particular type of structured classification problem, where the output of the classification algorithm is defined over a class taxonomy; whilst the term structured classification is broader and denotes a classification problem where there is some structure (hierarchical or not) among the classes.

It is important then to define what exactly is a class taxonomy. Wu et al. (2005) have defined a class taxonomy as a tree structured regular concept hierarchy defined over a partially order set $(C, \prec)$, where C is a finite set that enumerates all class concepts in the application domain, and the relation $\prec$ represents the "IS-A" relationship. Wu et al. (2005) define the "IS-A" relationship as both anti-reflexive and transitive. However, we prefer to define the "IS-A" relationship as asymmetric, anti-reflexive and transitive:

- The only one greatest element "R" is the root of the tree.
- $\forall c_i, c_j \in C, if\ c_i \prec c_j\ then\ c_j \not\prec c_i$.
- $\forall c_i \in C, c_i \not\prec c_i$.
- $\forall c_i, c_j, c_k \in C, c_i \prec c_j\ and\ c_j \prec c_k\ imply\ c_i \prec c_k$.

This definition, although originally proposed for tree structured class taxonomies, can be used to define DAG structured class taxonomies as well. Ruiz and Srinivasan (2002) give a good example of the asymmetric and transitive relations: The "IS-A" relation is asymmetric (e.g. all dogs are animals, but not all animals are dogs) and transitive (e.g., all pines are evergreens, and all evergreens are trees; therefore all pines are trees).

Note that, for the purposes of this survey, any classification problem with a class structure satisfying the aforementioned four properties of the IS-A hierarchy can be considered as a hierarchical classification problem, and in general the hierarchical classification methods surveyed in this work assume (explicitly or implicitly) the underlying class structure satisfies those problems. In the vast majority of works on hierarchical classification, the actual class hierarchy in the underlying problem domain can indeed be called a IS-A hierarchy from a semantical point of view. However, in a few cases the semantics of the underlying class hierarchy might be different, but as long as the aforementioned four properties are satisfied, we would consider the target problem as a hierarchical classification one. For instance, the class taxonomy associated with cellular localization in the Gene Ontology (an ontology which is briefly discussed in Sect. 8.2) is essentially, from a semantical point of view, a PART-OF class hierarchy, but it still satisfies the four properties of the aforementioned definition of a IS-A hierarchy, so we consider the prediction of cellular location classes according to that class hierarchy as a hierarchical classification problem.

Whether the taxonomy is organized into a tree or a DAG influences the degree of difficulty of the underlying hierarchical classification problem. Notably, as it will be seen in Sect. 7, most of the current literature focus on working with trees as it is an easier problem. One of the main contributions of this survey is to organize the existing hierarchical classification approaches into a taxonomy, based on their essential properties, regardless of the application domain. One of the main problems, in order to do this, is to deal with all the different terminology that has already been proposed, which is often inconsistent across different works. In order to understand these essential properties, is important to clarify a few aspects of hierarchical classification methods.

Let us consider initially two types of conventional classification methods that cannot directly cope with hierarchical classes: binary and multi-class classifiers. First, the main difference between a binary classifier and a multi-class classifier is that the binary classifier can only handle two-class problems, whilst a multi-class clas-

sifier can handle in principle any number of classes. Secondly, there are multi-class classifiers that can also be multi-label, i.e. the answer from the classifier can be more than one class assigned to a given example. Thirdly, since these types of classifiers were not designed to deal with hierarchical classification problems, they will be referred to as flat classification algorithms. Fourthly, in the context of hierarchical classification most approaches could be called multi-label. For instance, considering the hierarchical class structure presented in Fig. 1 (where R denotes the root node), if the output of a classifier is class 2.1.1, it is natural to say that it also belongs to classes 2 and 2.1, therefore having three classes as the output of the classifier. In Tikk et al. (2004) this notion of multi-label is used and they call this a particular type of multi-label classification problem. However, since this definition is trivial, as any hierarchical approach could be considered multi-label in this sense, in this work we will only consider a hierarchical classifier to be hierarchically multi-label if it can assign more than one class at any given level of the hierarchy to a given example. This distinction is particularly important, as a hierarchically multi-label classification algorithm is more challenging to design than a hierarchically single-label one. Also, recall that in hierarchical classification we assume that the relation between a node and its parent in the class hierarchy is a "IS-A" relationship.

According to Freitas and de Carvalho (2007) and Sun and Lim (2001) hierarchical classification methods differ in a number of criteria. The first criterion is the type of hierarchical structure used. This structure is based on the problem structure and it typically is either a tree or a DAG. Figure 2 illustrates these two types of structures.

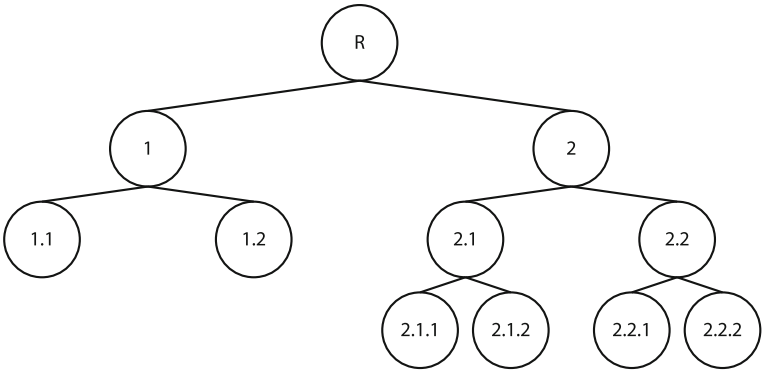

Fig. 1 An example of a tree-based hierarchical class structure

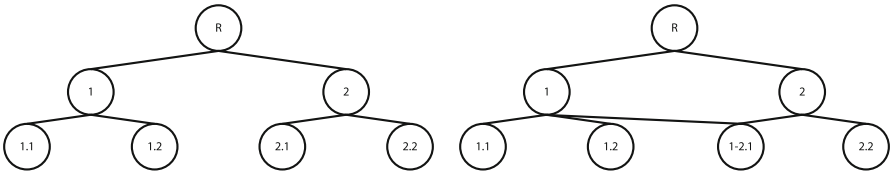

Fig. 2 A simple example of a tree structure (*left*) and a DAG structure (*right*)

The main difference between them is that in the DAG a node can have more than one parent node.

The second criterion is related to how deep the classification in the hierarchy is performed. That is, the hierarchical classification method can be implemented in a way that will always classify a leaf node [which Freitas and de Carvalho (2007) refer to as mandatory leaf-node prediction (MLNP) and Sun and Lim (2001) refer to as virtual category tree] or the method can consider stopping the classification at any node in any level of the hierarchy [which Freitas and de Carvalho (2007) refer to as non-mandatory leaf node prediction and Sun and Lim (2001) refer to as category tree]. In this paper we will use the term (non-)mandatory leaf node prediction, which can be naturally used for both tree-structured and DAG-structured class taxonomies.

The third criterion is related to how the hierarchical structure is explored. The current literature often refers to top-down (or local) classifiers, when the system employs a set of local classifiers; big-bang (or global) classifiers, when a single classifier coping with the entire class hierarchy is used; or flat classifiers, which ignore the class relationships, typically predicting only the leaf nodes. However, a closer look at the existing hierarchical classification methods reveals that:

1. The top-down approach is not a full hierarchical classification approach by itself, but rather a method for avoiding or correcting inconsistencies in class prediction at different levels, during the testing (rather than training) phase;
2. There are different ways of using local information to create local classifiers, and although most of them are referred to as top-down in the literature, they are very different during the training phase and slightly different in the test phase;
3. Big-bang (or global) classifiers are trained by considering the entire class hierarchy at once, and hence they lack the kind of modularity for local training of the classifier that is a core characteristic of the local classifier approach.

These are the main points which will be discussed in detail in the next four sections.

3 Flat classification approach

The flat classification approach, which is the simplest one to deal with hierarchical classification problems, consists of completely ignoring the class hierarchy, typically predicting only classes at the leaf nodes. This approach behaves like a traditional classification algorithm during training and testing. However, it provides an indirect solution to the problem of hierarchical classification, because, when a leaf class is assigned to an example, one can consider that all its ancestor classes are also implicitly assigned to that instance (recall that we assume a "IS-A" class hierarchy).

However, this very simple approach has the serious disadvantage of having to build a classifier to discriminate among a large number of classes (all leaf classes), without exploring information about parent-child class relationships present in the class hierarchy. Figure 3 illustrates this approach. We use here the term flat classification approach, as it seems to be the most commonly used term in the existing literature, although in Burred and Lerch (2003) the authors refer to this approach as "the direct approach", while in Xiao et al. (2007) this approach is referred to as a "global classifier"—which

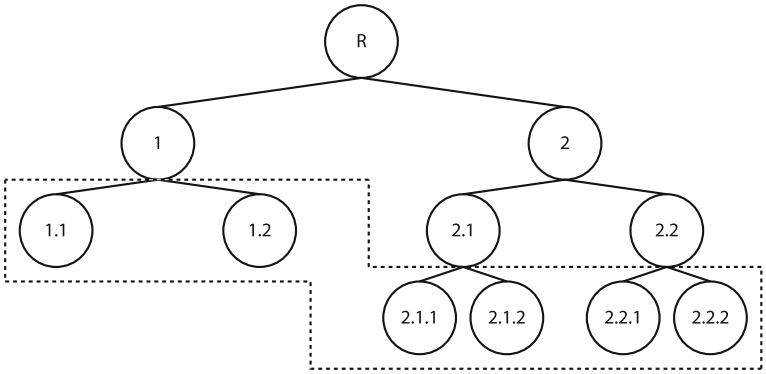

Fig. 3 Flat classification approach using a flat multi-class classification algorithm to always predict the leaf nodes

is misleading as they are referring to this naïve flat classification algorithm, and the term global classifier is often used to refer to the "big-bang" approach (Sect. 5).

In Barbedo and Lopes (2007) the authors refer to this approach as a "bottom-up" approach. They justify this term as follows: "The signal is firstly classified according to the basic genres, and the corresponding upper classes are consequences of this first classification (bottom-up approach)." In this paper, however, we prefer to use the term flat classification to be consistent with the majority of the literature.

Considering the different types of class taxonomies (tree or DAG), this approach can cope with both of them as long as the problem is a mandatory-leaf node prediction problem, as it is incapable of handling non-mandatory leaf node prediction problems. In this approach training and testing proceed in the same way as in standard (non-hierarchical) classification algorithms.

4 Local classifier approaches

In the seminal work of Koller and Sahami (1997), the first type of local classifier approach (also known as top-down approach in the literature) was proposed. From this work onwards, many different authors used augmented versions of this approach to deal with hierarchical classification problems. However, the important aspect here is not that the approach is top-down (as it is commonly called), but rather that the hierarchy is taken into account by using a local information perspective. The idea behind this reasoning is that in the literature there are several papers that employ this local information in different ways. These approaches, therefore, can be grouped based on how they use this local information and how they build their classifiers around it. More precisely, there seems to exist three standard ways of using the local information: a local classifier per node (LCN), a local classifier per parent node (LCPN) and a local classifier per level (LCL). In the following subsections we discuss each one of them in detail. Also note that unless specified otherwise, the discussion will assume a single label tree-structured class hierarchy and mandatory leaf node prediction.

It should be noted that, although the three types of local hierarchical classification algorithms discussed in the next three sub-sections differ significantly in their training phase, they share a very similar top-down approach in their testing phase. In essence, in this top-down approach, for each new example in the test set, the system first predicts its first-level (most generic) class, then it uses that predicted class to narrow the choices of classes to be predicted at the second level (the only valid candidate second-level classes are the children of the class predicted at the first level), and so on, recursively, until the most specific prediction is made.

As a result, a disadvantage of the top-down class-prediction approach (which is shared by all the three types of local classifiers discussed next) is that an error at a certain class level is going to be propagated downwards the hierarchy, unless some procedure for avoiding this problem is used. If the problem is non-mandatory leaf node prediction, a blocking approach (where an example is passed down to the next lower level only if the confidence on the prediction at the current level is greater than a threshold) can avoid that misclassifications are propagated downwards, at the expense of providing the user with less specific (less useful) class predictions. Some authors use methods to give better estimates of class probabilities, like shrinkage (McCallum et al. 1998) and isotonic smoothing (Punera and Ghosh 2008). The issues of non-mandatory leaf node prediction and blocking are discussed in Sect. 4.4.

4.1 Local classifier per node approach

This is by far the most used approach in the literature. It often appears under the name of a top-down approach, but as we mentioned earlier, we shall see why this is not a good name as the top-down approach is essentially a method to avoid inconsistencies in class predictions at different levels in the class hierarchy. The LCN approach consists of training one binary classifier for each node of the class hierarchy (except the root node). Figure 4 illustrates this approach.

