Hierarchical Classification by Expected Utility Maximization

Korinna Bade, Eyke Hüllermeier, Andreas Nürnberger
Faculty of Computer Science,
Otto-von-Guericke-University Magdeburg, D-39106 Magdeburg, Germany
{kbade,nuernb}@iws.cs.uni-magdeburg.de
huellerm@iti.cs.uni-magdeburg.de

Abstract

Hierarchical classification refers to an extension of the standard classification problem, in which labels must be chosen from a class hierarchy. In this paper, we look at hierarchical classification from an information retrieval point of view. More specifically, we consider a scenario in which a user searches a document in a topic hierarchy. This scenario gives rise to the problem of predicting an optimal entry point, that is, a topic node in which the user starts searching. The usefulness of a corresponding prediction strongly depends on the search behavior of the user, which becomes relevant if the document is not immediately found in the predicted node. Typically, users tend to browse the hierarchy in a top-down manner, i.e., they look at a few more specific subcategories but usually refuse exploring completely different branches of the search tree. From a classification point of view, this means that a prediction should be evaluated, not solely on the basis of its correctness, but rather by judging its usefulness against the background of the user behavior. The idea of this paper is to formalize hierarchical classification within a decision-theoretic framework which allows for modeling this usefulness in terms of a user-specific utility function. The prediction problem thus becomes a problem of expected utility maximization. Apart from its theoretical appeal, we provide first empirical results showing that the approach performs well in practice.

1. Introduction

Classifying objects into hierarchical structures is a task that is needed in a growing number of applications, for example in maintaining ontologies or keyword hierarchies in web-based systems. This includes, e.g., classifying web pages into a web directory or a hierarchy defined by groups of users. As an example of such groups we mention social web communities such as, e.g., the open directory project [8]. Despite the possibility of creating such hierarchies manually and assigning documents by hand, automatic classification and hierarchy extension would be beneficial for many domains as the number of documents to classify has become huge.

Hierarchical structures are used to help locating information of value to the user. When searching for information in a hierarchy, the user has some idea about the topic this information belongs to. This knowledge is used to browse the hierarchy labels in a top-down manner until a subclass is found which is expected to contain the information. Once the user reaches the most specific class describing his information need, he starts scanning the documents. If none contains the information, he might also browse more general classes. However, it is very unlikely that he would browse other specific classes in branches of the hierarchy that deal with different topics.

What does this search behavior imply for a classification method? Each object that is classified in a wrong subclass will most likely not be retrieved by the user. Each object that is classified into a class that is a generalization of the correct class might still be retrieved, depending on how much time the user is willing to spend on his search. Furthermore, a specific class might not yet exist for a document. Here, classification in one of the most specific classes would prevent the user from retrieving the document as he would not look so deep down the hierarchy. In this case, predicting a more general class is the only way of making retrieval possible.

In our approach, we formalize these assumptions about user behavior in terms of a *utility function*. Roughly speaking, predicting a class that is on the path from the root of the hierarchy to the true class should receive a positive utility since, starting from the former, the user has a good chance to find the latter. As opposed to this, predicting a class that is not on this path should be punished by a low (negative) utility, since it makes finding the true node very unlikely. Assuming information about the true node to be given in terms of probability degrees, we consider the problem of class prediction as a problem of decision making under uncertainty. More specifically, referring to the well-founded decision-theoretic principle of expected utility maximization, we predict the class with the highest expected utility. This decision principle is closely related to and completely in agreement with optimal classification policies in cost-sensitive learning [10].

Our paper is structured as follows. In the next section, we review related work on hierarchical classification and provide a succinct background on decision theory and cost-sensitive learning. Our decision-theoretic approach to hierarchical classification is then introduced in Sect. 3 and evaluated empirically in Sect. 4. Conclusions and a discussion of future work are given in Sect. 5.

2. Related Work

This section provides a brief review of related work on hierarchical classification. Since our approach is developed within a decision-theoretic framework, we also make some remarks on statistical decision theory. Finally, a close connection to cost-sensitive learning will be pointed out.

2.1. Hierarchical Classification

Most of the related work for hierarchical classification deals with integrating the hierarchy information into the classification process by adapting existing methods or building new classifiers. The authors of [3] try to integrate hierarchy information directly into a support vector machine (SVM) classifier by integrating a hierarchical loss function, which is motivated by a document filtering setting. Their motivation is similar to ours. Different SVM classifiers for each hierarchy node are learned by the authors of [21] and [9] to find a suitable node in the tree. However, an inner node can only be predicted, when data is assigned to it. We discuss this issue in more detail in Sect. 3.

The authors of [16] applied shrinkage to the estimates of a Bayes classifier to improve the probability estimates. They reported large improvements. As estimating class probabilities is one step of our algorithm, this method could be applied to improve the estimates. In [5], an incremental algorithm with performance close to SVM and also a new loss function for evaluation is presented. The authors propose to learn linear threshold classifiers for each node to decide whether a document should be classified into the node or further down the hierarchy.

