In Proc. 4th SIAM Int. Conf. on Data Mining, pp. 102-114, Lake Buena Vista, FL, 2004

Using Support Vector Machines for Classifying Large Sets of Multi-Represented Objects *

Hans-Peter Kriegel Peer Kröger Alexey Pryakhin Matthias Schubert

Institute for Computer Science
University of Munich
Oettingenstr. 67, 80538 Munich, Germany
{kriegel,kroegerp,pryahkin,schubert}@dbs.informatik.uni-muenchen.de

Abstract

Databases are a key technology for molecular biology which is a very data intensive discipline. Since molecular biological databases are rather heterogeneous, unification and data integration is mandatory to make use of the huge amount of available information. Currently, the most promising approach for integration is the use of ontologies. Since mapping biological entities into ontologies is usually achieved manually or semi-automatically, a system for automatic classification of biological entities into ontologies saves time and effort. Therefore, we present a support vector machine based approach that automatically classifies biological entities into a given ontology. To solve this difficult task, our method copes with the following aspects. Biological entities might belong to more than one class or may be placed in classes on varying abstraction levels. An object may be described by several representations. Thus, the classifier has to be enabled to draw information from all of them, but must consider the possibility that some objects are described incompletely. Therefore, our method introduces the technique of object-adjusted weighting which regulates the impact of each representation dynamically for each object. To significantly improve the time performance of the classifier we exploit the inheritance relations of the given ontology. Our experimental evaluation on protein data and several parts of an established molecular biological ontology shows that our prototype offers impressive accuracy and is efficient enough to cope with the large number of classes encountered in real world problems.

*Supported by the German Ministery for Education, Science, Research and Technology (BMBF) under grant no. 031U112F within the BFAM (Bioinformatics for the Functional Analysis of Mammalian Genomes) project which is part of the German Genome Analysis Network (NGFN).

Keywords

support vector machines, multi-represented objects, combined classifiers, hierarchical classification, multi-class SVMs.

1 Introduction

In recent years, the amount of publicly available biological information has increased dramatically. As a consequence, many databases have emerged, offering diverse information on all kinds of biological entities such as proteins, nucleotides, pathways, etc. Though most of these information sources are accessible via the web, the use of the information is strongly limited due to the heterogeneity of data formats, data models, and access facilities between these sources [3].

Currently, the most promising approach for overcoming these problems is the use of ontologies and taxonomies for data integration. Several ontologies have been developed for molecular biology but only a small fraction of them is widely accepted. One of the most popular ontologies in molecular biology is Gene Ontology (GO) [4] which models the function of genes and gene products (e.g. proteins). Most of the major protein databases (such as SWISS-PROT [2]) provide a mapping of their entries to GO. However, not all of the entries are already mapped and biologists all over the world produce new entries every day. To obtain a mapping of a so-far unlinked protein database entity into GO, usually some information about the biological function of the protein, representing this entry, has to be explored. Since this usually has to be done manually throughout a series of biological experiments and tests, it is a very time consuming and costly task. It would be of great benefit, if the mapping could be done automatically by computer-supported prediction of the biological function out of the raw data stored in the major protein databases (e.g. the amino acid sequence of

a protein which can be obtained very easily) without laborious experiments. More generally, a framework for the automatic prediction of the function of biological entities such as proteins is needed that classifies these entities according to ontologies such as GO.

In this paper, we introduce a classification system that provides a good mapping for so far unlinked biological entities and thus gives a prediction of the biological function of these entities. The fact that the objects are not linked to classes or other entities yet, restricts the use of general relationships modelled in the ontology. Thus, our system exploits the class inheritance of the ontology for classification, i.e. the taxonomic part of it.

Due to the nature of biological entities, our system copes with the following demands: Several instances may belong to more than one class in the taxonomy. Different instances might be placed at varying abstraction levels. At last, biological ontologies may employ multiple inheritance in order to model their classes.

Another aspect is that the representations of biological entities usually can be derived from multiple sources, e.g. for most proteins, amino acid sequence data and textual descriptions of experimental results are available from databases such as SWISS-PROT [2]. For a smaller number of proteins additional data is available, e.g. the 3-dimensional foldings. Usually such data can be derived from other public databases such as the Protein Data Bank (PDB) [1].