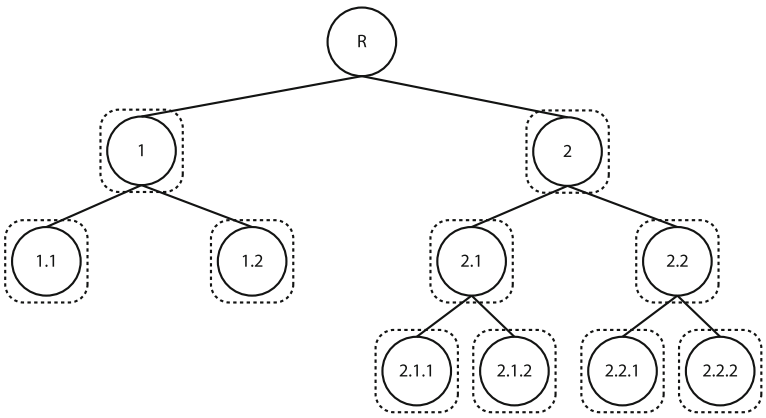

Fig. 4 Local classifier per node approach (*circles* represent classes and *dashed squares* with round corners represent binary classifiers)

Table 1 Notation for negative and positive training examples

Symbol	Meaning
Tr	The set of all training examples
$Tr^+(c_j)$	The set of positive training examples of c_j
$Tr^-(c_j)$	The set of negative training examples of c_j
$\uparrow(c_j)$	The parent category of c_j
$\downarrow(c_j)$	The set of children categories of c_j
$\Uparrow(c_j)$	The set of ancestor categories of c_j
$\Downarrow(c_j)$	The set of descendant categories of c_j
$\leftrightarrow(c_j)$	The set of sibling categories of c_j
$*(c_j)$	Denotes examples whose most specific known class is c_j

There are different ways to define the set of positive and negative examples for training the binary classifiers. In the literature most works often use one approach and studies like Eisner et al. (2005) and Fagni and Sebastiani (2007) where different approaches are compared are not common. In the work of Eisner et al. (2005) the authors identify and experiment with four different policies to defining the set of positive and negative examples. In Fagni and Sebastiani (2007) the authors focus on the selection of the negative examples and empirically compare four policies (two standard ones compared with two novel ones). However the novel approaches are limited to text categorization problems and achieved similar results to the standard approaches; and for that reason they are not further discussed in this paper. The notation used to define the sets of positive and negative examples is based on the one used in Fagni and Sebastiani (2007) and is presented in Table 1.

- The "exclusive" policy [as defined by Eisner et al. (2005)]: $Tr^+(c_j) = *(c_j)$ and $Tr^-(c_j) = Tr \setminus *(c_j)$. This means that only examples explicitly labeled as c_j as their most specific class are selected as positive examples, while everything else is used as negative examples. For example, using Fig. 4, for $c_j = 2.1$, $Tr^+(c_{2.1})$ consists of all examples whose most specific class is 2.1; and $Tr^-(c_{2.1})$ consists of the set of examples whose most specific class is 1, 1.1, 1.2, 2, 2.1.1, 2.1.2, 2.2, 2.2.1 or 2.2.2. This approach has a few problems. First, it does not consider the hierarchy to create the local training sets. Second, it is limited to problems where partial depth labeling instances are available. By partial depth labeling instances we mean instances whose class label is known just for shallower levels of the hierarchy, and not for deeper levels. Third, using the descendant nodes of c_j as negative examples seems counter-intuitive considering that examples who belong to class $\Downarrow(c_j)$ also implicitly belong to class c_j according to the "IS-A" hierarchy concept.
- The "less exclusive" policy [as defined by Eisner et al. (2005)]: $Tr^+(c_j) = *(c_j)$ and $Tr^-(c_j) = Tr \setminus *(c_j) \cup \Downarrow(c_j)$. In this case, using Fig. 4 as example, $Tr^+(c_{2.1})$ consists of the set of examples whose most specific class is 2.1; and $Tr^-(c_{2.1})$ consists of the set of examples whose most specific class is 1, 1.1, 1.2, 2, 2.2, 2.2.1 or 2.2.2. This approach avoids the aforementioned first and third

problems of the exclusive policy, but it is still limited to problems where partial depth labeling instances are available.

- The "less inclusive" policy [as defined by Eisner et al. (2005), it is the same as the "ALL" policy defined by Fagni and Sebastiani (2007)]: $Tr^+(c_j) = *(c_j) \cup \Downarrow (c_j)$ and $Tr^-(c_j) = Tr \setminus *(c_j) \cup \Downarrow (c_j)$. In this case $Tr^+(c_{2.1})$ consists of the set of examples whose most specific class is 2.1, 2.1.1 or 2.1.2; and $Tr^-(c_{2.1})$ consists of the set of examples whose most specific class is 1, 1.1, 1.2, 2, 2.2, 2.2.1 or 2.2.2.
- The "inclusive" policy [as defined by Eisner et al. (2005)]: $Tr^+(c_j) = *(c_j) \cup \Downarrow (c_j)$ and $Tr^-(c_j) = Tr \setminus *(c_j) \cup \Downarrow (c_j) \cup \Uparrow (c_j)$. In this case $Tr^+(c_{2.1})$ is the set of examples whose most specific class is 2.1, 2.1.1 or 2.1.2; and $Tr^-(c_{2.1})$ consists of the set of examples whose most specific class is 1, 1.1, 1.2, 2.2, 2.2.1 or 2.2.2.
- The "siblings" policy [as defined by Fagni and Sebastiani (2007), and which Ceci and Malerba (2007) refers to as "hierarchical training sets"]: $Tr^+(c_j) = *(c_j) \cup \Downarrow (c_j)$ and $Tr^-(c_j) = \leftrightarrow (c_j) \cup \Downarrow (\leftrightarrow (c_j))$. In this case $Tr^+(c_{2.1})$ consists of the set of examples whose most specific class is 2.1, 2.1.1 or 2.1.2; and $Tr^-(c_{2.1})$ consists of the set of examples whose most specific class is 2.2, 2.2.1, 2.2.2.
- The "exclusive siblings" policy [as defined by Ceci and Malerba (2007) and referred to as "proper training sets"]: $Tr^+(c_j) = *(c_j)$ and $Tr^-(c_j) = \leftrightarrow (c_j)$. In this case $Tr^+(c_{2.1})$ consists of the set of examples whose most specific class is 2.1; and $Tr^-(c_{2.1})$ consists of the set of examples whose most specific class is 2.2.

It should be noted that in the aforementioned policies for negative and positive training examples, we have assumed that the policies defined in Fagni and Sebastiani (2007) follow the usual approach of using as positive training examples all the examples belonging to the current class node ($*(c_j)$) and all of its descendant classes ($\Downarrow (c_j)$). Although this is the most common approach, several other approaches can be used, as shown by Eisner et al. (2005). In particular, the exclusive and less exclusive policies use as positive examples only the examples whose most specific class is the current class, without using the examples whose most specific class is a descendant from the current class in the hierarchy. It should be noted that the aim of the work of Eisner et al. (2005) was to evaluate different ways of creating the positive and negative training sets for predicting functions based on the Gene Ontology, but it seems that they overlooked the use of the siblings policy which is common in the hierarchical text classification domain. Given the above discussion, one can see that it is important that authors be clear on how they select both positive and negative examples in the local hierarchical classification approach, since so many ways of defining positive and negative examples are possible, with subtle differences between some of them.

Concerning which approach one should use, Eisner et al. (2005) note that as the classifier becomes more inclusive (with more positive training examples) the classifiers perform better. Their results (using F-measure as a measure of performance) comparing the different measures are: Exclusive: 0.456, Less exclusive: 0.528, Less inclusive: 0.696 and Inclusive: 0.697. In the experiments of Fagni and Sebastiani (2007), where they compare the siblings and less-inclusive policies, concerning predictive accuracy there is no clear winner. However, they note that the siblings policy uses considerably less data in comparison with the less-inclusive policy, and since they have the same

accuracy, that is the one that should be used. In any case, more research, involving a wider variety of datasets, would be useful to better characterise the relative strengths and weakness of the aforementioned different policies in practice.

During the testing phase, regardless of how positive and negative examples were defined, the output of each binary classifier will be a prediction indicating whether or not a given test example belongs to the classifier's predicted class. One advantage of this approach is that it is naturally multi-label in the sense that it is possible to predict multiple labels per class level, in the case of multi-label problems. Such a natural multi-label prediction is achieved using just conventional single-label classification algorithms, avoiding the complexities associated with the design of a multi-label classification algorithm (Tsoumakas and Katakis 2007). In the case of single-label (per level) problems one can enforce the prediction of a single class label per level by assigning to a new test example just the class predicted with the greatest confidence among all classifiers at a given level—assuming classifiers output a confidence measure of their prediction. This approach has, however, a disadvantage. Considering the example of Fig. 4 it would be possible, using this approach, to have an output like class 1 = false and class 1.2 = true (since the classifiers for nodes 1 and 1.2 are independently trained), which leads to an inconsistency in class predictions across different levels. Therefore, if no inconsistency correction method is taken into account, this approach is going to be prone to class-membership inconsistency.

As mentioned earlier, one of the current misconceptions in the literature is the confusion between local information-based training of classifiers and the top-down approach for class prediction (in the testing phase). Although they are often used together, the local information-based training approach is not necessarily coupled with the top-down approach, as a number of different inconsistency correction methods can be used to avoid class-membership inconsistency during the test phase. Let us now review the existing inconsistency correction methods for the LCN approach.

The class-prediction top-down approach seems to have been originally proposed by Koller and Sahami (1997), and its essential characteristic is that it consists of performing the *testing* phase in a top-down fashion, as follows. For each level of the hierarchy (except the top level), the decision about which class is predicted at the current level is based on the class predicted at the previous (parent) level. For example, at the top level, suppose the output of the local classifier for class 1 is *true*, and the output of the local classifier for class 2 is *false*. At the next level, the system will only consider the output of classifiers predicting classes which are children of class 1. Originally, the class-prediction top-down method was forced to always predict a leaf node (Koller and Sahami 1997). When considering a non-mandatory leaf-node prediction (NMLNP) problem, the class-prediction top-down approach has to use a stopping criterion that allows an example to be classified just up to a non-leaf class node. This extension might lead to the *blocking* problem, which will be discussed in Sect. 4.4.

Besides the class-prediction top-down approach, other methods were proposed to deal with inconsistencies generated by the LCN approach. One such method consists of stopping the classification once the binary classifier for a given node gives the answer that the unseen example does not belong to that class. For example, if the output for the binary classifier of class 2 is *true*, and the outputs of the binary classifiers

for classes 2.1 and 2.2 are *false*, then this approach would ignore the answer of all the lower level classifiers predicting classes that are descendant of classes 2.1 and 2.2 and output the class 2 to the user. By doing this, the class predictions respect the hierarchy constraints. This approach was proposed by Wu et al. (2005) and was referred to as "Binarized Structured Label Learning" (BSLL).

In Dumais and Chen (2000) the authors propose two class-membership inconsistency correction methods based on thresholds. In order for a class to be assigned to a test example, the probabilities for the predicted class were used. In the first method, they use a boolean condition where the posterior probability of the classes at the first and second levels must be higher than a user specified threshold, in the case of a two-level class hierarchy. The second method uses a multiplicative threshold that takes into account the product of the posterior probability of the classes at the first and second levels. For example, let us suppose that, for a given test example, the posterior probability for each class in the first two levels in Fig. 4 were: $p(c_1) = 0.6$, $p(c_2) = 0.2$, $p(c_{1.1}) = 0.55$, $p(c_{1.2}) = 0.1$, $p(c_{2.1}) = 0.2$, $p(c_{2.2}) = 0.3$. Considering a threshold of 0.5, by using the boolean rule the classes predicted for that test example would be class 1 and class 1.1 as both classes have a posterior probability higher than 0.5. By using the multiplicative threshold, the example would be assigned to class 1 but not class 1.1, as the posterior probability of class 1 $\times$ the posterior probability of class 1.1 is 0.33, which is below the multiplicative threshold of 0.5.

In the work of Barutcuoglu and DeCoro (2006), Barutcuoglu et al. (2006), DeCoro et al. (2007) another class-membership inconsistency correction method for the LCN approach is proposed. Their method is based on a Bayesian aggregation of the output of the base binary classifiers. The method takes the class hierarchy into account by transforming the hierarchical structure of the classes into a Bayesian network. In Barutcuoglu and DeCoro (2006) two baseline methods for conflict resolution are proposed: the first method propagates negative predictions downward (i.e. the negative prediction at any class node is used to overwrite the positive predictions of its descendant nodes) while the second baseline method propagates the positive predictions upward (i.e. the positive prediction at any class node is used to overwrite the negative predictions of all its ancestors). Note that the first baseline method is the same as the BSLL.

Another approach for class-membership inconsistency correction based on the output of all classifiers has been proposed by Valentini (2009), where the basic idea is that by evaluating all the classifier nodes' outputs it is possible to make consistent predictions by computing a "consensus" probability using a bottom-up algorithm.

Xue et al. (2008) propose a strategy based on pruning the original hierarchy. The basic idea is that when a new document is going to be classified it can possibly be related to just some of the many hierarchical classification classes. Therefore, in order to reduce the error of the top-down class-prediction approach, their method first computes the similarity between the new document and all other documents, and creates a pruned class hierarchy which is then used in a second stage to classify the document using a top-down class-prediction approach.

Bennett and Nguyen (2009) propose a technique called expert refinements. The refinement consists of using cross-validation in the training phase to obtain a better estimation of the true probabilities of the predicted classes. The refinement technique

is then combined with a bottom-up training approach, which consists of training the leaf classifiers using refinement and passing this information to the parent classifiers.

So far we have discussed the LCN approach mainly in the context of a single label (per level) problem with a tree-structured class hierarchy. In the multi-label hierarchical classification scenario, this approach is still directly employable, but some more sophisticated method to cope with the different outputs of the classifiers should be used. For example, in Esuli et al. (2008) the authors propose the TreeBoost.MH which uses during training at each classification node the AdaBoost.MH base learner. Their approach can also (optionally) perform feature selection by using information from the sibling classes. In the context of a DAG, the LCN approach can still be used in a natural way as well, as it has been done in Jin et al. (2008) and Otero et al. (2009).