In [7], a greedy probabilistic hierarchical classifier is used to determine the most suitable path in the hierarchy from the root to a leaf. In a second step, another classifier is used to determine the best class along this path. The authors also suggest some criteria to create a new category. In [14], the performance of two Boosting algorithms, BoosTexter and CentroidBooster, is compared to the performance of support vector machines.

The influence of different training sets (with and without using hierarchy information) is examined in [4] using Naïve Bayes and centroid learning for different numbers of extracted features. The author of [12] adapts the determined weights for each category for a certain document in a post-processing step by integrating the weights of all other categories according to their proximity (the distance in the category tree/graph) to this category. However, he makes no differences concerning the relation between two nodes.

In summary, three main approaches to hierarchical classification can be distinguished. Firstly, the original training data can be reinterpreted in a pre-processing step, which is mostly done by assigning it not only to one class but also to parent and/or child classes in the hierarchy. This approach is critical in our context, since we do not assume the class associated with a node to be the union of the classes of the successor nodes (e.g. a document may belong to an inner node but not to any of the more specific topics associated with the successor nodes).

Secondly, the hierarchy could be used directly in the classifier design, which means developing completely new classification methods. Some examples of this approach have been mentioned above.

Our approach belongs to a third category, in which the class hierarchy is exploited in a post-processing step. More specifically, this step consists of reinterpreting basic probability assignments, which typically come from a standard (non-hierarchical) classifier, in terms of a user-specific utility function. In other words, the hierarchical structure of the problem is exploited via this utility function. As an advantage of this class of methods let us mention that it allows for using well-established standard classifiers in the first step (this advantage is of course shared by the first class of methods).

2.2. Decision Theory

Statistical decision theory is concerned with decision making under uncertainty and has been developed to a fairly advanced level. Classical contributions in this field have been made by Ramsey [18], de Finetti [11], von Neumann and Morgenstern [17], and Savage [19]. One of the most famous results, which has first been shown by von Neumann and Morgenstern, concerns the characterization of a rational agent in terms of expected utility maximization. Roughly speaking, it is shown that, in a context where uncertainty is represented in terms of a probability measure over a set of potential world states, an agent who obeys some (reasonable) rationality postulates behaves like an expected utility maximizer. More specifically, there is a utility function (on outcomes of action/world state combinations) such that the agent prefers one action over a second one if and only if the expected utility of choosing the former is higher than the expected utility of the latter, where the expectation is taken with respect to the aforementioned probability measure.

It is important to mention that expected utility theory (EUT) not only provides an axiomatic foundation of a particular decision behavior, but also offers a formal framework for modeling decision problems in a systematic way. The basic EUT setup (in the finite case) can simply be illustrated in the form of a table as follows:

$$\begin{array}{c|cccc} & \rho_1 & \rho_2 & \cdots & \rho_n \\ & \omega_1 & \omega_2 & \cdots & \omega_n \\ \hline a_1 & u_{11} & u_{12} & \cdots & u_{1n} \\ a_2 & u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_m & u_{m1} & u_{m2} & \cdots & u_{mn} \end{array} \tag{1}$$

Here, $\omega_1 \ldots \omega_n$ denote the world states that are not under the control of the decision maker. Each state ω_j is assumed to occur with probability ρ_j. The a_i define the set of actions the decision maker can choose from. Choosing action a_i in world state ω_j yields a utility of u_{ij}. In this simple setting, the expected utility of action a_i is hence given by

$$EU(a_i) = \sum_{j=1}^{n} \rho_j \times u_{ij}.$$

As a normative theory of rational behavior, statistical decision theory has played an essential role in fields like economics or the social sciences for a long time. Meanwhile, however, ideas and concepts from decision theory are also used in other research areas, notably in artificial intelligence. In fact, a large number of problems has already been formalized and successfully solved within a decision-theoretic framework, including generic ones like search and planning [1] as well as concrete applications such as, e.g., database selection in networked information retrieval [13].

2.3 Cost-Sensitive Classification

The approach proposed in this paper can also be seen as a special type of cost-sensitive classification, where the cost matrix is determined by the hierarchical structure of the classes. Cost-sensitive classification generalizes the common classification setting by assuming that misclassification errors may incur different penalties. More specifically, if c_j is the true class, then predicting class c_i produces a cost of c_{ij}. These misclassification costs are typically summarized in the form of a cost matrix:

$$\begin{array}{c|cccc} & c_1 & c_2 & \cdots & c_n \\ \hline c_1 & c_{11} & c_{12} & \cdots & c_{1n} \\ c_2 & c_{21} & c_{22} & \cdots & c_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n & c_{n1} & c_{n2} & \cdots & c_{nn} \end{array} \tag{2}$$

This matrix is obviously a special case of (1), except that utility values are replaced by cost values. Moreover, given probability estimates for the different classes, the recommended prediction in cost-sensitive learning is the one with minimum *expected* cost [10]. Thus, cost-sensitive learning can indeed be seen as a special case of EUT, where actions correspond to predictions and world states to classes.

We nevertheless prefer a decision-theoretic perspective as it is more general and allows for several interesting extensions. For example, even the simple EUT setting (1) gives more freedom with respect to defining actions and world states. Regarding the problem of hierarchical classification more generally as one of maintaining a document hierarchy, one might allow, e.g., for an action "creating a new subcategory". We shall come back to this point in the conclusions.