Different sources have different views on the entities depending on their research scope. Since it is desirable that our classification system uses as much of the available information as possible, it should be able to exploit multiple representations for the same data object to improve accuracy. As the quality of each type of representation may vary for different entries and types of representations, the classifier should automatically weight the influence of each representation. For example, the textual annotations of proteins are – if existing – usually more useful for classification than the sequence data of the proteins. Last but not least, the framework should be flexible enough to handle missing representations, i.e. in the case that one of the representations of an instance is missing, the classifier should still be able to make a preditcion. Since biological ontologies are built of large numbers of classes and biological entities occur in large cardinalities, good efficiency is mandatory to handle large problems in applicable time.

To take up these challenges, we present a novel approach to hierarchical classification based on support vector machines. The main contributions of this paper are:

- An efficient and accurate method for handling multi-classified instances employing support vector machines.
- A method for classifying objects built of multiple representations that is able to cope with missing representations.
- A discussion of methods for hierarchical classification under the aspect of large classification problems and taxonomic directed acyclic graphs instead of strict taxonomy trees.
- A thorough experimental evaluation demonstrating effectiveness and efficiency of our prototype on several subsections of GO.

Our prototype was designed to automatically map proteins based on their SWISS-PROT entries into GO. We use sequence data and the text annotations as different representations for proteins. Let us note that further data sources such as secondary and tertiary structures can easily be incorporated into the prototype. The rest of the paper is organized as follows. In Section 2, we briefly review related work. In Section 3, we present the major concepts of our approach that cope with the challenges mentioned above. The proposed prototype is evaluated based on a realistic experimental setting in section 4. Section 5 offers a summary of the presented work and gives perspectives for future work.

2 Related Work

Support Vector Machines. In recent years, support vector machines (SVMs) [5] have received much attention offering superior performance in various applications [13, 8]. Basic SVMs distinguish between two classes by calculating the maximum margin hyperplane between the training examples of both given classes. The use of soft margins and kernel functions enables them to classify any kind of data. To employ SVMs for distinguishing more than two classes, several approaches were introduced [22].

Biological Sequence Classification. Classifying biological sequences such as nucleotide or protein sequences is an active area of research. Common approaches are based on k-nearest neighbor classifiers (KNN) and all kinds of Markov models (MM) including simple MM, selective MM and higher-order MM. Examples of KNN and MM approaches are given by [11, 21]. KNN-methods for biological sequence classification usually use edit distance as similarity functions. Although being simple and comprehensible to biologists, these approaches suffer from the expensive computation. MM are widely used for biological sequence classification since they have an inherent ability to model sequential constraints in the data. Recently,

SVMs have been applied to sequence classification by [8] and demonstrated excellent accuracy when a suitable feature extraction method is employed. To be suitable for sequence classification an extraction method should model the sequential nature of the data. Since finding such an adequate extraction is not a trivial task, recent research addresses this challenge, e.g. [16, 19, 26, 8].

Text Classification. The classification of text documents has been investigated by the research community in the areas of information retrieval, machine learning and data mining. To process a text document, most algorithms rely on projecting the documents into the feature space of relevant terms. Thus, documents are described by a vector of term frequencies (TF) or a vector of term frequencies weighted by the inverse document frequency (TFIDF) [24]. To classify the gained vectors all established classification methods are applicable, as long as they can handle the large number of features common to text applications. Since several thousand dimensions are not uncommon in text applications and the sparseness of the feature vectors used to represent documents (most of the dimensions show a zero count for most of the documents), several approaches to classify text have been introduced [12, 28, 23]. SVMs have also demonstrated their high value for making very accurate predictions in the field of text classification [13].

Hierarchical Multi-Classification. Employing class hierarchies to improve large scale classification problems has predominantly been used in text classification. Therefore, the used taxonomies are taken from directory services for HTML documents [20, 10], structural class systems like the U.S. patent codes or are constructed to have a proper test bed [18]. The idea of hierarchical classification is that solving a set of small problems with less classes can be achieved faster and more effective than solving one large scale classification problem distinguishing a large amount of classes. To do so, several approaches have been introduced [20, 10, 18, 15, 7, 25, 27]. Most of them achieved a big performance improvement and some gain in classification accuracy. However, none of these approaches examined an arbitrary shaped class system of functional data types employing multiple inheritance so far. Furthermore, only [10] examines the combination of support vector machines and class hierarchies so far, but the evaluation is based on a strict two level taxonomy tree. There have been several approaches to employ general ontologies for classification via relational learning like [6]. However, since those approaches rely on general relations, the problem they try to solve is dissimilar to the challenge we want to take up.