4.2 Local classifier per parent node approach

Another type of local information that can be used, and it is also often referred to as top-down approach in the literature, is the approach where, for each parent node in the class hierarchy, a multi-class classifier (or a problem decomposition approach with binary classifiers like One-Against-One scheme for Binary SVMs) is trained to distinguish between its child nodes. Figure 5 illustrates this approach.

In order to train the classifiers the "siblings" policy, as well as the "exclusive siblings" policy, both presented in Sect. 4.1, are suitable to be used.

During the testing phase, this approach is often coupled with the top-down class prediction approach, but this coupling is not necessarily a must, as new class prediction approaches for this type of local approach could be developed. Consider the top-down class-prediction approach and the same class tree example of Fig. 5, and suppose that the first level classifier assigns the example to the class 2. The second level classifier, which was only trained with the children of the class node 2, in this case

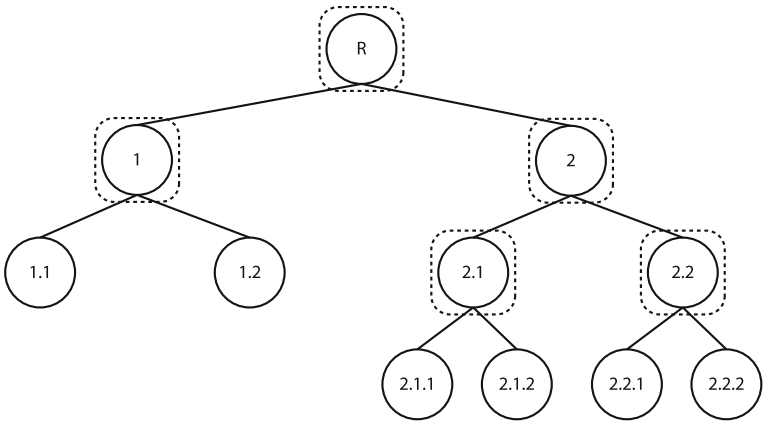

Fig. 5 Local classifier per parent node (*circles* represent classes and *dashed squares* with round corners in parent nodes represent multi-class classifiers—predicting their child classes)

2.1 and 2.2, will then make its class assignment (and so on, if deeper-level classifiers were available), therefore avoiding the problem of making inconsistent predictions and respecting the natural constrains of class membership.

An extension of this type of local approach known as the "selective classifier" approach was proposed by Secker et al. (2007). The authors refer to this method as the Selective Top-Down approach, but it is here re-named to "selective classifier" approach to emphasize that what are being selected are the classifiers, rather than attributes as in attribute (feature) selection methods. In addition, we prefer to reserve the term "top-down" to the class prediction method during the testing phase, as explained earlier. Usually, in the LCPN approach the same classification algorithm is used throughout all the class hierarchy. In Secker et al. (2007), the authors hypothesise that it would be possible to improve the predictive accuracy of the LCPN approach by using different classification algorithms at different parent nodes of the class hierarchy. In order to determine which classifier should be used at each node of the class hierarchy, during the training phase, the training set is split into a sub-training and validation set with examples being assigned randomly to each of those datasets. Different classifiers are trained using that sub-training set and are then evaluated on the validation set. The classifier chosen for each parent class node is the one with the highest classification accuracy on the validation set. An improvement over the selective classifier approach was proposed by Holden and Freitas (2008), where a swarm intelligence optimization algorithm was used to perform the classifier selection. The motivation behind this approach is that the original selective classifier approach uses a greedy, local search method that has only a limited local view of the training data when selecting a classifier, while the swarm intelligence algorithm performs a global search that considers the entire tree of classifiers (having a complete view of the training data) at once. Another improvement over the selective classifier approach was proposed by Silla Jr and Freitas (2009b), where both the best classifier and the best type of example representation (out of a few types of representations, involving different kinds of predictor attributes) are selected for each parent node classifier. In addition, Secker et al. (2010) extended their previous classifier-selection approach in order to select both classifiers and attributes at each classifier node.

So far we have discussed the LCPN approach in the context of a single label problem with a tree-structured class hierarchy. Let us now briefly discuss this approach in the context of a multi-label problem. In this multi-label scenario, this approach is not directly employable. There are, at least, two approaches that could be used to cope with the multi-label scenario. One is to use a multi-label classifier at each parent node, as done by Wu et al. (2005). The second approach is to take into account the different confidence scores provided by each classifier and have some kind of decision thresholds based on those scores to allow multiple labels. One way of doing this would be to adapt the multiplicative threshold proposed by Dumais and Chen (2000). When dealing with a DAG-structured class hierarchy, this approach is also not directly employable, as the created local training sets might be highly redundant (due to the fact that a given class node can have multiple parents, which can be located at different depths). To the best of our knowledge this approach has not yet been used with DAG-structured class hierarchies.

4.3 Local classifier per level approach

This is the type of "local" (broadly speaking) classifier approach least used so far on the literature. The local classifier per level approach consists of training one multi-class classifier for each level of the class hierarchy. Figure 6 illustrates this approach. Considering the example of Fig. 6, three classifiers would be trained, one classifier for each class level, where each classifier would be trained to predict one or more classes (depending on whether the problem is single-label or multi-label) at its corresponding class level. The creation of the training sets here is implemented in the same way as in the local classifier per parent node approach.

This approach has been mentioned as a possible approach by Freitas and de Carvalho (2007), but to the best of our knowledge its use has been limited as a baseline comparison method in Clare and King (2003) and Costa et al. (2007b).

One possible (although very naïve) way of classifying test examples using classifiers trained by this approach is as follows. When a new test example is presented to the classifier, get the output of all classifiers (one classifier per level) and use this information as the final classification. The major drawback of this class-prediction approach is being prone to class-membership inconsistency. By training different classifiers for each level of the hierarchy it is possible to have outputs like class 2 at the first level, class 1.2 at the second level, and class 2.2.1 at the third level, therefore generating inconsistency. Hence, if this approach is used, it should be complemented by a post-processing method that tries to correct the prediction inconsistency.

To avoid this problem, one approach that can be used is the class-prediction top-down approach. In this context, the classification of a new test example would be done in a top-down fashion (similar to the standard top-down class-prediction approach), restricting the possible classification output at a given level only to the child nodes of the class node predicted in the previous level (in the same way as it is done in the LCPN approach).

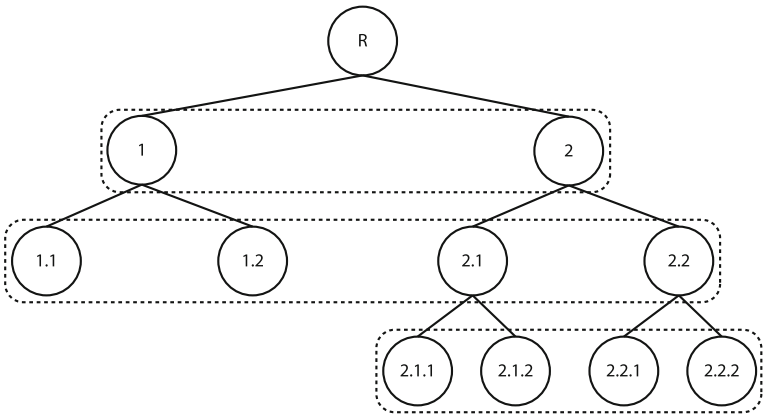

Fig. 6 Local classifier per level (*circles* represent classes and each *dashed rectangle* with round corners encloses the classes predicted by a multi-class classifier)

This approach could work with either a tree or a DAG class structure. Although depth is normally a tree concept, it could still be computed in the context of a DAG, but in the latter case this approach would be considerably more complex. This is because, since there can be more than one path between two nodes in a DAG, a class node can be considered as belonging to several class levels, and so there would be considerable redundancy between classifiers at different levels. In the context of a tree structured class hierarchy and multi-label problem, methods based on confidence scores or posterior probabilities could be used to make more than one prediction per class level.

4.4 Non-mandatory leaf node prediction and the blocking problem

In the previous sections, we discussed the different types of local classifiers but we avoided the discussion of the non-mandatory leaf node prediction problem. The non-mandatory leaf node prediction problem, as the name implies, allows the most specific class predicted to any given instance to be a class at any node (i.e. internal or leaf node) of the class hierarchy, and was introduced by Sun and Lim (2001). A simple way to deal with the NMLNP problem is to use a threshold at each class node, and if the confidence score or posterior probability of the classifier at a given class node—for a given test example—is lower than this threshold, the classification stops for that example. A method for automatically computing these thresholds was proposed by Ceci and Malerba (2007).

The use of thresholds can lead to what Sun et al. (2004) called the *blocking* problem. As briefly mentioned in Sect. 4.1, *blocking* occurs when, during the top-down process of classification of a test example, the classifier at a certain level in the class hierarchy predicts that the example in question does not have the class associated with that classifier. In this case the classification of the example will be "blocked", i.e., the example will not be passed to the descendants of that classifier. For instance, in Fig. 1 blocking could occur, say, at class node 2, which would mean that the example would not be passed to the classifiers that are descendants of that node.

Three strategies to avoid blocking are discussed by Sun et al. (2004): threshold reduction method, restricted voting method and extended multiplicative thresholds. These strategies were originally proposed to work together with two binary classifiers at each class node. The first classifier (which they call local classifier) determines if an example belongs to the current class node, while the second classifier (which they call sub-tree classifier) determines whether the example is going to be given to the current node's child-node classifiers or if the system should stop the classification of that example at the current node.

These blocking reduction methods work as follows:

– *Threshold reduction method*: This method consists of lowering the thresholds of the subtree classifiers. The idea behind this approach is that by reducing the thresholds this will allow more examples to be passed to the classifiers at lower levels. The challenge associated with this approach is how to determine the threshold value of each subtree classifier. This method can be easily used with both tree-structured and DAG-structured class hierarchies.

– *Restricted voting*: This method consists of creating a set of secondary classifiers that will link a node and its grandparent node. The motivation for this approach is that, although the threshold reduction method is able to pass more examples to the classifiers at the lower levels, it is still possible to have examples wrongly rejected by the high-level subtree classifiers. Therefore, the restricted voting approach gives the low-level classifiers a chance to access these examples before they are rejected. This approach is motivated by ensemble-based approaches and the set of secondary classifiers are trained with a different training set than the original subtree classifiers. This method was originally designed for tree-structured class hierarchies and extending it to DAG-structured hierarchies would make it considerably more complex and more computationally expensive, as in a DAG-structured class hierarchy each node might have multiple parent nodes.
– *Extended multiplicative thresholds*: This method is a straightforward extension of the multiplicative threshold proposed by Dumais and Chen (2000) (explained in Sect. 4.1), which originally only worked for a 2-level hierarchy. The extension consists simply of establishing thresholds recursively for every two levels.

5 Global classifier (or big-bang) approach

Although the problem of hierarchical classification can be tackled by using the previously described local approaches, learning a single global model for all classes has the advantage that the total size of the global classification model is typically considerably smaller, by comparison with the total size of all the local models learned by any of the local classifier approaches. In addition, dependencies between different classes with respect to class membership (e.g. any example belonging to class 2.1 automatically belongs to class 2) can be taken into account in a natural, straightforward way, and may even be explicitated (Blockeel et al. 2002). This kind of approach is known as the big-bang approach, also called "global" learning. Figure 7 illustrates this approach.

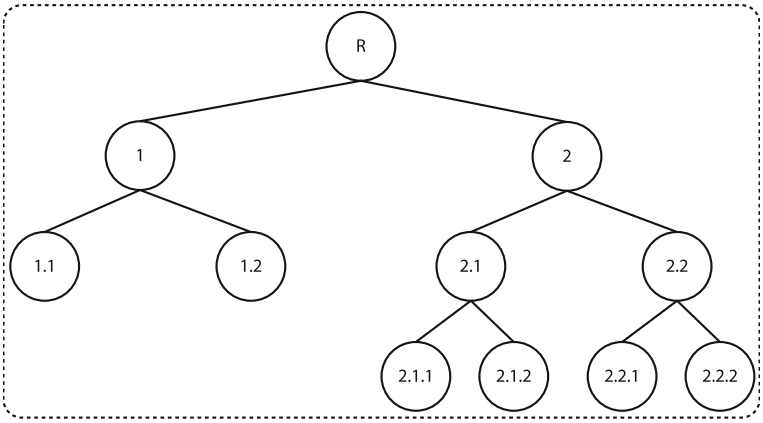

Fig. 7 Big-bang classification approach using a classification algorithm that learns a global classification model about the whole class hierarchy

In the global classifier approach, a single (relatively complex) classification model is built from the training set, taking into account the class hierarchy as a whole during a single run of the classification algorithm. When used during the test phase, each test example is classified by the induced model, a process that can assign classes at potentially every level of the hierarchy to the test example (Freitas and de Carvalho 2007).

Originally in the work of Sun and Lim (2001) the authors stated that there were two approaches to hierarchical classification: top-down and big-bang. This statement has been followed by several works in the field up to the writing of this paper (Costa et al. 2007b; Secker et al. 2007; Alves et al. 2008; Xue et al. 2008). Based on the new perspective about the top-down approach discussed earlier, this approach is essentially a strategy for avoiding class-prediction inconsistencies across class levels during the testing (rather than training) phase, when using a local hierarchical classification method. However, we still lack a clear definition for the big-bang approach, as such definition has not been made so far. Usually, by a mutual exclusion criterion, any hierarchical classification method not considered top-down has been called big-bang.

Therefore, one of the contributions of this survey is clarifying what kinds of approaches can be considered as global classifier approaches. Compared to the local classifier approaches, much less research has been reported using the global classifier approach. Although the latter approach has the advantage of learning, during the training phase, a global model for all the classes in a single step, it has an added complexity to it.