3. Hierarchical Classification: A Decision-theoretic Approach

The main idea of our approach is motivated by the retrieval scenario described in the introduction, in which a user tries to locate information in a hierarchy. To this end, probability estimates of classes are combined with utility values of each class, and the final (classification) decision is guided by the principle of expected utility maximization.

In the following, we first discuss the definition of a suitable utility function and then turn to the problem of decision making.

3.1. Defining Utility of Predictions

From our scenario at hand we derive the notion of the *retrieval path*, which starts at the correct node and goes up the hierarchy until the root node. I.e. the *retrieval path* rp_n associated with a node n contains each node n_i from the hierarchy H, which is either the node itself or a parent node thereof (denoted by $n_i \geq_h n$):

$$rp_n = \{n_i \in H | n_i \geq_h n\} \tag{3}$$

In Fig. 1, the retrieval path of node 4 is marked in bold as an example. Please note that the child nodes of node 4 (here nodes 6 and 7) do not belong to the retrieval path.

We denote by $\mathrm{dist}_H(c_1, c_2)$ the number of edges on the path between c_1 and c_2 in the tree, e.g., $\mathrm{dist}_H(4,4) = 0$ and $\mathrm{dist}_H(1,4) = 2$.

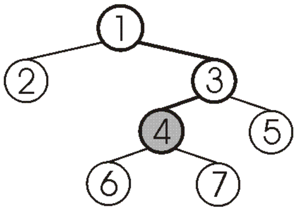

Figure 1. Example for a retrieval path

As motivated by our retrieval scenario, nodes, which do not belong to the retrieval path, are of no benefit to the user. Therefore, such nodes should be penalized in terms of a low utility value. For classifications along the retrieval path, the utility of corresponding nodes depends on the distance to the correct class. The highest utility should get of course the correct node, as this class represents the optimal entry point for a document search (most likely, the user will find the requested document quickly). The higher a document is classified along the retrieval path, the more steps the user will need, i.e., the more effort he has to invest to find the document. Therefore, the utility should decrease on the way from the correct node to the root.

The above requirements are fulfilled by a large number of utility functions. Here, we propose the following concrete measure:

$$\mathrm{util}(\hat{c}|c) = \begin{cases} \exp(-\gamma \cdot \mathrm{dist}_{\mathrm{H}}(\hat{c}, c)) & \text{if } \hat{c} \geq_h c \\ L & \text{else} \end{cases} \qquad (4)$$

This utility function has two parameters: L represents a penalty (loss) term and punishes a deviation from the retrieval path. This parameter will usually be non-positive and should at least be smaller than the utility of the root node. The parameter γ models the "laziness" of the user: The higher γ, the smaller the utility of a node with a certain distance from the correct class, hence the more important it becomes to predict a node, which is close to the true class. In particular, it is worth considering two extreme parameter configurations:

- With $L = 0$ and $\gamma \to \infty$, the utility becomes 0 for every node except the correct one, for which the utility is 1. This configuration models an extremely lazy user, who only explores a single node: As a prediction will not be corrected through generalization, it is useful only if it directly identifies the true class. For a classifier this means that information about the hierarchical structure of the classes is essentially meaningless, since no distinction is made between incorrect predictions: Being on the retrieval path is no longer better than being aside.

- The other extreme is obtained for $\gamma = 0$. Now, the utility is 1 along the complete retrieval path. Exploiting information about the hierarchical structure now becomes fairly easy: Predicting the root node by default is an optimal strategy and guarantees a utility of 1.

3.2. Decision Making

We assume to be given a class hierarchy together with a set of training data in the form of documents with associated classes. As an aside, we note that some classes might well be empty, i.e., there might be no instances of that class among the training data. As will be seen later on, our approach still allows to choose such classes as optimal prediction for new query documents.

In a first step, we train a (standard, possibly flat) classifier on the training data. Given a new query document d, we assume this classifier to output posterior probabilities $\hat{\rho}_i = P(c_i|d)$ for each potential class c_i. In our implementation, we employed two classifiers, namely a standard Naïve Bayes (NB) classifier and a SVM (based on the libSVM implementation [6]). However, any other classifier could be used instead, as long as it produces a probability distribution over the set of classes.

From a decision making point of view, we thus obtain a special EUT scenario as defined in (1), in which actions correspond to class predictions, the unknown world state corresponds to the true class, and the utility degrees u_{ij} are given by $\mathrm{util}(c_i|c_j)$ as defined in Eq. (4):

$$\begin{array}{c|cccc} & \rho_1 & \rho_2 & \cdots & \rho_n \\ & c_1 & c_2 & \cdots & c_n \\ \hline c_1 & u_{11} & u_{12} & \cdots & u_{1n} \\ c_2 & u_{21} & u_{22} & \cdots & u_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ c_n & u_{n1} & u_{n2} & \cdots & u_{nn} \end{array} \qquad (5)$$

Given the above decision scenario, EUT prescribes to predict the class with highest expected utility, where the expected utility of predicting class c_i for document d is given by

$$\mathrm{EU}(c_i|d) = \sum_{c_j \in H} P(c_j|d) \cdot \mathrm{util}(c_i|c_j) \qquad (6)$$

The complete algorithm, which we call HUClass (*H*ierarchical expected *U*tility based *Class*ification), is summarized in Fig. 2.