Classifier Fusion. The task of learning from objects given by multiple representations has recently drawn some attention in the pattern recognition community [14, 9, 17]. In [9] the author describes the method of classifier fusion to combine the results from multiple classifiers for one and the same object. Furthermore, [9] surveys the four basic combination methods and introduces a combined learner to achieve combination rules offering better accuracy. However, the introduced method is not discussed under a data mining point of view. Thus, the described method does not treat two major demands of our system, scalability and the handling of incomplete objects.

3 Classification of Biological Objects

In this paper, we address the problem of classifying biological objects like genes or proteins into a large class system like Gene Ontology [4].

The goal of classification is to learn a function $F : O \rightarrow C$ that maps as much objects $o \in O$ to their correct class $c \in C$ as possible. For training, a set of tuples (o, c), of objects o and their correct classes c is given to the classifier – the so-called training set. A variant of simple classification is multi-classification $F_{multi} : O \rightarrow 2^C$ which maps each object o to a subset of C. In our application multi-classification is mandatory, since large parts of the objects are associated with more than one class.

In most application domains, the objects are represented by one (possibly complex) data type. Thus, the established way to process the objects $o \in O$ is to extract a set of meaningful features from the object representation. For most kinds of data representations, e.g. text, there already exist several approaches of feature extraction. The derived features span the so-called feature space. Each dimension of the feature space represents a feature. Thus, an object o is represented by a feature vector. A classifier is trained on the set of feature vectors derived from the training set.

As stated above, biological entities are often built of multiple data types (representations), such as sequence data, text, etc. Thus, the input space O of our classifier F (analogously for F_{multi}) is structured, i.e. the structure of O is determined by the set of different representations an object in O might have. Though building one joined feature space for all different representations like texts or sequences is principally applicable, the corresponding feature vectors might mirror the properties of the data object only poorly. We argue that classification benefits from incorporating the knowledge about the structure of the input space – i.e. the knowledge about the representation a feature is extracted from – into the classifier. In other words, features from the

same representation should be treated in a more similar way than those from different representations.

A second property of biological entities could be utilized to enhance the performance of our prototype. Not only the input space O but also the output space C of F (analogously for F_{multi}) is structured. In fact, the classes in C are usually organized in a sophisticated class system like a taxonomy or an ontology (in our case the GO). To solve a classification problem it is not necessary to consider any relations between the classes $c_i \in C$. But the fact that the classes in C are structured can be exploited for dealing with large cardinalities of C more efficiently.

In the following, we will first introduce an approach for multi-classification based on Support Vector Machines. Afterwards, we will integrate the knowledge about varying object representations (i.e. the structure of the input space) to enable the accurate incorporation of as much information about the objects as possible into the classification process. Finally, we will discuss hierarchical classification utilizing the structure of the output space in order to enhance the performance of our framework.

3.1 Using Support Vector Machines for Making Set Valued Predictions

Support Vector Machines (SVMs) are capable of providing superior accuracy compared to other classification methods for most representations of biological objects [13, 8]. Standard SVMs (or binary SVMs) classifiy objects into two classes by finding a hyperplane that separates a given training set according to these classes in a best possible way. Since SVMs are only capable of making binary decisions, it is necessary to enhance them to distinguish between more than two classes and to make set-valued predictions. In [22] three methods for implementing SVMs are compared that distinguish more than two classes. The first, the so-called **one-versus-rest** approach employs one binary SVM for each class to decide, if an object belongs to that class or not. Thus, this approach is capable to predict any class combinations possible. For example, an object could be mapped to all classes, if all binary SVMs decide for the class they distinguish from the rest of the classes. The second approach, the so-called **one-versus-one** approach uses $\frac{N \cdot (N-1)}{2}$ many binary SVMs for distinguishing between each pair of the given $N = |C|$ classes. When classifying an object, the results are aggregated into a so-called **voting vector**. This vector provides a dimension for each class and its components correspond to the number of binary SVMs that have predicted this class. Thus, there are $\frac{N \cdot (N-1)}{2}$ votes. Since there are only $N-1$ binary SVMs distinguishing a class from the other classes, a class can attain a maximum of $N-1$ votes. A single class decision is accomplished by returning the class having the maximum number of votes in the vector. Note that this maximum vote is not necessarily $N-1$. The third described approach uses decision directed acyclic graphs to reduce the number of binary SVMs considered for classification to $N-1$ out of $\frac{N \cdot (N-1)}{2}$ trained SVMs. The approach trains the same number of SVMs, but does employ only a fraction of them for classification.