Although there seems to be no specific core characteristic shared by all global classifier approaches, in general global classifiers have two related broad characteristics, as follows. They consider the entire class hierarchy at once (as mentioned earlier) and they lack the kind of modularity for local training of the classifier that is a core characteristic of the local classifier approach. We emphasize that the crucial distinction between the global (big-bang) and local classifier approaches is in the training phase. A global classifier can even use a top-down approach (typically used by local classifier approaches) in its testing phase. In this latter case, as long as the classifier's training phase follows the two aforementioned broad characteristics of the global approach, it should be classified as a global (rather than local) classifier.

From the current state of the art, the main kinds of approaches that are usually considered to be global approaches are as follows. First, there is an approach based on the Rocchio classifier (Rocchio 1971). This approach uses the idea of class clusters, where new examples are assigned to the nearest class by computing the distance between the new test example and each class.

One example of this approach is found in Labrou and Finin (1999). In this work the system classifies web pages into a subset of the Yahoo! hierarchical categories. This method is specific to text mining applications. During the testing phase each new document has its similarity computed with respect to each document topic. The final classification is given based on some threshold.

Another type of global classifiers is based on casting the hierarchical class problem as a multi-label classification problem (Kiritchenko et al. 2005, 2006). In order to be able to predict any class in the hierarchy, the training phase is modified to take into account all the classes in the hierarchy, by augmenting the non-leaf nodes with the

information of its ancestor classes. During the test phase, since the algorithm does not take the hierarchy into account, it may suffer from the same limitations of the LCN, that is, it is prone to class-prediction inconsistency. For this reason, in the approach of Kiritchenko et al. (2005, 2006), the authors have a post-processing step, which takes all the outputs into account in order to ensure that the hierarchical constrains are respected.

Another type of global classifiers consists of modifying existing classifiers to directly cope with the class hierarchy and benefit from this additional information. Global classifiers of this type are heavily specific to the underlying flat classification algorithm, as the original classification algorithms are modified in some way to take into account the entire class hierarchy. This might represent a disadvantage when compared to the local classifier approaches, which are not specific to a classification algorithm and can be augmented in a number of different ways. However, to the user, the output of a global classifier approach might be easier to understand/interpret than the one from a local classifier approach, due to the typically much smaller size of the classification model produced by the former approach, as mentioned earlier. This is the case for instance in Vens et al. (2008), where the number of rules generated by the global approach is much smaller than the number of rules generated by the local approaches used in their experiments. Also, the global classifier approach does not suffer from the major drawback of the local classifier approach, namely the fact that a misclassification at a given class level is propagated to the lower levels of the class hierarchy. Different modifications of the base flat classification algorithms have been proposed by different authors, as follows.

In Wang et al. (2001) an association rule mining algorithm is heavily modified in order to handle hierarchical document categorization. The main modification was to make the algorithm work with a set of labels instead of a single label.

In Clare and King (2003) a modified version of the decision tree algorithm C4.5 to handle the class hierarchy (HC4.5) was used. However, there are few details available about how this algorithm is different from the standard C4.5. The only information the authors provide is that they modified the entropy calculation formula to consider some form of weighting. It seems that, other things being equal, deeper nodes are preferred over shallower ones, because deeper nodes provide more specific class predictions to users. In Silla Jr and Freitas (2009a) the authors have used the same principle to create a global-model Naive Bayes classifier.

In Blockeel et al. (2002, 2006) and Vens et al. (2008) the authors present the Clus-HMC algorithm, which is based on predictive cluster trees. The main idea of the method is to build a set of classification trees to predict a set of classes, instead of only one class. To do this, the authors transform the classification output into a vector with boolean components corresponding to the possible classes. They also need to take into account some sort of distance-based metric to calculate how similar or dissimilar the training examples are in the classification tree. Originally the metric used was the weighted Euclidian Distance. In the work of Aleksovski et al. (2009) the authors investigated the use of other distance measures, namely the Jaccard distance, the SimGIC distance and the ImageClef distance. They concluded that there was no statistically significant difference between the different distance metrics. Also, in Dimitrovski et al. (2008) the authors have proposed the use of two ensembles approaches (bagging

Table 2 Global classifier approaches and their underlying flat counterpart

Base algorithm	Global approach
Ant-Miner	Otero et al. (2009)
Association rule-based classifier	Wang et al. (2001)
C4.5	Clare and King (2003)
Naive Bayes	Silla Jr and Freitas (2009a)
Predictive clustering trees	Blockeel et al. (2006) and Vens et al. (2008)
Kernel machines	Cai and Hofmann (2004, 2007), Dekel et al. (2004a,b), Rousu et al. (2005, 2006), Seeger (2008), Qiu et al. (2009), and Wang et al. (2009)

and random forests) applied to the Clus-HMC algorithm and concluded that the use of ensembles improves the classification accuracy.

In Otero et al. (2009) the authors proposed the hAnt-Miner algorithm, a global-model hierarchical Ant-Miner classification method (a type of swarm intelligence method based on the paradigm of ant colony optimization) to cope with DAGs.

Table 2 lists the original flat classification algorithm and which authors have modified it in order to create global classification approaches.

6 A unifying framework for hierarchical classification

Based on our discussion so far, there are very many types of hierarchical classification algorithms and a number of different types of hierarchical classification problems. Hence, there is a clear need for a more precise way of describing (using a standardized terminology as much as possible) which kind of hierarchical classification problem is being solved, and what are the characteristics of the hierarchical classification algorithm being used. For this reason, in this section we propose a unifying framework for hierarchical classification problems and algorithms.

6.1 Categorization of the different types of hierarchical classification problems

In the proposed framework a hierarchical classification problem is described as a 3-tuple $\langle \Upsilon, \Psi, \Phi \rangle$, where:

- Υ specifies the type of graph representing the hierarchical classes (nodes in the graph) and their interrelationships (edges in the graph). The possible values for this attribute are:
  - *T* (tree), indicating that the classes to be predicted are arranged into a tree structure;
  - *D* (DAG), indicating that the classes to be predicted are arranged into a DAG (Direct Acyclic Graph).

– Ψ indicates whether a data instance is allowed to have class labels associated with a single or multiple paths in the class hierarchy. For instance, in the tree-structured class hierarchy of Fig. 4, if there is a data instance whose most specific labels are, say, both 2.1.1 and 2.2.1, that instance has multiple paths of labels (MPL). This attribute can take on two values, as follows (the values' names are self-explained):
  – SPL—Single path of labels. This term is equivalent to the term "single label per class level" which was used in the previous sections of this paper (to be consistent with some works in the literature). In the proposed unifying framework we prefer the new term because it can be naturally applied to both trees and DAGs, whilst the definition of "class level" is not so clear in the case of DAGs.
  – MPL—Multiple paths of labels. This term is equivalent to the term "hierarchically multi-label" which was used in the previous sections.
– Φ describes the label depth of the data instances, as follows.
  – The value FD (full depth labeling) indicates that all instances have a full depth of labeling, i.e. every instance is labeled with classes at all levels, from the first level to the leaf level.
  – The value PD (partial depth labeling) indicates that at least one instance has a partial depth of labeling, i.e. the value of the class label at some level (typically the leaf level) is unknown. In practice it is often useful to know not only that a dataset has at least one instance with a partial depth of labeling, but also the precise proportion of instances with such partial depth of labeling. Hence, in the problem-describing tuple of the proposed framework, the value of this attribute can be specified in a more precise way as $\text{PD}_{\%}$, where % means the percentage of the instances that have partial depth labeling.

6.2 Categorization of different types of hierarchical classification algorithms

A hierarchical classification algorithm is described as a 4-tuple $\langle \Delta, \Xi, \Omega, \Theta \rangle$, where:

– Δ indicates whether or not the algorithm can predict labels in just one or multiple (more than one) different paths in the hierarchy. For instance, in the tree-structured class hierarchy of Fig. 4, if the algorithm can predict both class 1.1 and 1.2 to a given instance, which is equivalent to predicting the paths R-1-1.1 and R-1-1.2, then the algorithm is capable of multiple label path prediction. This attribute can take on two values, as follows:
  – SPP (single path prediction) indicates that the algorithm can assign to each data instance at most one path of predicted labels.
  – MPP (multiple path prediction) indicates that the algorithm can potentially assign to each data instance multiple paths of predicted labels.

  Note that this attribute is conceptually similar to the aforementioned Ψ attribute used to describe hierarchical classification problems; but they refer to different entities (algorithms vs. problems). If the target problem is a SPL (single path of (true) labels) one, it would be more natural to use a SPP (single path predic-

tion) algorithm, since a MPP (multiple path prediction) algorithm would have "too much flexibility" for the target problem and could produce invalid classifications, wrongly assigning MPL to some instances. If the target problem is a MPL (multiple paths of (true) labels) one, then one should use a MPP algorithm, since a SPP algorithm would clearly have "too little flexibility" for the target problem, not predicting true labels to some instances. In practice, however, in order to avoid the complexities associated with MPP algorithms, some works simply transform an original MPL problem into a simpler SPL problem, and then apply a SPP algorithm to the simplified data set. This kind of transformation can be achieved by using, for instance, variations of the methods for transforming flat multi-label problems into flat single-label ones described by Tsoumakas and Katakis (2007), with proper adaptations for the context of hierarchical classification. In any case, when such a problem simplification is done, it should be clearly indicated in the work.

- Ξ is the prediction depth of the algorithm. It can have two values:
  - MLNP (mandatory leaf-node prediction) which means the algorithm always assign leaf class(es).
  - NMLNP (non-mandatory leaf-node prediction) which means the algorithm can assign classes at any level (including leaf classes).
    Again, there is a natural relationship between this Ξ attribute for describing algorithms and its counterpart Φ attribute for describing problems. If the target problem is a FD (full depth labeling) one, one should of course use a MLNP algorithm, since a NMLNP algorithm would have "too much flexibility" and would "under-classify" some instances. If the target problem is a PD (Partial Depth Labeling) one, one should of course use a NMLNP algorithm, since a MLNP algorithm would have "too little flexibility" and would "over-classify" some instances.
- Ω is the taxonomy structure the algorithm can handle. It has two values:
  - *T* (tree), indicating that the classes to be predicted are arranged into a tree structure;
  - *D* (DAG), indicating that the classes to be predicted are arranged into a DAG (Direct Acyclic Graph).

  In principle an algorithm designed for coping with DAGs can be directly applied (without modification) to trees. However, the converse is not true, i.e., if an algorithm was designed for coping with tree-structured class hierarchies only, it would have to be significantly extended to cope with DAGs, as discussed across earlier sections of this paper.
- Θ is the categorization of the algorithm under the proposed taxonomy (Sect. 3) and has the values:
  - LCN (local classifier per node). Within this category, there is also another argument that needs to be specified, which is the strategy used for selecting negative and positive examples. It can have the following values (most of them defined previously in Sect. 4.1):
    - E (Exclusive).
    - LE (Less exclusive).

    - LI (Less inclusive).
    - I (Inclusive).
    - S (Siblings).
    - ES (Exclusive siblings).
    - D (Dynamic) for the cases where the positive and negative examples are selected in a dynamic way [like in Fagni and Sebastiani (2007)], but in this case the paper should clearly state how the examples are chosen.
- LCL (Local classifier per level).
- LCPN (Local classifier per parent node).
- GC (Global classifier).

Hence, researchers in hierarchical classification can use this unifying framework to make precisely clear what are the main characteristics of the problem they are solving and also the main characteristics of the hierarchical classification algorithm being used.

7 Conceptual and empirical comparison between different hierarchical classification approaches

In the previous sections, we provided a critical review of the existing approaches for the task of hierarchical classification. Therefore, it is interesting to compare the existing approaches on an abstract level. Table 3 provides a summary of the different approaches, considering their advantages and disadvantages. In that table, the three rows referring to the three types of local classifiers consider only the training phase of those local approaches. The next row considers the testing phase of any of those three types of local classifiers using the top-down approach. For each row in the table, the description of advantages and disadvantages is self-explanatory.

Also, it is interesting to verify what kinds of approaches have been investigated and what kinds of class structure (tree or DAG) have been used so far in the literature. Table 4 classifies the works reviewed in this paper according to the new proposed taxonomy. The analysis of Table 4 shows that the majority of the research carried out so far deals with tree-structured classification problems, rather than DAG-structured ones. Also, the number of papers found in the literature using local classifiers is more than twice the number of papers using global classifiers. This is expected as developing new global classifiers is more complicated than using local approaches with well-known classifiers.

Considering the issues of single/multiple path predictions and prediction depth, a more detailed analysis is carried out in Table 5. Note, however, that this table contains only the papers in the literature which provide clear information about these two issues. Therefore, Table 5 refers to fewer papers than Table 4, although the papers which are mentioned in the former are reported in more detail, according to the standardized terminology of the proposed unified framework. It should be noted that a significant number of papers that are mentioned in Table 4 are not mentioned in Table 5 because those papers did not provide clear information about some characteristics of the corresponding hierarchial classification problem or algorithm. This reinforces the need for the hierarchical classification community in general to be clearer on which kind

Table 3 Summary of characteristics of different approaches, at a high level of abstraction

Hierarchical approach	Advantages	Disadvantages
Flat classifier	Simplicity;	Completely ignores the class hierarchy;
LCN (training phase)	Simplicity; Naturally multi-label;	May suffer from the blocking problem; Prone to inconsistency; Employs a greater number of classifiers;
LCPN (training phase)	Simplicity; Employs fewer classifiers than LCN;	May suffer from the blocking problem; Prone to inconsistency;
LCL (training phase)	Simplicity; Employs a small number of classifiers;	Prone to inconsistency; A classifier might have to discriminate among a large number of classes (at deep levels); Ignores parent–child class relationships during training;
(Any) Local classifier with the top-down class prediction approach	Preserves natural constrains in class membership; Considers the class hierarchy during testing and during the creation of the training sets; Generality (can be used with any base classifier);	May suffer from the blocking problem; Depending on the problem at hand, can create a very complex set of cascade of classifiers, which in turn leads to a complex classification model; Misclassification at a given class node is propagated downwards to all its descendant classes;
Global classifier	Preserves natural constrains in class membership; Considers the class hierarchy during training and testing; Single (although complex) decision model;	Classifier-specific;

of problem and what type of algorithms they are using, and the proposed unifying framework offers a standardized terminology and a taxonomy for this purpose.