Let us again note that the matrix in (5) in principle corresponds to what is called the *cost-matrix* in cost-sensitive learning, with the only exception that we consider utility instead of cost values. Besides, as already indicated in Sect. 2.3, the above strategy of first estimating class probabilities from the training data set as given, and then making maximal expected utility (minimal expected loss) predictions is

```
HUClass($d$, $classifier$, $\gamma$, $L$)
    For each $c_i \in H$:
        Compute probability estimate $P(c_i|d)$ by
        $classifier$
    $c_{best}$ = null
    For each $c_i \in H$:
        Compute expected utility $\mathrm{EU}(c_i|d)$ by Eq. 6
        If $\mathrm{EU}(c_i|d) > \mathrm{EU}(c_{best}|d)$
            $c_{best} = c_i$
    Return $c_{best}$
```

Figure 2. The HUClass algorithm

completely in agreement with the recommended policy in cost-sensitive learning [10].

To understand the basic principle underlying the decision-theoretic approach to hierarchical classification as outlined above, it is useful to consider the classification itself as a hierarchical process: Starting at the root of the tree, the user has to decide whether he should start exploring the hierarchy at this node or further specialize, that is, immediately proceed to one of the successor nodes. The expected utility of specializing will be high only in relatively unambiguous situations, i.e., in situations where most of the probability mass is located in one of the subtrees. Otherwise, it might be better to stop at the current node, which will then have a higher expected utility than any successor. Please note that this situation may occur even if the current node itself has a rather low probability, perhaps even 0. As a concrete example, imagine a 2-level hierarchy with a root node having probability 0 and 3 successor nodes, each of which has a probability of $1/3$. The expected utility of the root is then given by $\exp(-\gamma)$ which, depending on γ, might be bigger than the expected utility of the three leaf nodes ($1/3$). Roughly speaking, selecting one of the successor nodes is simply too dangerous in this situation, as it is quite likely to make a wrong decision and, hence, to incur a high penalty. A random decision in favor of any of the leaf nodes will appear preferable only for a very lazy user characterized by a high γ (more concretely, $\gamma > \ln(3)$).

3.3 Training the Classifier

In order to evaluate predictions in our experimental studies in Sect. 4 below, we shall use the utility function (4) with $L = 0$ and $\gamma = 1/2$:

$$\mathrm{util}(\hat{c}|c) = \begin{cases} \exp(-\mathrm{dist_H}(\hat{c}, c)/2) & \text{if } \hat{c} \geq_h c \\ 0 & \text{else} \end{cases} \quad (7)$$

Thus, we assume that this is the "true" utility function of the user. Nevertheless, for making predictions, we shall allow the classifier to use any member of the parameterized class (4) of utility functions. In other words, in the training phase, γ and L will be used as tuning parameters for the classifier. Of course, using different utility functions (parameters) inside and outside the classifier, one for optimizing performance and the other for evaluation, might appear weird at first sight. One should recognize, however, that the probabilities $P(c_i|d)$ are only *estimations* of the true probabilities. Thus, what we compute in (6) is not the true expected utility of a prediction, but only an approximation thereof. More correctly, our classifier is hence an *estimated expected utility* maximizer.

Having the freedom to tune the parameters L and γ, the learner possesses a means to react to poor probability estimates. Roughly speaking, the lower the quality of the estimates, the more conservative the classifier should become, for example by decreasing L. To illustrate, consider a situation, in which the flat classifier is extremely misleading: It predicts a probability close to 1 for a randomly selected class and distributes the remaining probability mass uniformly over the other classes. Computing the expected utility according to (7), the optimal prediction of the hierarchical classifier will be the class with probability close to 1. However, as this class is chosen at random by the flat classifier, the true performance of the hierarchical classifier will be very poor. By making L small enough, the hierarchical classifier can achieve that the root node does always have the highest (estimated!) expected utility. The corresponding decision strategy of always predicting the root node is extremely cautious but will definitely have a higher (external) performance.

It is quite interesting to note that the above strategy of optimizing performance by adapting the utility function used inside the classifier is somewhat comparable to using a utility function in order to model the risk-aversion of a decision maker in economic applications. Generally speaking, the utility function in EUT can be seen as a means for controlling the decision behavior of an agent, and it is exactly used for this purpose in our application.

To find optimal parameters L and γ, our current implementation follows a rather simple strategy: We optimize these parameters on a training set, doing exhaustive search on a finite two-dimensional grid. Needless to say, this strategy is not efficient and shall hence be replaced by more sophisticated optimization techniques in the future.

4. Evaluation

We evaluated our algorithm with two datasets. The first is the banksearch dataset [20], consisting of 11000 web pages in a 3 level hierarchy (see Fig. 3). The second is a part of the Open Directory [8], consisting of 8132 web pages in a hierarchy of depth up to 5 (see Fig. 4). The number of child nodes varies from 2 to 17. We crawled the data in

April 2006 and selected the categories presented in Fig. 4. Small subcategories were merged into their parent category.