We choose the one-versus-one approach for our system, since voting vectors provide a meaningful intermediate result. To enable the approach to predict a set of class combinations, we exploit a characteristic of our application. Since there are several class combinations that do not make sense, we can limit the set of valid class combinations. For example, an object that is predicted to be a dog is unlikely to be a cat at the same time. Therefore, we collect the set of combinations that occur within the training data. For classification we extend the set of original classes by those class combinations. Thus, all valid combinations are predictable. Let us note that the one-versus-rest approach also solves the problem of set-valued predictions, but offered inferior results due to the prediction of invalid class combinations (see Section 4 for experimental results).

A drawback of all of the mentioned approaches of multi-class SVMs is that the extra effort for introducing another class leads to the use of additional binary SVMs distinguishing the new class. To avoid this additional overhead, we do not extend the class set at once. Instead we refine the post processing of the voting vectors gained from the one-versus-one approach. Thus our classification function has the following form:

Let O be the set of objects, let C be the set of classes, let C^* be the extended class set including all valid class combinations and let V with $dim(V) = |C|$ be the vector space of voting vectors, then our classifier has the following form:

$$Cl : O \rightarrow C^*$$
$$Cl(o) = F_2(F_1(o))$$

where $F_1 : O \rightarrow V$ and $F_2 : V \rightarrow C^*$ are classifiers. For F_1 and F_2 our prototype employs a one-versus-one multi-class support vector machine.

The vector space of voting vectors is well suited for describing the results of the first classifier. A one-versus-one multi-class SVM partitions the feature space along each of its binary SVMs. A voting vector corresponds to a partition of the feature space (Figure 1). Note that those partitions might not be continuous, but are placed between a certain set of classes. Since the partitions are

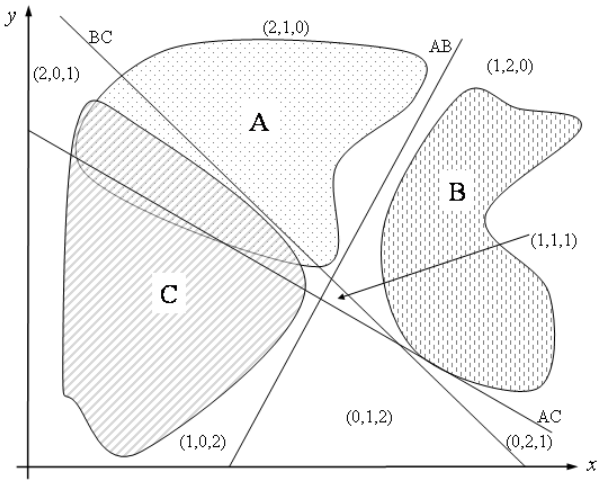

Figure 1: 3 binary SVMs distinguish the classes A, B and C. For each partition the corresponding voting vector is given. Note that voting vector (2,0,1) describes a partition where the majority of objects belongs to both classes A and C.

made to separate the objects with respect to their class, it is very likely that the majority of objects belonging to a partition belong to the same class combination.

The reason why the classifier F_2 still offers better efficiency than using just one function $F : O \rightarrow C^*$ is that usually the cardinality of the output space C (and therefore that of V) is much smaller than the number of features describing an object o. Though F_2 might employ many additional SVMs, in most cases the method offers a performance benefit, due to the much simpler input space V.

3.2 Structuring the Input Space

An important aspect of the system we propose is that the classification of a biological object should be based on all available information about it. Thus, the classifier should be able to use as much different representations as possible or if the object is described in more than one form, it should use all available representations. For proteins common representations are describing text, sequence data, secondary structure and 3D foldings. Our prototype uses text and sequence data, but can be extended easily to additional representations.

To extract features from each representation, there are several standard techniques for each kind of representation (see Section 2). Thus, the first step is to extract features from the objects in the training set for each representation. As mentioned before a simple solution is given by building up one feature space incorporating the features drawn from all the representations, but for the following reasons a more sophisticated approach offers better results. The number of features best suitable for each representation yields an unbalanced weighting of the impact of each representation. For example, the number of features used for a suitable text representation might be orders of magnitude higher than those used for 3D foldings. Thus, most classifiers will favor the representation providing more features instead of the representation carrying more information. Furthermore, the techniques proposed to handle different representations vary in the parametrization of the classifiers e.g. the SVM for text and sequence may use different kernel functions to distinguish the objects. By using a combined feature space, we are forced to find a compromise for these tuning decisions that might not offer optimal results. Last but not least the handling of missing representations of a data object is difficult, since the classifier expects at least some values in the missing dimensions of the input space.