Although the great majority of research has been carried out on local classifiers, one question that naturally arises is whether a particular type of approach is better than the others or not. In order to investigate that, a compilation of the existing literature (based on the conclusions of the authors of each paper) is shown in Table 6, where the symbols $\uparrow, \downarrow, \sim$ represent whether each approach (corresponding to a given row in the table) obtained a better ($\uparrow$), worse ($\downarrow$) or similar ($\sim$) predictive performance than the approach shown in the corresponding column. The names of the approaches in the rows and columns of this table are abbreviated as follows: LCN is the local classifier per node, LCPN is the Local Classifier per Parent Node and LCL is the Local Classifier per Level. It should be noted that when a particular approach is compared against itself, e.g. LCPN against LPCN, this represents the case where the authors propose a new method within the same broad approach and use the standard approach of that type as a baseline. Also, the lack of any comparisons in a given table cell should not

Table 4 Categorization of hierarchical classification methods proposed in the literature according to the taxonomy proposed in this paper

Approach (Θ)	Class structure (Ω)	List of works
Flat classifier	Tree	Barbedo and Lopes (2007)
	DAG	Hayete and Bienkowska (2005)
LCN	Tree	D'Alessio et al. (2000), Dumais and Chen (2000), Sun and Lim (2001), Mladenic and Grobelnik (2003), Sun et al. (2003, 2004), Liu et al. (2005), Wu et al. (2005), Cesa-Bianchi et al. (2006a,b), Cesa-Bianchi and Valentini (2009), Esuli et al. (2008), Punera and Ghosh (2008), Xue et al. (2008), Bennett and Nguyen (2009), Binder et al. (2009), Valentini (2009) and Valentini and Re (2009)
	DAG	Barutcuoglu and DeCoro (2006), Barutcuoglu et al. (2006), DeCoro et al. (2007), Guan et al. (2008) and Jin et al. (2008)
LCPN	Tree	Koller and Sahami (1997), Chakrabarti et al. (1998), McCallum et al. (1998), Weigend et al. (1999), D'Alessio et al. (2000), Ruiz and Srinivasan (2002), Burred and Lerch (2003), Tikk and Biró (2003), Tikk et al. (2003), McKay and Fujinaga (2004), Li and Ogihara (2005), Brecheisen et al. (2006a), Tikk et al. (2007), Holden and Freitas (2005, 2006, 2008, 2009), Xiao et al. (2007), Secker et al. (2007, 2010), Costa et al. (2008), Silla Jr and Freitas (2009b) and Gauch et al. (2009)
	DAG	Kriegel et al. (2004)
LCL	Tree	Clare and King (2003)
	DAG	
Global classifier	Tree	Labrou and Finin (1999), Wang et al. (1999, 2001), Clare and King (2003), Blockeel et al. (2006), Cai and Hofmann (2004, 2007), Dekel et al. (2004a,b), Peng and Choi (2005), Rousu et al. (2005, 2006), Astikainen et al. (2008), Seeger (2008), Silla Jr and Freitas (2009a) and Qiu et al. (2009)
	DAG	Kiritchenko et al. (2005, 2006), Alves et al. (2008), Dimitrovski et al. (2008), Vens et al. (2008), Aleksovski et al. (2009), Otero et al. (2009), Wang et al. (2009)

be interpreted as no comparisons were done in the corresponding cell. Sometimes this is the case, while in others the authors compare only different variations of their own approach (e.g. parameter tuning) and not to other approaches.

A careful analysis of the data compiled in Table 6 shows that, taking into account the works that compare their hierarchical approaches against flat classification,

Table 5 A more detailed categorization of some hierarchical classification works, according to the following attributes of the proposed unifying framework: approach (Θ), class structure (Ω), label cardinality prediction (Δ) and prediction depth (Ξ)

$\langle\Theta, \Omega, \Delta, \Xi\rangle$	List of works
⟨LCN, T, SPP, NMLNP⟩	Punera and Ghosh (2008) and Binder et al. (2009)
⟨LCN, T, MPP, NMLNP⟩	Dumais and Chen (2000), Cesa-Bianchi et al. (2006a,b), Cesa-Bianchi and Valentini (2009), Bennett and Nguyen (2009), Valentini (2009) and Valentini and Re (2009)
⟨LCN, D, MPP, NMLNP⟩	Barutcuoglu and DeCoro (2006), Barutcuoglu et al. (2006), DeCoro et al. (2007), Guan et al. (2008) and Jin et al. (2008)
⟨LCPN, T, SPP, MLNP⟩	Koller and Sahami (1997), Chakrabarti et al. (1998), Weigend et al. (1999), Ruiz and Srinivasan (2002), Burred and Lerch (2003), Tikk and Biró (2003), McKay and Fujinaga (2004), Li and Ogihara (2005), Holden and Freitas (2005, 2006, 2008, 2009), Xiao et al. (2007), Secker et al. (2007, 2010), Costa et al. (2008), Silla Jr and Freitas (2009b) and Gauch et al. (2009)
⟨LCPN, T, SPP, NMLNP⟩	Tikk et al. (2007)
⟨GC, T, SPP, MLNP⟩	Qiu et al. (2009)
⟨GC, T, SPP, NMLNP⟩	Labrou and Finin (1999) and Silla Jr and Freitas (2009a)
⟨GC, T, MPP, NMLNP⟩	Clare and King (2003), Blockeel et al. (2006), Rousu et al. (2005, 2006), Dimitrovski et al. (2008) and Aleksovski et al. (2009)
⟨GC, D, SPP, NMLNP⟩	Otero et al. (2009)
⟨GC, D, MPP, NMLNP⟩	Alves et al. (2008) and Vens et al. (2008)

the hierarchical approaches are usually better than the flat classification approach. However, it is less clear if the global approach is better or worse to deal with hierarchical classification problems than the local approach.

Two studies that tried to answer that question were Costa et al. (2007b) and Ceci and Malerba (2007). In Costa et al. (2007b) an evaluation comparing: the flat classification approach, the local classifier per level approach, the local classifier per parent node approach and a global approach [using HC4.5 (Clare 2004)] was performed using two biological datasets with the same base classifier (C4.5). In those experiments, the local classifier per parent node with the top-down class prediction approach performed better on the first dataset while the global approach performed better on the second dataset.

In Ceci and Malerba (2007) the authors investigated the use of flat classifiers against the hierarchical local classifier per parent node approach using the same base classifier: SVM or Naive Bayes depending on the experiment. In their experiments, using accuracy as the evaluation measure, the flat SVM obtained better results than its hierarchical counterpart. In a deeper analysis of the misclassified instances, the authors used a combination of four measures into one. The idea behind the use of different measures was to verify different types of errors in a hierarchical classification sce-

Table 6 An analysis of how the hierarchical classification methods proposed in the literature performed when compared to other approaches

Approach	Work	Result when compared against				
		Flat	LCN	LCPN	LCL	GC
LCN	Brecheisen et al. (2006a)	∼				
	D'Alessio et al. (2000)	↑				
	Liu et al. (2005)	↑				
	Cesa-Bianchi et al. (2006a,b)	↑	↑			
	Cesa-Bianchi and Valentini (2009)	↑				
	DeCoro et al. (2007)	↑				
	Guan et al. (2008)	↑				
	Valentini (2009)	↑	↑			
	Valentini and Re (2009)	↑	↑			
	Sun et al. (2004)		↑			
	Barutcuoglu and DeCoro (2006)		↑			
	Punera and Ghosh (2008)		↑			
	Bennett and Nguyen (2009)		↑			
	DeCoro et al. (2007)			↑		
LCPN	Koller and Sahami (1997)	∼				
	Burred and Lerch (2003)	∼				
	Chakrabarti et al. (1998)	↑				
	McCallum et al. (1998)	↑				
	Dumais and Chen (2000)	↑				
	Ruiz and Srinivasan (2002)	↑				∼
	Kriegel et al. (2004)	↑				
	McKay and Fujinaga (2004)	↑				
	Li and Ogihara (2005)	↑				
	Xiao et al. (2007)	↑				
	Jin et al. (2008)	↑				
	Gauch et al. (2009)	↑				
	Secker et al. (2007)			↑		
	Costa et al. (2008)			↑		
	Holden and Freitas (2008)			↑		
LCL	Clare and King (2003)					∼
GC	Dekel et al. (2004a,b)	↑		↑		
	Wang et al. (2001)	↑				
	Peng and Choi (2005)	↑				
	Rousu et al. (2005, 2006)	↑				
	Blockeel et al. (2006)	↑				
	Cai and Hofmann (2004, 2007)	↑				
	Wang et al. (1999)	↑				
	Kiritchenko et al. (2005, 2006)	↑		∼		

Table 6 continued

Approach	Work	Result when compared against				
		Flat	LCN	LCPN	LCL	GC
	Astikainen et al. (2008)	↑				
	Wang et al. (2009)	↑				
	Vens et al. (2008)		↑			
	Otero et al. (2009)			↑		
	Silla Jr and Freitas (2009a)			↑		
	Clare and King (2003)				∼	
	Aleksovski et al. (2009)					∼
	Blockeel et al. (2006)					↑
	Dimitrovski et al. (2008)					↑
	Qiu et al. (2009)					↑

nario (e.g. sibling classification error; predicting only higher-level classes (and not the most specific class) for a given example, etc.). After this deeper analysis of the misclassification errors, the authors noticed that although the flat SVM is more accurate, it commits more serious errors than its hierarchical counterpart. Therefore, it seems that whether one particular approach is better than another remains an open question. Hence, the issue of whether one particular approach is better than another naturally depends on the evaluation measure used.

Regardless of which approach is better, the analysis of Ceci and Malerba (2007) raises an important concern: How to evaluate hierarchical classification algorithms? Since the use of flat classification measures might not be enough to give us enough insight at which algorithm is really better. Before trying to answer this question, we analysed how the evaluation was carried out in the surveyed papers. The analysis shows that most researchers used standard flat classification evaluation measures, while recognizing that they are not ideal, because the errors at different levels of the class hierarchy should not be penalized in the same way. Other authors propose their own hierarchical classification evaluation measures, which are often only used by the ones who propose it, and in some cases there is not a clear definition of the evaluation measure being suggested. There are also cases when researchers use more than one existing evaluation measure and also propose their own! A good review of hierarchical classification evaluation measures is found in Sun et al. (2003), although it is out of date now. A more recent survey on evaluation measures for hierarchical classification was presented by Costa et al. (2007a), however it was limited to tree-structured problems with single path of labels. An evaluation measure that can cope with multiple paths of labels in tree-structured problems was proposed by Cesa-Bianchi et al. (2006b), called h-loss (for hierarchical loss) as opposed to the traditional zero-one loss. The h-loss however cannot cope with DAGs.

There seems to be no studies that empirically compare the use of the different hierarchical classification evaluation measures, in different application domains (which is important as they have very different class structures), against the flat classifica-

tion accuracy measure. This would be particularly interesting because most of the approaches currently use flat classification evaluation measures. When comparing a hierarchical classification approach against a flat classification approach authors usually report small gains in accuracy, while when using the hierarchical evaluation measures proposed in Kiritchenko et al. (2006) the difference in predictive accuracy over the flat approach in the worst case was of 29.39%. This poses an interesting question, if hierarchical approaches overall show similar or better results against the flat classification approach when using flat classification evaluation measures, couldn't the results be actually much better if a hierarchical classification evaluation measure was used instead?

This question naturally leads to the question of which hierarchical classification measure to use? Based on our experience, we suggest the use of the metrics of hierarchical precision (hP), hierarchical recall (hR) and hierarchical f-measure (hF) proposed by Kiritchenko et al. (2005). They are defined as follows: $\text{hP} = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{P}_i|}$, $\text{hR} = \frac{\sum_i |\hat{P}_i \cap \hat{T}_i|}{\sum_i |\hat{T}_i|}$, $\text{hF} = \frac{2*\text{hP}*\text{hR}}{\text{hP}+\text{hR}}$, where $\hat{P}_i$ is the set consisting of the most specific class(es) predicted for test example i and all its(their) ancestor classes and $\hat{T}_i$ is the set consisting of the true most specific class(es) of test example i and all its(their) ancestor classes. The summations are of course computed over all test examples. Note that these measures are extended versions of the well known metrics of precision, recall and f-measure but tailored to the hierarchical classification scenario. To determine if there is statistically significant difference between different algorithms, the interested reader is referred to García and Herrera (2008).

Although no hierarchical classification measure can be considered the best one in all possible hierarchical classification scenarios and applications, the main reason for recommending the hP, hR and hF measures is that, broadly speaking, they can be effectively applied (with a caveat to be discussed later) to any hierarchical classification scenario; i.e., tree-structured, DAG-structured, SPL, MPL, MLNP or NMLNP problems. Let us elaborate on these points, in the context of the categorization of different types of hierarchical classification problems and algorithms proposed in the previous section.