- **Banking & Finance** (0 docs)
  - Commercial Banks (1000 docs)
  - Building Societies (1000 docs)
  - Insurance Agencies (1000 docs)
- **Programming Languages** (0 docs)
  - Java (1000 docs)
  - C/C++ (1000 docs)
  - Visual Basic (1000 docs)
- **Science** (0 docs)
  - Astronomy (1000 docs)
  - Biology (1000 docs)
- **Sport** (1000 docs)
  - Soccer (1000 docs)
  - Motor Sport (1000 docs)

Figure 3. The Banksearch Dataset

Our two datasets have quite different characteristics. The banksearch dataset has a rather shallow hierarchy and its classes can be separated quite well (in a classification sense). It was created by researchers for evaluation purposes only. On the other hand, the Open Directory data set is far more difficult. The data was structured by different people without any intention to provide a dataset which is easy to classify (by machine learning methods). In other words, it is truly a "real world" dataset, which makes it clearly more interesting from a practical point of view.

4.1. Methodology

All documents were preprocessed. After parsing the documents, we filtered out terms that occurred in less than five documents, in almost all documents ($> |D| - 5$), stop words, terms with less than four characters, and terms containing numbers. After that, we selected out of these the 100 most distinctive features per class as described in [2]. Each document that had an empty feature vector after this preprocessing was ignored.

We trained different classifiers to compare their results. Each classifier was evaluated five times on five randomly chosen training sets. The mean values and standard deviation over these five runs are presented. Each classifier was learned with the same five training sets for better comparability of the results. For the banksearch data, we have chosen 300 randomly selected documents from each class to form a training set. For the Open Directory data, we have randomly chosen two third of the data in each class. The classifiers were tested with the remainder of the data.

As baseline of the evaluation of our algorithm, we used two standard flat algorithms, a Naïve Bayes classifier and a SVM classifier. However, we like to emphasize again that the use of any classifier that provide probability estimates for class assignments is possible. More specifically, it is also possible to use a hierarchical classifier and further enhance its classification by our method. The SVM implementation was done based on the libSVM [6] tool. We used the linear kernel with its default settings. The Naïve Bayes implementation computes the (not normalized) probabilities $\overline{P}(c|d)$ for a class c given the document d by the following formula:

$$\begin{aligned} \overline{P}(c|d) &= P(d|c) \cdot P(c) \\ &= P(c) \cdot \prod_{t \in dict} \left(P(t|c)^{w_{t,d}} \cdot (1 - P(t|c))^{1 - w_{t,d}} \right) \end{aligned}$$

where $w_{t,d}$ is the boolean term weight of term t from the dictionary $dict$ for document d. The probability of a class $P(c)$ is estimated by the percentage of documents from the collection belonging to this class. The probability of a term given the class $P(t|c)$ is estimated with the percentage of documents from the class that contain the term.

4.2. Performance Measures

To compare results between different algorithms, it is necessary to define appropriate performance measures. For standard (flat) classification, evaluation is mostly done by precision and recall [15]. The precision of a class c is the fraction of all documents retrieved for this class that are correctly retrieved (see Eq. 8), while the recall of c is the fraction of all documents belonging to this class that are actually retrieved (see Eq. 9). The combination of the two, the F-Score, is usually used to evaluate overall performance and describes the trade-off between precision and recall (see Eq. 10). Furthermore, the accuracy is often determined, which gives the percentage of correctly classified documents (see Eq. 11).

$$prec_c = \frac{|relevant_c \cap retrieved_c|}{|retrieved_c|} \tag{8}$$

$$rec_c = \frac{|relevant_c \cap retrieved_c|}{|relevant_c|} \tag{9}$$

$$F_c = \frac{2}{1/rec_c + 1/prec_c} \tag{10}$$

$$acc = \frac{|\{d \in D | \text{predClass}(d) = \text{class}(d)\}|}{|D|} \tag{11}$$

These measures treat all classes equally. There is just one correct class and all others are wrong. However, as we already argued, in hierarchical classification, not all "wrong" classifications are equally "bad". Therefore, as indicated in Sect. 3.3, we reutilize our utility definition to better describe the quality of a prediction. This idea is combined with the

- **Fitness** (124)
  - Certification (38)
  - Gyms (4)
    - Europe (88)
    - North_America (0)
      - Canada (35)
      - United_States (351)
    - Oceania (32)
  - Personal_Training (86)
  - Pilates_Method (55)
  - Services (31)
- **Society** (0)
  - Activism (131)
    - Anti-Corporation (75)
    - Internet (35)
    - In_Daily_Life (49)
    - Media (50)
      - Culture_Jamming (215)
      - Radio (149)
    - Nonviolence (55)
    - Regional (180)
    - Resources (41)