As a consequence our classification system considers varying representations separately. The idea is that each data source is handled by some specialized classifier first. Afterwards the results are combined to build up a prediction for the object.

Thus our classifier has the following form: Let $O = R_1 \times .. \times R_n$ be the set of objects $o = (r_1, \ldots, r_n)$ represented by an n-tuple of feature vectors $r_1, \ldots, r_n$ drawn from the single representations $R_1, \ldots, R_n$, let C be the set of classes, let C^* be the extended class set including all valid class combinations and let V with $dim(V) = |C|$ be the vector space of voting vectors, then our classifier has the following form:

$$Cl : O \rightarrow C^*$$
$$Cl((r_1, .., r_n)) = F_2(comb(F_{1,1}(r_1)), .., F_{1,n}(r_n))$$

where $F_{1,j} : R_j \rightarrow V$, $F_2 : V \rightarrow C^*$ and $comb : 2^V \rightarrow V$.

Each of the feature vectors r_j is classified by a specialized classifier $F_{1,j}$ into a voting vector. The function *comb* combines the voting vectors of each available representation into one common voting vector which is afterwards mapped into the expanded class space C^* by F_2.

Due to this design each representation can be classified in a best possible way by a specially tuned SVM. No initial weighting is made by the number of features for one representation. Last but not least, missing representations can be handled by a properly designed combination function. Since the combination function is designed to handle an input of j voting vectors with $1 \leq j \leq n$ and generates an output vector that is independent from j, missing representations are process-

able. Note that though the missing representations can be processed, the quality of the prediction is still likely to suffer depending on the significance of the remaining descriptions.

Our general combination function has the following form:

$comb : 2^V \rightarrow V$, where

$$comb(o) = \begin{pmatrix} f_1(r_{1,1}, \ldots, r_{1,m}) \\ \vdots \\ f_N(r_{N,1}, \ldots, r_{N,m}) \end{pmatrix}$$

and V is the feature space of voting vectors for N base classes and f is a normalized function to combine the components of the m input vectors, where $1 \leq m \leq n$ and n is the number of representations. Common choices for f are the minimum, the product, the sum and the maximum, where the sum and the product have to be normalized by m. [9] offers a survey which of those 4 strategies is suited best for which kind of object. Furthermore, [9] introduces the idea of employing an additional learner to improve predictions. This idea is kept up by our second classifier as long as it does not collide with the requirement of handling objects with missing representations. As a result, we loose the possibility to consider correlations between votes for different classes drawn from different representations.

Since the results achieved by employing the methods described in [9] were not capable to improve accuracy, we introduce a weighted strategy to achieve much better results. The main problem of the basic strategies is that each data source always has the same impact on the result. To model the influence of different data sources, we introduce weight factors for each representation j and each object o. These weight factors should reflect the following aspect: how confident is a specialized classifier $F_{1,j}$ about the voting vector it produced for a special feature vector r_j. Our rule for calculating the components of the general voting vector is:

$$f_i(r_{i,1}, \ldots, r_{i,m}) = \frac{\sum_{j=1}^{m} w_{r_j} \cdot (F_{1,j}(r_j)_i)}{m}$$

where w_{r_j} is a weight describing the confidence of the prediction derived from $F_{1,j}$ for r_j and $F_{1,j}(r_j)_i$ is the i-th component of the voting vector derived from the j-th data source. Note that we choose the sum-function as base combination strategy, since all data sources should contribute to the result.

To derive meaningful weights, we use an established method for deriving confidence values for binary SVMs. This method calculates the distance of the feature vector to the separating hyperplane. The idea is that the closer the feature vector is, the less confident is the prediction. This is based on the characteristic of SVMs that objects which are difficult to decide are placed in the surrounding of the hyperplane. To derive confidence values and to model that after a certain distance to the separating hyperplane the decision is considered as secure, a sigmoid function is usually applied to the distance. Furthermore, the closer surrounding of the hyperplane is treated in a more sensitive way. Thus the confidence $conf$ of a SVM svm is given by:

$$conf_{svm}(o) = \frac{1}{1 - e^{\alpha \cdot svmdist(o)}}$$

for object o, $svmdist(o)$ the distance of o to the separating hyperplane of svm and α a parameter for regulating the sensitivity.

Since our systems employ multi-class SVMs that usually employ more than just one binary SVM, the process of deriving a proper weight has to consider several distances. Therefore, we determine the class having the maximum vote in the voting vector derived from one data source. For this class, we determine the minimum confidence value belonging to the SVMs that characterize the predicted class (cf. Figure 2).