First, the hP, hR and hF measures can be applied not only to tree-structured classes, but also to DAG-structured classes. In the latter case, although in a DAG a node can have multiple paths, one can still compute the set of all ancestors of a node (possibly involving multiple paths from the node to the root) without any ambiguity, and this set of ancestors is basically what is needed to compute these hierarchical classification measures. Secondly, these measures can be applied not only to SPL problems, but also to MPL problems, since one can also compute the set of all ancestors of multiple nodes without any ambiguity. Thirdly, the hP, hR and hF measures can also be naturally applied to full depth labeling problems, associated with MLNP algorithms.

The fourth case to be considered here, and the most interesting and complex one, is the case of partial depth labeling problems associated with NMLNP algorithms. This is a scenario where the application of these measures faces some problems, in particular due to a relationship between the concepts of hierarchical precision and hierarchical recall and the concepts of generalization and specialization errors presented in Ceci

and Malerba (2007). In the latter work, a generalization error refers to the case where the most specific class predicted for an example is more generic than the true most specific known class associated with the example; e.g., predicting only class R.1 for an example whose most specific known class is R.1.1. A specialization error refers to the case where the most specific class predicted for an example is more specific than the true most specific known class associated with the example; e.g. predicting class R.1.1 for an example whose most specific known class is R.1.

To illustrate some issues associated with the hP, hR and hF measures in the context of generalization and specialization errors, let us consider some hypothetical examples. Consider the following three cases of generalization errors:

– (A) Predicted classes: "R.1", True Known Classes: "R.1.2"
– (B) Predicted classes: "R.1", True Known Classes: "R.1.2.1"
– (C) Predicted classes: "R.1", True Known Classes: "R.1.2.1.1"

In these cases the values of hP and hR will be, respectively:

– (A) hP = 1/1; hR = 1/2
– (B) hP = 1/1; hR = 1/3
– (C) hP = 1/1; hR = 1/4

Hence, one can see that for a fixed predicted class, the larger the generalization error (corresponding to a deeper true known class), the lower the hR value, whilst the hP value remains constant. Now let us consider the following cases of specialization errors:

– (D) Predicted classes: "R.2.2", True known class: "R.2"
– (E) Predicted classes: "R.2.2.1", True known class: "R.2"
– (F) Predicted classes: "R.2.2.1.3", True known class: "R.2"

In these cases the values of hP and hR will be, respectively:

– (D) hP = 1/2; hR = 1/1
– (E) hP = 1/3; hR = 1/1
– (F) hP = 1/4; hR = 1/1

Hence, one case see that for a fixed true known class, the larger the specialization error (corresponding to a deeper predicted class), the lower the hP value, whilst the hR value remains constant. In summary, the hF measure, which aggregates hP and hR into a single formula, seems to be able to effectively penalize both generalization and specialization errors, at first glance.

However, there is a problem associated with the use of the hP measure in the context of the so-called "specialization error", as follows. Suppose that the most specific true known class for an example is R.1, and the algorithm predicts to that example the class R.1.1, leading to a hP value of 1/2. Is this penalization fair? Can we be sure that this kind of over-specialized prediction is really an error? This seems to depend on the application. In some applications perhaps we could consider this penalization fair, and consider this over-specialization as an error, if we interpret the most specific true known class as representing the absolute truth about the classes associated with the example. In practice, however, in many applications this interpretation seems unfair,

because the most specific true known class associated with an example represents, as emphasized by the use of the keyword "known", just our current state of knowledge, which includes current uncertainties about deeper classes that might be solved later, as more knowledge about the example's classes becomes available.

To consider a concrete example, a major application of hierarchical classification is in the prediction of protein functions where classes are terms in the Gene Ontology, which is briefly reviewed in the next Section. In this application, many proteins are currently annotated with very generic classes only. However, this does not mean the protein really does not have more specific classes, it just means the more specific classes of the protein are not known at present, but might very well be discovered later by biologists. In this kind of application, if the most specific known class of an example is R.1, if the algorithm predicts for that example class R.1.1, the only thing we can really say for sure is that the prediction was correct at the first level, we simply do not know if the prediction was correct or not at the second level, since the true class of the example at the second level is unknown. Therefore, in this kind of application there is an argument to modify the definition of hP in such a way that over-specialized predictions are not considered as errors and so are not penalized.

Despite the above problem, overall the measures of hP, hR and hF seem effective measures of hierarchical classification across a broad range of scenarios, as discussed above, which justifies their recommendation. In Sokolova and Lapalme (2009) the authors also consider these measures to be adequate as they do not depend on subjective and user-specific parameters like the distance-based or semantics-based measures. It should also be noted that, in contrast to the hP, hR and hF measures, some other measures of hierarchical classification face some problems in their computation when applied to DAG-structured and/or MPL problems. For instance, although a distance-based measure can naturally be applied to a tree-structured class hierarchy, the concept of the distance between two nodes faces some ambiguity in a DAG, where there can be multiple paths between two nodes. In that case, it is not clear if the distance between two nodes should be given by the shortest, longest or average distance among all paths connecting those two nodes.

8 Major applications of hierarchical classification

8.1 Text categorization

The use of hierarchical classification in the field of text categorization dates back to at least 1997, when Koller and Sahami (1997) proposed the use of a local classifier per parent node for training coupled with the top-down class-prediction method for testing. There are different types of motivation to work with hierarchical classification in this field. The first one is due to the large growing of the number of electronic documents, and a natural way to handle them is to organize them into hierarchies. Indeed, virtually any type of electronic document can be organized into a taxonomy, e.g. webpages, digital libraries, patents, e-mails, etc. For instance, in Chakrabarti et al. (1998) the authors propose an interesting example showing how the use of hierarchies can improve the use of information retrieval systems. The example they use is to search for

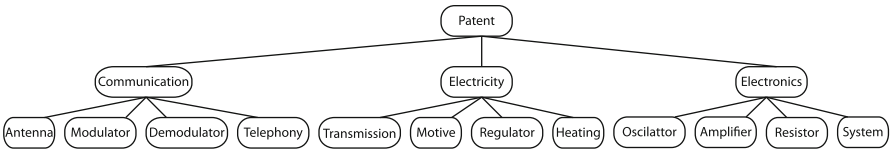

Fig. 8 A small part of one of the two taxonomies used in Chakrabarti et al. (1998) that represents a portion of the US patent database taxonomy

the keywords *jaguar* (and other related words to the animal) on web-search websites. They note that for the user it would be very difficult to retrieve the information he/she was seeking, as a huge amount of information about the car was returned. However, if the user could limit his/her search within a hierarchy (e.g. search for jaguar in the part of the hierarchy rooted at animals), that would help to disambiguate polysemous terms. Figure 8 illustrates one example of a document-related class hierarchy.

8.2 Protein function prediction

In bioinformatics, particularly in the task of protein function prediction, the classes to be predicted (protein functions) are naturally organized into class hierarchies. Examples of these hierarchies are the Enzyme Commission (Barret 1997) and the Gene Ontology (Ashburner et al. 2000). The Enzyme Commission class hierarchy is—as suggested by its name—specific to enzymes (proteins that speed up chemical reactions), but the Gene Ontology class hierarchy is extremely generic, and can be applied to potentially any type of protein. Protein function prediction is important because this type of information can be potentially used to develop drugs and for better diagnosis and treatment of diseases, since many diseases are caused by or related to malfunctioning of proteins. Figure 9 illustrates a very small part of the Gene Ontology hierarchy. It is important to note that other hierarchical classification schemes exist to annotate proteins, e.g. the MIPS FunCat (Ruepp et al. 2004). These different hierarchies have been used by different authors (Clare and King 2003; Kriegel et al. 2004; Wu et al. 2005; Barutcuoglu et al. 2006; Blockeel et al. 2006; Rousu et al. 2006; Alves et al. 2008; Guan et al. 2008; Costa et al. 2008; Vens et al. 2008; Otero et al. 2009).

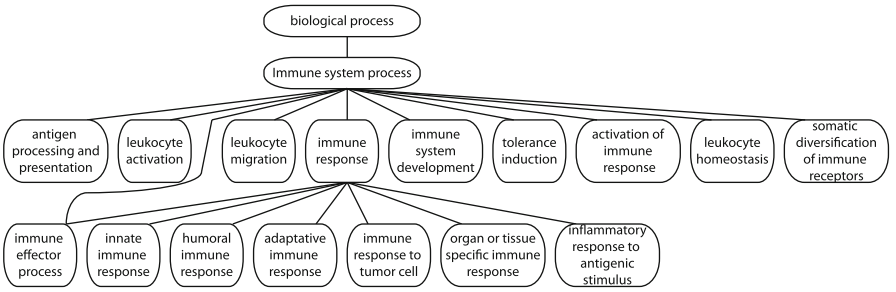

Fig. 9 Illustration of the top level structure of the immune system processes in the Gene Ontology (Ashburner et al. 2000)

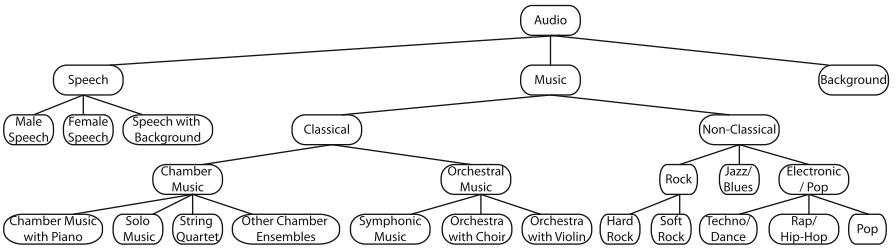

Fig. 10 The audio class hierarchy Used in Burred and Lerch (2003)

8.3 Music genre classification

In organizing and retrieving music information, the genre plays an important concept, as there are studies that show that genre is one of the most used concepts to search for music in music information systems (Downie and Cunningham 2002; Lee and Downie 2004). As with other applications, having the genres organized into a class hierarchy helps users to browse and retrieve this information. So far most of the work in this area is only concerned with music genres as a flat classification problem, although many researchers acknowledge the possibility of using class hierarchies in their future works. Some of the works that have used class hierarchies in this application domain are: (Burred and Lerch 2003; McKay and Fujinaga 2004; Li and Ogihara 2005; Brecheisen et al. 2006a; Barbedo and Lopes 2007; DeCoro et al. 2007; Silla Jr and Freitas 2009b). The idea of using the hierarchy for browsing and retrieval has been explored so far in two existing tools for organizing music collections: Zhang (2003) demonstrates an end-user system based on the use of hierarchies to organize music collections; and Brecheisen et al. (2006b) allows the system to have user feed-back in order to re-organize the pre-existing class hierarchy as the users see fit. Figure 10 illustrates the audio class hierarchy used in Burred and Lerch (2003).

8.4 Other applications

Although the existing literature has used hierarchical classification methods to deal with the types of applications described in the previous sections, of course the use of those methods is not limited to those applications. In this section, we briefly review some projects that use hierarchical classification approaches to deal with different types of applications.

In Dekel et al. (2004b) the authors use a large margin classifier in the task of hierarchical phoneme classification. This task consists of classifying the phonetic identity of a (typically short) speech utterance (Dekel et al. 2004b). In this context the class hierarchy plays the role of making the misclassifications less severe. Figure 11 illustrates the phonetic hierarchy for American English.

In Barutcuoglu and DeCoro (2006) the authors use their Bayesian Network aggregation with k-NN base classifiers in the task of 3D shape classification. The motivation to use hierarchical approaches to this problem is that in 3D shape classification sce-

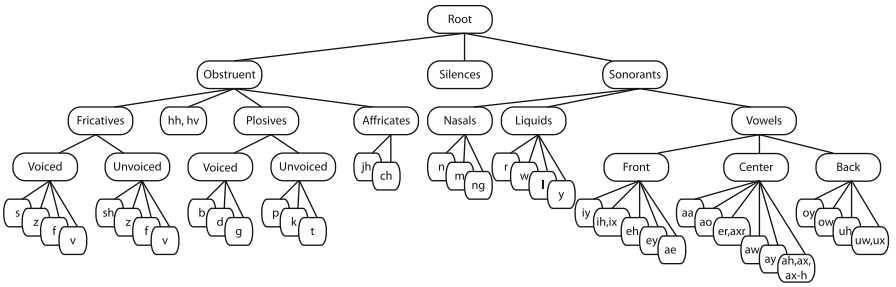

Fig. 11 The phonetic tree of American English (Dekel et al. 2004b)

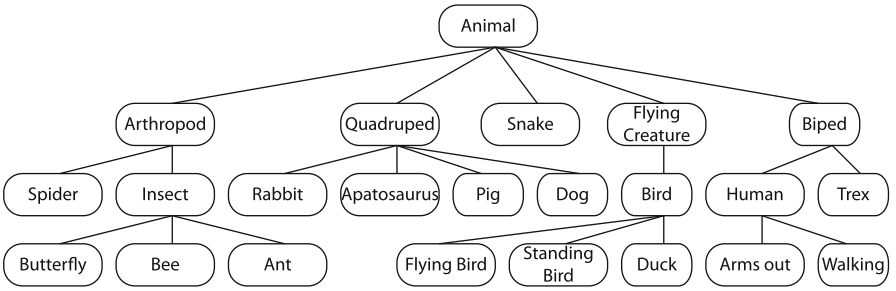

Fig. 12 The animal branch of the Princeton Shape Database (Shilane et al. 2004) used by Barutcuoglu and DeCoro (2006)

narios classes are arranged in a hierarchy from most general to most specific shapes. Moreover, a common problem in shape analysis involves assigning semantic meaning to geometry by using a pre-existing class hierarchy. In their experiments they used the Princeton Shape Benchmark (Shilane et al. 2004), which has a 4 level depth hierarchy and 198 leaf classes. Figure 12 illustrates the sub-tree Animals of the hierarchy. Other works that deal with hierarchical image classification are (Dimitrovski et al. 2008) and (Binder et al. 2009).