- **Society** (cont.)
  - Paranormal (144)
    - Bermuda_Triangle (32)
    - Crop_Circles (87)
    - Ghosts (47)
      - Investigators (214)
      - Personal_Pages (47)
      - Places_and_Hauntings (68)
      - Stories (39)
    - Personal_Pages (83)
    - Prophecies (81)
    - Psychic (75)
      - Animals (60)
      - Entertainers (30)
      - ESP (59)
      - Healers (28)
      - Ouija (86)
      - Personal_Pages (30)
      - Readings (444)
      - Teaching (56)
    - UFOs (282)

- **Travel** (26)
  - Guides_and_Directories (115)
  - Image_Galleries (149)
  - Lodging (13)
    - Bed_and_Breakfast (18)
    - Consolidators (65)
    - Directories (19)
    - Home_Exchanges (11)
    - Hospitality_Clubs (15)
    - Hostels (24)
      - Africa (31)
      - Asia (16)
      - Europe (95)
      - North_America (14)
      - Oceania (72)
    - Hotels_and_Motels (26)
    - Vacation_Rentals (73)
  - Preparation (37)
    - Currencies (22)
    - Health (84)
    - Passports_and_Visas (84)
  - Publications (225)

- **Travel** (cont.)
  - Specialty_Travel (246)
    - Adventure_and_Sports (215)
    - Archaeology (41)
    - Arts (126)
    - Backpacking (81)
    - Battlefields (38)
    - Boat_Charters (677)
    - Corporate (130)
    - Cruises (289)
    - Culinary (114)
    - Ecotourism (253)
    - Educational (40)
    - Family (45)
    - Pilgrimage (22)
    - Rail (41)
    - Spas (86)
    - Students (91)
    - Volunteering (137)
  - Transportation (14)
    - Air (259)
    - Car_Rentals (33)
    - Limousines_and_Shuttles (104)

Figure 4. The Open Directory Dataset; the number behind the category name indicates the number of documents directly assigned to this category.

standard measures presented to get a performance measure that can take the hierarchy information and user preferences into account.

The performance measures given above can be interpreted as measuring a very simple notion of the utility of a prediction. A correct prediction has the utility of 1, while every other prediction receives a utility of 0. However, this is not appropriate in our setting. Instead, we want to integrate the utility function defined in Eq. (7), as this corresponds to the user's notion of the utility of a prediction. With it, the standard performance measures can be generalized to equations (12) - (14). In general, every utility function producing values between 0 and 1 could be used. Nevertheless, we use for our evaluation the utility defined by Eq. (7).

$$prec_{h,c} = \frac{\sum_{d \in retrieved_c} \text{util}(c|\text{class}(d))}{|retrieved_c|} \tag{12}$$

$$rec_{h,c} = \frac{\sum_{d \in relevant_c} \text{util}(\text{predClass}(d)|c)}{|relevant_c|} \tag{13}$$

$$acc_h = \frac{\sum_{d \in D} \text{util}(\text{predClass}(d)|\text{class}(d))}{|D|} \tag{14}$$

Other researchers also proposed hierarchical performance measures, e.g. in [21]. However, their focus is more on evaluating the classifier performance itself, e.g. by taking category similarities or tree distances into account. Our focus is on the usefulness of the classification from a user's point of view, which is based on user behavior in the described application scenario.

Besides these performance measures, we further determined some statistics which allow for distinguishing different types of misclassification and, hence, to get an idea about the behavior of the different classification methods. The following statistics were determined:

(a) $\#n_c$ – number of predictions in the correct node

(b) $\overline{\#n_c}$ – predictions in a node that is a parent node of the correct node, i.e., predictions on the retrieval path but not the correct node itself

(c) $\overline{ml(n_c)}$ – average number of hierarchy levels between the correct node and the predicted node in case of (b)

(d) $\underline{\#n_c}$ – predictions of nodes not on the retrieval path, i.e., not counted for $\#n_c$ and $\overline{\#n_c}$

(e) $\underline{ml(n_c)}$ – average number of hierarchy levels between the retrieval path and the predicted node in case of (d)

4.3. Results

Tables 1 and 2 summarize our results in terms of the above evaluation measures. Given are the results for the standard classifiers and our expected utility-based approach (HUClass). As can be seen, our approach generally achieves significant improvements for all performance measures on both data sets.

Interestingly, the parameter settings providing the best results are quite different for each combination of classifier and data set. Noticeable, an extremely large penalty term L was used in the combination with Naïve Bayes. This can be attributed to the quality of the probability estimates of the Naïve Bayes classifier which is known to be rather poor (including many estimates very close to 0). As already explained above, implementing a cautious strategy by increasing the penalty for leaving the retrieval path is reasonable in this situation. And indeed, looking at the misclassification statistics, it can be seen that HUClass(NB) often stops at higher-level nodes. As an aside, note that poor probability estimates do not imply a low classification rate, as the classification remains correct as long as the true class receives the highest probability. In fact, the classification rate of the Naïve Bayes is not so bad, even though it is significantly lower then the rate of the SVM.