Let $F_{1,j}$ be the multi-class SVM treating the representation j. Then $F_{1,j}$ is built from the following matrix of binary SVMs:

$$F_{1,j} = \begin{pmatrix} - & svm_{1,2} & \ldots & svm_{1,N} \\ svm_{2,1} & - & svm_{2,3} & \ldots \\ \ldots & \ldots & \ddots & \ldots \\ svm_{N,1} & \ldots & svm_{N,N-1} & - \end{pmatrix}$$

Note that this matrix of SVMs is symmetric, since the classifier distinguishing i from j is the same as the one distinguishing j from i. Then we determine the weight in the following way:

$$w_{r_j} = \min_{svm_{i,maxdim(v_j)} \in F_{1,j}} conf_{svm_{i,maxdim(v_j)}}(r_j)$$

where v_j is the voting vector derived by $F_{1,j}$ for r_j and $maxdim(v_j)$ is the class in v_j having the maximum number of votes.

The idea is that the class having the maximum count is most likely part of the prediction. If the feature vector is predicted with a high confidence value, it needs to have a sufficient distance from any of the other classes. Afterwards the weights are normalized and used in $comb$ as described above. Thus, classifiers offering highly reliable results have significantly more impact on the resulting voting vector. Since the weights are calculated for every single instance to be classified, our combination function adjusts to the current object

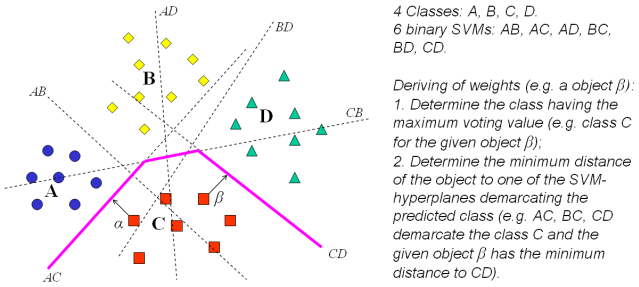

Figure 2: Illustration of the class confidence estimation (see text for details).

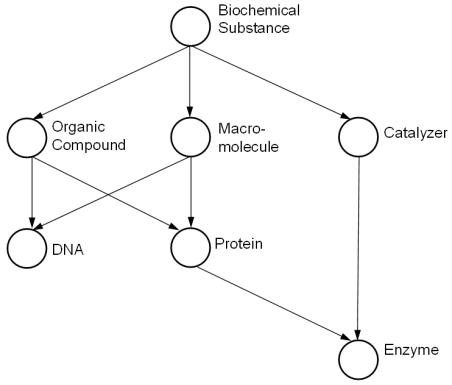

Figure 3: A sample TDAG.

and does not prejudge complete representations. Thus each object is predicted on the representation that is most significant for the current task. Therefore, we call this new method *object-adjusted weighting*.

3.3 Structuring the Output Space

Classification into large class sets providing over 100 classes is a very time consuming task. Remember that a one-versus-one multi class SVM needs 4950 binary SVMs for 100 classes. Thus, to make the system scalable, it is necessary to find an efficient way for classification of large class problems. One way to speed up classification is to employ additional knowledge about the class set. Considering a class system like an ontology or a taxonomy not just a set of classes, opens the possibility to split the large classification problem into several smaller ones that are faster to process. Let us note that the accuracy achieved on smaller systems also tends to be significantly higher, due to the easier problem.

Ontologies are a common approach to model class information in molecular biology. Though an ontology usually models all kinds of relations, most of them are not useable for classification in our system. The problem is that the objects we want to classify, do not have any link to any other object, yet. Thus, exploiting general relations to determine the class of an object is very difficult in our application. On the other hand, we can use the inheritance relations of the ontology, because knowing that an object is part of a super-type opens up the possibility that it is part of a sub-type, too. Thus, we use the taxonomy part of the given ontology. This taxonomy varies from the majority of class hierarchies used in other projects regarding the following 3 aspects:

- Instances can be placed at varying abstraction levels. It is common to biological ontologies to collect entities not further specified in non-leaf nodes of the ontology though there might be several refinements of the class.
- It is possible that database entries may link to varying classes in the class system. Thus, we have to treat multi-classified objects belonging to one or more classes.
- A class hierarchy of an ontology might use multiple inheritance for some of its classes. This characteristic leaves us without a taxonomy tree and demands a taxonomic graph.