In Xiao et al. (2007) the authors build a class hierarchy for the task of hierarchical classification of emotional speech. The database used in this paper is Berlin emotional speech database (Burkhardt et al. 2005). They create a 3 level depth hierarchy to distinguish between 6 leaf classes (which are types of emotion): anger, boredom, fear, gladness, sadness and neutral. They use a LCPN approach with a Multi Layer Perceptron (MLP) Neural Network with sequential forward feature selection. Figure 13 illustrates their class hierarchy.

A summary of the literature cited above according to their application domain and type of hierarchical classification approach used is presented in Table 7.

9 Concluding remarks

In this paper we surveyed the literature on hierarchical classification problems and methods, which is scattered across different application domains. Based on the exist-

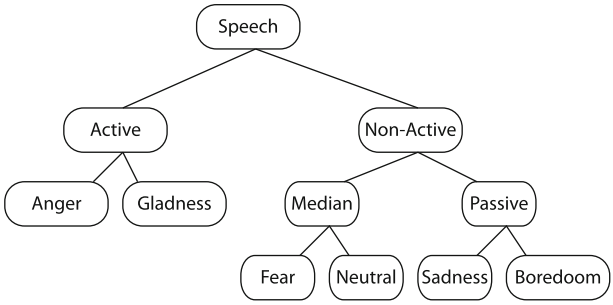

Fig. 13 The hierarchy used for mood classification based on speech in the Berlin Dataset (Burkhardt et al. 2005) used by Xiao et al. (2007)

ing literature we proposed a new unifying framework for hierarchical classification, including a taxonomy of hierarchical classification methods, in order to clarify the similarities and differences between a number of these types of problems and methods. The new proposed unifying framework can be used by the hierarchical classification community to clearly describe the characteristics of their proposed methods and the problems they are working on, and to facilitate the comparison of their proposed methods with other methods proposed in the literature.

One of the main contributions of the proposed taxonomy is to identify the "top-down" approach as an approach concerning mainly the testing phase of a classification algorithm, being therefore an approach to a large extent independent of the particular local approach used for training. By contrast, in the literature the term "top-down" approach is used to refer to both the testing and training phases, and without distinguishing the type of local classifier approach used in the training phase of the system. We also performed an analysis of the advantages and disadvantages of each type of method identified in the new taxonomy.

We also investigated what kind of class structures are most often used (tree or DAG) as well as how to evaluate the different hierarchical classification systems. We have observed that most of the research so far has been concerned with tree-structured class hierarchies, probably due to the fact that trees are considerably simpler structures than DAGs. However, some DAG-structured class hierarchies are very useful—a major example is the Gene Ontology, which has become very important in biology. Hence, there is a clear need for more research on hierarchical classification methods for DAG-structured class hierarchies.

We also analyzed the results reported in a number of works in order to verify if there is a type of hierarchical classification approach which is better than others, and it seems that any hierarchical classification approach (local or global) is overall better than the flat classification approach, when solving a hierarchical classification problem. Only in rare cases the authors obtained similar results for hierarchical and flat classification approaches, but in these cases they have used a flat classification evaluation measure, which does not take the different types of hierarchical classification errors into account.

Table 7 Summary of the existing literature on hierarchical classification according to the type of application domain and the type of hierarchical classification approach

Type of application	Hierarchical classification approach (Θ)	List of works
Text categorization	LCN	D'Alessio et al. (2000), Sun and Lim (2001), Mladenic and Grobelnik (2003), Sun et al. (2003, 2004), Wu et al. (2005), Cesa-Bianchi et al. (2006a,b), Esuli et al. (2008), Jin et al. (2008), Punera and Ghosh (2008), Xue et al. (2008) and Bennett and Nguyen (2009)
	LCPN	Koller and Sahami (1997), Chakrabarti et al. (1998), McCallum et al. (1998), Weigend et al. (1999), D'Alessio et al. (2000), Dumais and Chen (2000), Ruiz and Srinivasan (2002), Tikk and Biró (2003), Tikk et al. (2003, 2007), Kriegel et al. (2004) and Gauch et al. (2009)
	GC	Labrou and Finin (1999), Wang et al. (1999, 2001), Dekel et al. (2004a), Cai and Hofmann (2004, 2007), Rousu et al. (2005, 2006), Kiritchenko et al. (2005, 2006), Peng and Choi (2005), Seeger (2008) and Qiu et al. (2009)
Protein function prediction	LCN	Wu et al. (2005), Barutcuoglu et al. (2006), Guan et al. (2008), Valentini (2009), Valentini and Re (2009) and Cesa-Bianchi and Valentini (2009)
	LCPN	Holden and Freitas (2005, 2006, 2008, 2009), Secker et al. (2007, 2010), Costa et al. (2008) and Kriegel et al. (2004)
	LCL	Clare and King (2003)
	GC	Clare and King (2003), Blockeel et al. (2006), Alves et al. (2008), Rousu et al. (2006), Vens et al. (2008), Astikainen et al. (2008), Otero et al. (2009), Silla Jr and Freitas (2009a), Aleksovski et al. (2009) and Wang et al. (2009)
Music genre classification	LCN	DeCoro et al. (2007)
	LCPN	Burred and Lerch (2003), McKay and Fujinaga (2004), Li and Ogihara (2005), Brecheisen et al. (2006a) and Silla Jr and Freitas (2009b)
Image classification	LCN	Barutcuoglu and DeCoro (2006) and Binder et al. (2009)
	GC	Dimitrovski et al. (2008)
Emotional speech classification	LCPN	Xiao et al. (2007)
Phoneme classification	GC	Dekel et al. (2004a,b)

Finally, we also briefly reviewed several types of applications of hierarchical classification methods, ranging from text categorization to bioinformatics to music genre classification and other application domains. We hope that this review of applications will stimulate further research on the important topic of hierarchical classification.

Concerning future research directions, in addition to the aforementioned need for more research involving DAG-structured class hierarchies, there is also a need for larger-scale comparisons of different hierarchical classification methods. Fortunately, a set of datasets for large-scale hierarchical classification in bioinformatics has been recently made freely available at http://www.cs.kuleuven.be/~dtai/clus/hmcdatasets/, which provides the hierarchical classification community with a useful set of benchmarking datasets.

Also, how to efficiently perform feature selection in hierarchical classification remains a topic deserving more attention. One issue that most authors agree on is that, intuitively, different features are better at discriminating between classes at different levels of the class hierarchy (Koller and Sahami 1997; Mladenic and Grobelnik 2003; Ceci and Malerba 2007; Esuli et al. 2008; Secker et al. 2010). In Esuli et al. (2008) the authors mention that both feature selection and the selection of negative training examples should be performed "locally", paying attention to the topology of the classification scheme. For global classification approaches, however, how to perform feature selection remains an open question.

Other research directions include the use of relational learning, semi-supervised learning (Ceci 2008) and other more sophisticated types of machine learning in hierarchical classification scenarios.

Acknowledgements The first author is financially supported by CAPES—a Brazilian research-support agency (process number 4871-06-5). We also thank the anonymous reviewers for their insightful feedback on the earlier version of this manuscript.

References

Aleksovski D, Kocev D, Dzeroski S (2009) Evaluation of distance measures for hierarchical multilabel classification in functional genomics. In: Proceedings of the 1st workshop on learning from multi-label data (MLD) held in conjunction with ECML/PKDD, pp 5–16

Altun Y, Hofmann T (2003) Large margin methods for label sequence learning. In: Proceedings of the 8th European conference on speech communication and technology (EuroSpeech)

Alves RT, Delgado MR, Freitas AA (2008) Multi-label hierarchical classification of protein functions with artificial immune systems. In: Advances in bioinformatics and computational biology, Lecture notes in bioinformatics, vol 5167. Springer, Berlin, pp 1–12

Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G (2000) Gene ontology consortium. Gene ontology: tool for the unification of biology. Nat Genet 25:25–29

Astikainen K, Holmand L, Pitkanen E, Szedmak S, Rousu J (2008) Towards structured output prediction of enzyme function. BMC Proc 2(Suppl 4)

Barbedo JGA, Lopes A (2007) Automatic genre classification of musical signals. EURASIP J Adv Signal Process 2007:12

Barret AJ (1997) Nomenclature committee of the international union of biochemistry and molecular biology (NC-IUBMB). Enzyme Nomenclature. Recommendations 1992. Supplement 4: corrections and additions. Eur J Biochem 250(1):1–6

Barutcuoglu Z, DeCoro C (2006) Hierarchical shape classification using bayesian aggregation. In: Proceedings of the IEEE conference on shape modeling and applications

Barutcuoglu Z, Schapire RE, Troyanskaya OG (2006) Hierarchical multi-label prediction of gene function. Syst Biol 22:830–836

Bennett PN, Nguyen N (2009) Refined experts: improving classification in large taxonomies. In: Proceedings of the 32nd international ACM SIGIR conference on research and development in information retrieval, pp 11–18

Binder A, Kawanabe M, Brefeld U (2009) Efficient classification of images with taxonomies. In: Proceedings of the 9th Asian conference on computer vision

Blockeel H, Bruynooghe M, Dzeroski S, Ramon J, Struyf J (2002) Hierarchical multi-classification. In: Proceedings of the first SIGKDD workshop on multirelational data mining (MRDM-2002), pp 21–35

Blockeel H, Schietgat L, Struyf J, Džeroski S, Clare A (2006) Decision trees for hierarchical multilabel classification: a case study in functional genomics. In: Knowledge discovery in databases: PKDD 2006. Lecture notes in computer science, vol 4213. Springer, Berlin, pp 18–29

Brecheisen S, Kriegel HP, Kunath P, Pryakhin A (2006a) Hierarchical genre classification for large music collections. In: Proceedings of the IEEE 7th international conference on Multimedia & Expo, pp 1385–1388

Brecheisen S, Kriegel HP, Kunath P, Pryakhin A, Vorberger F (2006b) MUSCLE: music classification engine with user feedback. In: Springer (ed) Proceedings of the 10th international conference on extending database technology, vol 3896 in Lecture notes in computer science, pp 1164–1167

Burkhardt F, Paeschke A, Rolfes M, Sendlmeier WF, Weiss B (2005) A database of German emotional speech. In: Proceedings of the 9th European conference on speech communication and technology, pp 1517–1520

Burred JJ, Lerch A (2003) A hierarchical approach to automatic musical genre classification. In: Proceedings of the 6th international conference on digital audio effects, pp 8–11

Cai L, Hofmann T (2004) Hierarchical document categorization with support vector machines. In: Proceedings of the 13th ACM international conference on information and knowledge management, pp 78–87

Cai L, Hofmann T (2007) Exploiting known taxonomies in learning overlapping concepts. In: Proceedings of the 20th international joint conference on artificial intelligence, pp 714–719

Ceci M (2008) Hierarchical text categorization in a transductive setting. In: Proceedings of the IEEE international conference of data mining workshops, pp 184–191

Ceci M, Malerba D (2007) Classifying web documents in a hierarchy of categories: a comprehensive study. J Intell Inform Syst 28(1):1–41

Cesa-Bianchi N, Valentini G (2009) Hierarchical cost-sensitive algorithms for genome-wide gene function prediction. In: Third international workshop on machine learning in systems biology

Cesa-Bianchi N, Gentile C, Zaniboni L (2006a) Hierarchical classification: combining Bayes with SVM. In: Proceedings of the 23rd international conference on machine learning, pp 177–184

Cesa-Bianchi N, Gentile C, Zaniboni L (2006b) Incremental algorithms for hierarchical classification. J Mach Learn Res 7:31–54

Chakrabarti S, Dom B, Agrawal R, Raghavan P (1998) Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. VLDB J 7:163–178

Chen Y, Crawford MM, Ghosh J (2004) Integrating support vector machines in a hierarchical output space decomposition framework. In: Proceedings of the IEEE international symposium on geoscience and remote sensing, vol 2, pp 949–952

Clare A (2004) Machine learning and data mining for yeast functional genomics. PhD thesis, University of Wales Aberystwyth

Clare A, King RD (2003) Predicting gene function in *Saccharomyces cerevisiae*. Bioinformatics 19(suppl 2):ii42–ii49

Costa E, Lorena A, Carvalho A, Freitas A (2007a) A review of performance evaluation measures for hierarchical classifiers. In: Evaluation methods for machine learning II: papers from the 2007 AAAI Workshop, AAAI Press, pp 1–6

Costa E, Lorena A, Carvalho A, Freitas AA, Holden N (2007b) Comparing several approaches for hierarchical classification of proteins with decision trees. In: Advances in bioinformatics and computational biology, Lecture notes in bioinformatics, vol 4643. Springer, Berlin, pp 126–137

Costa EP, Lorena AC, de Carvalho A, Freitas AA (2008) Top-down hierarchical ensembles of classifiers for predicting g-protein-coupled-receptor functions. In: Advances in Bioinformatics and computational biology. Lecture notes in bioinformatics, vol 5167. Springer, Berlin, pp 35–46

D'Alessio S, Murray K, Schiaffino R, Kershenbaum A (2000) The effect of using hierarchical classifiers in text categorization. In: Proceedings of the 6th international conference Recherche d´ Information Assistee par Ordinateur, pp 302–313

DeCoro C, Barutcuoglu Z, Fiebrink R (2007) Bayesian aggregation for hierarchical genre classification. In: Proceedings of the 8th international conference on music information retrieval, Vienna, Austria, pp 77–80