As an example of the effect of the expected utility-based classification, the classification tree of documents from the Sport class of the Banksearch dataset is shown in Fig. 5. As can be seen, most documents that had been classified in too specific classes by the pure Naïve Bayes approach (87 documents in *Soccer* and *MS*) were correctly moved up the hierarchy by the HUClass(NB) approach and only 15 documents remained in the too specific classes. Furthermore, 8 out of 25 documents classified in nodes not on the retrieval path had been moved up to the root node. This effect occurs for all classes as can be seen by the $\underline{\#n_c}$ values in Tab. 1. Furthermore, accuracy as well as precision and f-measure could be improved. If we assume the search strategy of a user as discussed in the introduction, a user would now be able to retrieve on average 713.2 documents more ($\underline{\#n_c}(NB) - \underline{\#n_c}(NB{+}EU) = 1376.4 - 594.6 = 713.2$). This is 9.4% of the whole data. However, this absolute number only indicates the number of documents that are now on the retrieval path, but neglects the additional effort of a user to find them as considered by our measures defined in equations (12) - (14). The improvement for the SVM classifier is not as significant, but also here all measure show a slight performance improvement.

By comparing the performance on the Open Directory data in Table 2 to the results on the banksearch data, one can find that the improvements gained by the HUClass method are larger for all evaluated settings. This was also expected by us as this dataset is a lot more difficult. In specific, the statistics show, e.g., that with HUClass(NB) 29.1% more of the data can be retrieved, in comparison to the 9.4% for the banksearch data. In general, double or more performance gain compared to the banksearch data set can be noted.

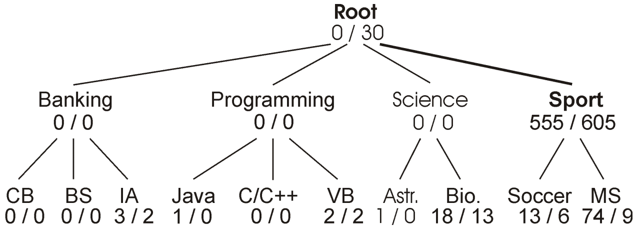

Figure 5. Classification of the Banksearch Sport Documents with Naïve Bayes / HUClass(NB).

5. Conclusions

In this paper, we presented a user-oriented hierarchical classification approach, which is based on a decision-theoretic framework. More specifically, we make use of a utility function to capture the user's search behavior. This function assigns a utility degree to every node of the class hierarchy that depends on the true classification. Given probability estimates for the different classes, our final prediction is then guided by the principle of expected utility maximization. The empirical evaluation in the previous section has shown significant improvements over standard (flat) classification methods.

Since the performance of our approach strongly depends on the quality of the class probabilities estimated in the first step, an important aspect of future work concerns the improvement of this part of the method.

As mentioned before, our algorithm has a close connection to cost-sensitive classification. Nevertheless, we prefer viewing information retrieval as a decision making process, since decision theory provides a more general framework that allows for several interesting extensions. We conclude the paper by outlining some examples of such extensions.

First, instead of using a fixed utility function (as we have done in the experimental part), one might think of "personalizing" this function by adapting the parameters to a specific user. For example, the "laziness" of a user, as expressed by the parameter γ, could be derived from his interaction with the hierarchy. Does the user only look at the entry node or also explore higher level nodes? If so, how many? Our method could then maximize the performance based on the specific γ value learned for the user, allowing for adaptivity in the classification process. This possibility shall be explored in future work.

Table 1. Performance Results for the Banksearch Data (showing mean value and standard deviation over 5 runs)

	Naïve Bayes	HUClass(NB, 3.4, -350)	SVM	HUClass(SVM, 0.6, 0)
acc_h	0.8278 ± 0.0055	0.8460 ± 0.0031	0.9301 ± 0.0014	0.9323 ± 0.0014
$prec_h$	0.8469 ± 0.0057	0.9135 ± 0.0049	0.9305 ± 0.0013	0.9435 ± 0.0019
rec_h	0.8277 ± 0.0056	0.8462 ± 0.0032	0.9300 ± 0.0014	0.9323 ± 0.0014
f_h	0.8372 ± 0.0051	0.8786 ± 0.0039	0.9303 ± 0.0013	0.9379 ± 0.0016
$\#n_c$	6281.4 ± 41.46	5839.2 ± 34.64	7079.6 ± 10.05	6994.8 ± 11.65
$\overline{\#n_c}$ $(\overline{ml(n_c)})$	42.8 ± 3.43 (1.0 ± 0.0)	1198.2 ± 103.16 (1.42 ± 0.04)	12.6 ± 3.38 (1.0 ± 0.0)	209.6 ± 13.75 (1.36 ± 0.04)
$\underline{\#n_c}$ $(\underline{ml(n_c)})$	1295.4 ± 41.75 (1.40 ± 0.03)	582.2 ± 72.10 (1.31 ± 0.37)	527.4 ± 9.50 (1.40 ± 0.02)	415.2 ± 12.17 (1.39 ± 0.02)

Table 2. Performance Results for the Open Directory Data (showing mean value and standard deviation over 5 runs)