According to these characteristics we restrict a given ontology to a *taxonomic directed acyclic graph*. A *directed acyclic graph* (DAG) is a connected directed graph that does not contain any cycles. An entry node to a DAG is a node without any incoming edge. If there is only one entry point the node is called root and we have a rooted DAG. A *taxonomic directed acyclic graph* (TDAG) is a rooted DAG where each node is connected to a class of objects. The class of a predecessor node is a super-type to all classes of the nodes the predecessor node has edges to. Furthermore, we require that the entries belonging to the super-type are exactly the union of the entries belonging to its sub-types. Though this requirement is not fulfilled in the first place, we can easily fix it by introducing additional leaf nodes for the super-types having instances that do not belong to any of the sub-types. Thus, we get a TDAG which is our choice of class system providing a more general setting. A sample TDAG is depicted in Figure 3.

To find out which method of hierarchical classification is best suited for exploiting TDAGs, we will discuss two basic approaches and their ability to support our setting.

The basic approach of hierarchical classification is to decompose a flat N class problem to several smaller problems of sizes $n_i << N$. Thus, common

hierarchical classifiers are class hierarchies where each super-type provides a classifier that predicts the sub-types a given object belongs to. The idea is that these smaller problems are easier and faster to decide than one big problem. The differences between the majority of introduced methods for hierarchical classification are mostly within the part of the class system that is traversed during classification. Principally, there are two strategies to tackle the problem:

- The probabilities (or a combination of classifier outputs) are considered for each leaf in the class hierarchy. Thus, the whole class hierarchy is visited and leaves getting smaller confidence values by the top-level classifiers, might still be considered, if the classifiers are confident on the rest of their decision paths.
- Step by step, at each level the sub-classes that are considered unlikely are pruned. Thus, only a small portion of the classifiers in the system is employed for classification.

The first approach tries to achieve the best possible accuracy while the second approach offers better efficiency, but might loose accuracy due to its restrictiveness. Thus, the second approach is favorable for our target to employ large TDAGs providing over 100 classes, if accuracy does not suffer too much. Further reasons for employing the second approach to achieve classification into large TDAGs are:

- The occurrence of multiple inheritance and leaves on varying abstraction levels makes it computationally demanding to calculate comparable probabilities for all leaves. To achieve such a calculation implies knowledge about all pathes leading to a leaf. Furthermore, the fact that leaves are placed at different abstraction levels requires proper normalization of the probabilities.
- Employing classifiers that do not consider the possibility that the object belongs to none of its classes, might generate confidence values that do not reflect a realistic estimation. Figure 4 shows an example of a hierarchical classifier based on SVMs employing the distance to the hyperplane as confidence value. In the described case, an object is misclassified due to an unrealisticly high second level confidence value.
- The possibility of multiple pathes leading to a class is able to compensate a wrong decision in the second approach. If one path to reach a class is pruned, it still might be reachable via another path in the TDAG.

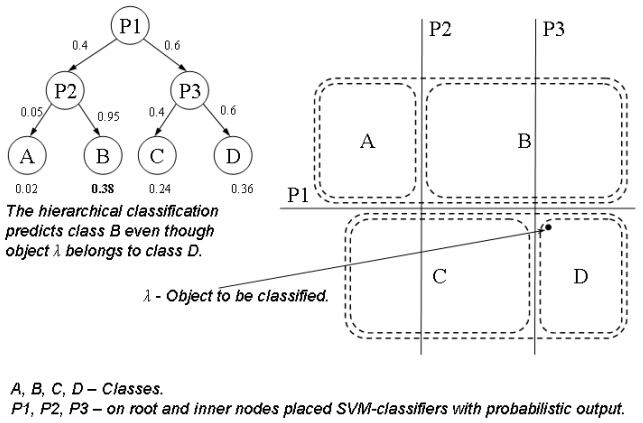

Figure 4: Example for a wrong decision due to a very high 2nd level confidence value.

Thus, we choose the second approach for building a classifier that employs TDAGs as a class system. Our System now consists of a TDAG organizing the classes we want to predict. At each node a classifier designed as described in the previous subsection is trained to decide the correct sub-types under the precondition that the object already belongs to the class the node is attached to. Hierarchical classification is now achieved by giving any object to the classifier at the root node and following all predicted paths until every branch of the process reaches a leaf. The set of reached leaf nodes is the prediction of the class set made by the system.