Dekel O, Keshet J, Singer Y (2004a) Large margin hierarchical classification. In: Proceedings of the 21th international conference on Machine learning

Dekel O, Keshet J, Singer Y (2004b) An online algorithm for hierarchical phoneme classification. In: Proceedings of the 1st machine learning for multimodal interaction workshop. Lecture notes in computer science, vol 3361. Springer, Berlin, pp 146–158

Dimitrovski I, Kocev D, Loskovska S, Dzeroski S (2008) Hierarchical annotation of medical images. In: Proceedings of the 11th international multiconference information society, vol A, pp 174–177

Downie JS, Cunningham SJ (2002) Toward a theory of music information retrieval queries: System design implications. In: Proceedings of the 3rd international conference on music information retrieval, pp 299–300

Dumais ST, Chen H (2000) Hierarchical classification of Web content. In: Belkin NJ, Ingwersen P, Leong MK (eds) Proceedings of the 23rd ACM international conference on research and development in information retrieval, pp 256–263

Eisner R, Poulin B, Szafron D, Lu P, Greiner R (2005) Improving protein function prediction using the hierarchical structure of the gene ontology. In: Proceedings of the IEEE symposium on computational intelligence in bioinformatics and computational biology, pp 1–10

Esuli A, Fagni T, Sebastiani F (2008) Boosting multi-label hierarchical text categorization. Inform Retr 11(4):287–313

Fagni T, Sebastiani F (2007) On the selection of negative examples for hierarchical text categorization. In: Proceedings of the 3rd language technology conference, pp 24–28

Freitas AA, de Carvalho ACPLF (2007) Research and trends in data mining technologies and applications, Idea Group, chap A: tutorial on hierarchical classification with applications in bioinformatics, pp 175–208

Freitas COA, Oliveira LS, Aires SBK, Bortolozzi F (2008) Metaclasses and zoning mechanism applied to handwriting recognition. J Univers Comput Sci 14(2):211–223

García S, Herrera F (2008) An extension on "statistical comparisons of classifiers over multiple data sets" for all pairwise comparisons. J Mach Learn Res 9:2677–2694

Gauch S, Chandramouli A, Ranganathan S (2009) Training a hierarchical classifier using inter document relationships. J Am Soc Inform Sci Technol 60(1):47–58

Gerlt JA, Babbitt PC (2000) Can sequence determine function? Genome Biol 1(5):1–10

Guan Y, Myers CL, Hess DC, Barutcuoglu Z, Caudy AA, Troyanskaya OG (2008) Predicting gene function in a hierarchical context with an ensemble of classifiers. Genome Biol 9(Suppl 1):S3

Hao PY, Chiang JH, Tu YK (2007) Hierarchically SVM classification based on support vector clustering method and its application to document categorization. Expert Syst Appl 33:627–635

Hayete B, Bienkowska J (2005) Gotrees: predicting go associations from protein domain composition using decision trees. In: Proceedings of the Pacific symposium on biocomputing, pp 127–138

Holden N, Freitas AA (2005) A hybrid particle swarm/ant colony algorithm for the classification of hierarchical biological data. In: Proceedings of the 2nd IEEE swarm intelligence symposium, pp 100–107

Holden N, Freitas AA (2006) Hierarchical classification of g-protein-coupled receptors with a pso/aco algorithm. In: Proceedings of the 3rd IEEE swarm intelligence symposium, pp 77–84

Holden N, Freitas AA (2008) Improving the performance of hierarchical classification with swarm intelligence. In: Proc. 6th European conference on evolutionary computation, machine learning and data mining in bioinformatics (EvoBio). Lecture notes in computer science, vol 4973. Springer, Berlin, pp 48–60

Holden N, Freitas AA (2009) Hierarchical classification of protein function with ensembles of rules and particle swarm optimisation. Soft Comput J 13:259–272

Jin B, Muller B, Zhai C, Lu X (2008) Multi-label literature classification based on the gene ontology graph. BMC Bioinform 9:525

Kiritchenko S, Matwin S, Famili AF (2005) Functional annotation of genes using hierarchical text categorization. In: Proceedings of the ACL workshop on linking biological literature, ontologies and databases: mining biological semantics

Kiritchenko S, Matwin S, Nock R, Famili AF (2006) Learning and evaluation in the presence of class hierarchies: application to text categorization. In: Proceedings of the 19th Canadian conference on artificial intelligence. Lecture notes in artificial intelligence, vol 4013, pp 395–406

Koerich AL, Kalva PR (2005) Unconstrained handwritten character recognition using metaclasses of characters. In: Proceedings of the IEEE international conference on image processing, vol 2, pp 542–545

Koller D, Sahami M (1997) Hierarchically classifying documents using very few words. In: Proceedings of the 14th international conference on machine learning, pp 170–178

Kriegel HP, Kroger P, Pryakhin A, Schubert M (2004) Using support vector machines for classifying large sets of multi-represented objects. In: Proceedings of the SIAM international conference on data mining, pp 102–114

Kumar S, Ghosh J, Crawford MM (2002) Hierarchical fusion of multiple classifiers for hyperspectral data analysis. Pattern Anal Appl 5:210–220

Labrou Y, Finin T (1999) Yahoo! as an ontology—using yahoo! categories to describe documents. In: Proceedings of the ACM conference on information and knowledge management, pp 180–187

Lee JH, Downie JS (2004) Survey of music information needs, uses, and seeking behaviours: preliminary findings. In: Proceedings of the fifth international conference on music information retrieval, Barcelona, Spain, pp 441–446

Li T, Ogihara M (2005) Music genre classification with taxonomy. In: Proceedings of the IEEE international conference on acoustics, speech, and signal processing, pp 197–200

Li T, Zhu S, Ogihara M (2007) Hierarchical document classification using automatically generated hierarchy. J Intell Inform Syst 29(2):211–230

Liu TY, Yang Y, Wan H, Zeng HJ, Chen Z, Ma WY (2005) Support vector machines classification with a very large-scale taxonomy. ACM SIGKDD Explor Newsl 7(1):36–43

Lorena AC, Carvalho ACPLF (2004) Comparing techniques for multiclass classification using binary svm predictors. In: Proceedings of the IV Mexican international conference on artificial intelligence. Lecture notes in artificial intelligence, vol 2972, pp 272–281

McCallum A, Rosenfeld R, Mitchell TM, Ng AY (1998) Improving text classification by shrinkage in a hierarchy of classes. In: Proceedings of the international conference on machine learning, pp 359–367

McKay C, Fujinaga I (2004) Automatic genre classification using large high-level musical feature sets. In: Proceedings of the international conference on music information retrieval, pp 525–530

Mladenic D, Grobelnik M (2003) Feature selection on hierarchy of web documents. Decis Support Syst 35:45–87

Otero FEB, Freitas AA, Johnson CG (2009) A hierarchical classification ant colony algorithm for predicting gene ontology terms. In: Pizzuti C, Ritchie M, Giacobini M (eds) Proceedings of the 7th European conference on evolutionary computation, machine learning and data mining in bioinformatics (EvoBio). Lecture Notes in Computer Science, vol 5483. Springer, Berlin, pp 68–79

Peng X, Choi B (2005) Document classifications based on word semantic hierarchies. In: Proceedings of the international conference on artificial intelligence and applications, pp 362–367

Punera K, Ghosh J (2008) Enhanced hierarchical classification via isotonic smoothing. In: Proceedings of the 17th international conference on World Wide Web, pp 151–160

Punera K, Rajan S, Ghosh J (2005) Automatically learning document taxonomies for hierarchical classification. In: Proceedings of the international World Wide Web conference, pp 1010–1011

Qiu X, Gao W, Huang X (2009) Hierarchical multi-class text categorization with global margin maximization. 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, Association for computational linguistics, pp 165–168

Rocchio JJ (1971) The SMART retrieval system: experiments in automatic document processing, chap: relevance feedback in information retrieval, Prentice Hall, pp 313–323

Rousu J, Saunders C, Szedmak S, Shawe-Taylor J (2005) Learning hierarchical multi-category text classification models. In: Proceedings of the 22nd international conference on machine learning, pp 744–751

Rousu J, Saunders C, Szedmak S, Shawe-Taylor J (2006) Kernel-based learning of hierarchical multilabel classification models. J Mach Learn Res 7:1601–1626

Ruepp A, Zollner A, Maier D, Albermann K, Hani J, Mokrejs M, Tetko I, Guldener U, Mannhaupt G, Munsterkotter M, Mewes HW (2004) The FunCat, a functional annotation scheme for systematic classification of proteins from whole genomes. Nucleic Acids Res 32(18):5539–5545

Ruiz ME, Srinivasan P (2002) Hierarchical text categorization using neural networks. Inform Retr 5: 87–118

Sasaki M, Kita K (1998) Rule-based text categorization using hierarchical categories. In: Proceedings of IEEE international conference on systems, man, and cybernetics, pp 2827–2830

Secker A, Davies M, Freitas A, Timmis J, Mendao M, Flower D (2007) An experimental comparison of classification algorithms for the hierarchical prediction of protein function. Expert Updat (the BCS-SGAI Mag) 9(3):17–22

Secker A, Davies M, Freitas AA, Clark E, Timmis J, Flower DR (2010) Hierarchical classification of g-protein-coupled-receptors with data-driven selection of attributes and classifiers. Int J Data Mining Bioinform 4(2):191–210

Seeger MW (2008) Cross-validation optimization for large scale structured classification kernel methods. J Mach Learn Res 9:1147–1178

Shilane P, Kazhdan M, Min P, Funkhouser T (2004) The Princeton shape benchmark. In: Proceedings of the shape modeling international

Silla Jr CN, Freitas AA (2009a) A global-model naive bayes approach to the hierarchical prediction of protein functions. In: Proceedings of the 9th IEEE international conference on data mining, pp 992–997

Silla Jr CN, Freitas AA (2009b) Novel top-down approaches for hierarchical classification and their application to automatic music genre classification. In: Proceedings of the IEEE international conference on systems, man, and cybernetics, pp 3599–3604

Sokolova M, Lapalme G (2009) A systematic analysis of performance measures for classification tasks. Inform Process Manag 45:427–437

Sun A, Lim EP (2001) Hierarchical text classification and evaluation. In: Proceedings of the IEEE international conference on data mining, pp 521–528

Sun A, Lim EP, Ng WK (2003) Performance measurement framework for hierarchical text classification. J Am Soc Inform Sci Technol 54(11):1014–1028

Sun A, Lim EP, Ng WK, Srivastava J (2004) Blocking reduction strategies in hierarchical text classification. IEEE Trans Knowl Data Eng 16(10):1305–1308

Tikk D, Biró G (2003) Experiment with a hierarchical text categorization method on the wipo-alpha patent collection. In: Proceedings of the 4th international symposium on uncertainty modeling and analysis, pp 104–109

Tikk D, Yang JD, Bang SL (2003) Hierarchical text categorization using fuzzy relational thesaurus. Kybernetika 39(5):583–600

Tikk D, Biró G, Yang JD (2004) A hierarchical text categorization approach and its application to frt expansion. Aust J Intell Inform Process Syst 8(3):123–131

Tikk D, Biró G, Torcsvári A (2007) Emerging technologies of text mining: techniques and applications, Idea Group, chap: a hierarchical online classifier for patent categorization, pp 244–267

Tsoumakas G, Katakis I (2007) Multi label classification: an overview. Int J Data Wareh Mining 3(3):1–13

Tsochantaridis I, Joachims T, Hofmann T, Altun Y (2005) Large margin methods for structured and interdependent output variables. J Mach Learn Res 6:1453–1484

Valentini G (2009) True path rule hierarchical ensembles. In: Kittler J, Benediktsson J, Roli F (eds) Proceedings of the eighth international workshop on multiple classifier systems. Lecture notes in computer science, vol 5519. Springer, Berlin, pp 232–241

Valentini G, Re M (2009) Weighted true path rule: a multilabel hierarchical algorithm for gene function prediction. In: Proceedings of the 1st workshop on learning from multi-label data (MLD) held in conjunction with ECML/PKDD, pp 132–145

Vens C, Struyf J, Schietgat L, Džeroski S, Blockeel H (2008) Decision trees for hierarchical multi-label classification. Mach Learn 73(2):185–214

Wang K, Zhou S, Liew SC (1999) Building hierarchical classifiers using class proximity. In: In Proceedings of the 25th conference on very large data base. Morgan Kaufmann Publishers, San Francisco, pp 363–374

Wang K, Zhou S, He Y (2001) Hierarchical classification of real life documents. In: Proceedings of the 1st SIAM international conference on data mining, Chicago, USA

Wang J, Shen X, Pan W (2009) Large margin hierarchical classification with multiple paths. J Am Stat Assoc 104(487):1213–1223

Weigend AS, Wiener ED, Pedersen JO (1999) Exploiting hierarchy in text categorization. Inform Retr 1:193–216

Wu F, Zhang J, Honavar V (2005) Learning classifiers using hierarchically structured class taxonomies. In: Proceedings of the symposium on abstraction, reformulation, and approximation, vol 3607. Springer, Berlin, pp 313–320

Xiao Z, Dellandréa E, Dou W, Chen L (2007) Hierarchical Classification of Emotional Speech. Technical report RR-LIRIS-2007-006, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/Ecole Centrale de Lyon, http://liris.cnrs.fr/publis/?id=2742

Xue GR, Xing D, Yang Q, Yu Y (2008) Deep classification in large-scale text hierarchies. In: Proceedings of the 31st annual international ACM SIGIR conference on research and development in information retrieval, pp 619–626

Zhang T (2003) Semi-automatic approach for music classification. In: Proceedings of the SPIE conference on internet multimedia management systems, pp 81–91