	Naïve Bayes	HUClass(NB, 0.6, $-615\,000$)	SVM	HUClass(SVM, 1.0, 0)
acc_h	0.4556 ± 0.0096	0.5078 ± 0.0096	0.6632 ± 0.0047	0.6786 ± 0.0082
$prec_h$	0.5753 ± 0.0243	0.6986 ± 0.0170	0.7146 ± 0.0148	0.7820 ± 0.0075
rec_h	0.3119 ± 0.0089	0.4007 ± 0.0096	0.5225 ± 0.0061	0.5508 ± 0.0038
f_h	0.4044 ± 0.0120	0.5092 ± 0.0106	0.6035 ± 0.0069	0.6463 ± 0.0050
$\#n_c$	1118.2 ± 25.69	898.6 ± 8.89	1725.0 ± 17.15	1524.4 ± 34.49
$\overline{\#n_c}$ $(\overline{ml(n_c)})$	195.0 ± 23.57 (1.25 ± 0.04)	1196.4 ± 76.06 (2.11 ± 0.08)	104.2 ± 10.61 (1.21 ± 0.05)	543.0 ± 25.98 (1.23 ± 0.02)
$\underline{\#n_c}$ $(\underline{ml(n_c)})$	1376.4 ± 32.04 (1.95 ± 0.03)	594.6 ± 78.72 (1.87 ± 0.18)	859.6 ± 10.21 (1.73 ± 0.01)	621.4 ± 12.71 (1.78 ± 0.02)

As the utility function is a modular component of our approach, it can easily be replaced by other types of functions capturing other types of user behavior. In future work, we plan to identify corresponding user strategies and extract suitable utility functions.

A more far reaching extension concerns modelling the retrieval of a document as a *sequential* decision process, that is, a process, in which several decisions have to be made in succession: From a user perspective, the first decision concerns the entry node in the hierarchy and essentially corresponds to what we studied in the current paper. What we did not consider, however, is the continuation of the search process: At every node, the user must decide whether or not to continue the search, and if so, in which direction. These decisions are made on the basis of the respective search history, i.e., the nodes that have been visited so far, and hence are not independent of each other. In particular, for making the $(i+1)$-st decision, the user has more information than for the i-th decision.

References

[1] J. Blythe. Decision-theoretic planning. *AI Magazine*, 20(2):37–54, 1999.

[2] C. Borgelt and A. Nürnberger. Fast fuzzy clustering of web page collections. In M. Nanni, M. Ceci, and M. Gori, editors, *Proc. of PKDD Workshop on Statistical Approaches for Web Mining (SAWM)*, 2004.

[3] L. Cai and T. Hofmann. Hierarchical document categorization w. support vector machines. In *Proc. of 13th ACM Conf. on Inf. and Knowl. Management*, 2004.

[4] M. Ceci and D. Malerba. Hierarchical classification of html documents with webclassii. In *Proc. of 25th Europ. Conf. on Inform. Retrieval*, 2003.

[5] N. Cesa-Bianchi, C. Gentile, A. Tironi, and L.Zaniboni. Incremental algorithms for hierarchical classification. In *Neural Information Processing Systems*, 2004.

[6] C. Chang and C. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at http://www.csie.ntu.edu.tw/˜cjlin/libsvm.

[7] B. Choi and X. Peng. Dynamic and hierarchical classification of web pages. *Online Information Review*, 28(2):139–147, 2004.

[8] Open directory project, www.dmoz.org, 2006.

[9] S. T. Dumais and H. Chen. Hierarchical classification of web content. In *Proceedings of SIGIR-00, 23rd ACM International Conference on Research and Development in Information Retrieval*, pages 256–263, 2000.

[10] C. Elkan. The foundations of cost-sensitive learning. In *Proc. IJCAI–01, 17th International Conference on Artificial Intelligence*, Stokholm, Sweden, 2001.

[11] B. D. Finetti. La prévision: Ses lois logiques, ses sources subjectives. *Annales de l'Insitut Henri Poincaré*, VII:1–68, 1937.

[12] I. Frommholz. Categorizing web documents in hierarchical catalogues. In *Proc. of the European Colloquium on Information Retrieval Research*, 2001.

[13] N. Fuhr. A decision-theoretic approach to database selection in networked IR. *ACM Transactions on Information Systems*, 17(3):229–249, 1999.

[14] M. Granitzer and P. Auer. Experiments with hierarchical text classification. In *Proc. of 9th IASTED International Conference on Artifical Intelligence*, 2005.

[15] A. Hotho, A. Nürnberger, and G. Paaß. A brief survey of text mining. *GLDV-J. for Computational Linguistics and Language Technology*, 20(1):19–62, 2005.

[16] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, 1998.

[17] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. John Wiley and Sons, 1953.

[18] F. Ramsey. Truth and probability. In *The Foundations of Mathematics and Other Logical Essays*. Kegan Paul, London, 1931.

[19] L. Savage. *The Foundations of Statistics*. John Wiley and Sons, Inc., New York, 1954.

[20] M. Sinka and D. Corne. A large benchmark dataset for web document clustering. In *Soft Computing Systems: Design, Management and Applications, Volume 87 of Frontiers in Artificial Intelligence and Applications*, pages 881–890, 2002.

[21] A. Sun and E. Lim. Hierarchical text classification and evaluation. In *Proc. of the 2001 IEEE International Conference on Data Mining*, pages 521–528, 2001.