4 Experimental Evaluation

4.1 Test Bed

In order to demonstrate the advantages of our system, we carried out a versatile experimental evaluation. The experiments were performed on 5 different classification problems. The test beds consist of 17 to 107 Gene Ontology[4] classes and their "is-a" relationships. The corresponding objects were taken from the SWISS-PROT[2] protein database and consist of a describing text and the amino acid sequence of the described protein. The properties of each test bed is shown in Table 1. In order to obtain a TDAG with sufficient training objects per class, the original environment was pruned. The result of the pruning is a TDAG that fulfills the following conditions:

1. Every leaf class refers to at least *MinSupport* proteins.
2. Every inner node in the TDAG has at least *MinSonNumber* direct son classes.
3. The pruning process contains as much training

	Set 1	Set 2	Set 3	Set 4	Set 5
Name	Response to external stimulus	Protein binding activity	Receptor binding activity	Oxidoreductase	Biosynthesis
Number of Goal Classes	17	19	26	94	107
References to proteins	1832	1166	1857	9907	1811
Multi-class Proteins (%)	5.36	13.63	14.29	17.97	20.58

Table 1: Details of the test environments

objects as possible. This condition is fulfilled by moving proteins from pruned classes to their direct parent.

The details of the classification problems are listed in Table 1.

All algorithms are implemented in Java and were tested on a work station that is equipped with a 1.4 GHz Pentium IV processor and 2 GB main memory. To measure the accuracy for multi-classified objects we used the following definition of classification accuracy:

$$Accuracy = 1 - \frac{\sum_{o \in T} (|(A(o) \cup B(o)) - (A(o) \cap B(o))|)}{\sum_{o \in T} |A(o)| + |B(o)|}$$

where o is a test object, T is the set of test objects, $A(o)$ is the correct class set for object o and $B(o)$ is the predicted class set of object o. In order to avoid overfitting, the evaluation used 10-fold cross-validation.

To classify protein sequences, we employed the approach described in [8]. The basic idea is to use local (20 amino acids) and global (6 exchange groups) characteristics of a protein sequence. To construct a meaningful feature space, we formed all possible 2-grams for each kind of characteristic, which provided us the 436 dimensions of our sequence feature space. For text descriptions, we employed a TFIDF vector for each description that was built of 100 extracted terms. Both representations were classified employing a degree 2 polynomial kernel. Due to the superior results of the described hierarchical approach, all of the following experiments use a structured output space with the exception of the flat classifier approach. The feature selections were applied to each node separately as described in [15].

4.2 Experimental Results

To show that the one-versus-rest approach is not suitable for our application, we compared its accuracy on the text descriptions to the one-versus-one approach. Since it offered significantly inferior results to the settings employing an extended class set and the one-versus-one approach (4.49% - 12.01% less accuracy), we did not follow this approach any further (cf. Figure 5). For example the classification accuracy achieved for the Set 4 test bed by the one-versus-one strategy was 82.12% , whereas the one-versus-rest approach only reached 70.11%.

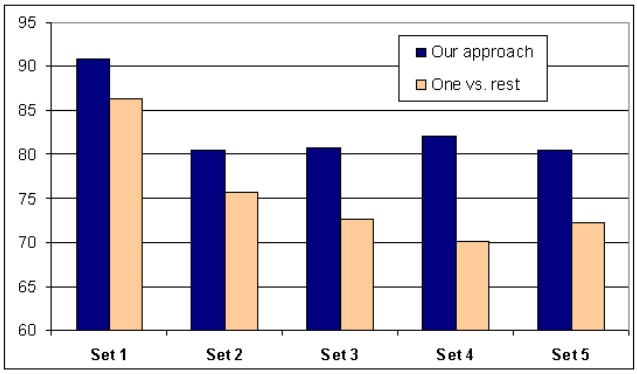

Figure 5: Classification accuracy (in %) of our method compared to the one-versus-rest approach.

Our second experiment demonstrates that a two-step classifier offers better results compared to a single classifier using a direct extension of the class set (cf. Section 3). The two-step approach achieved comparable accuracy and superior efficiency for all test sets (cf. Figure 6). In particular, our approach showed for Set 5 with 107 goal classes the classification accuracy of 81.37% and took an average of 1.75 seconds as classification time per object. The competing method using only one classifier and a direct extension of the class set achieved ca. 1 % less classification accuracy and was evidently slower - 2.66 seconds as average classification time per object. According to our results the two-step approach improved both efficiency and effectiveness of the classifier.

In order to show the advantages of the hierarchical approach against an unstructured class system, we compared both approaches for the introduced classifier

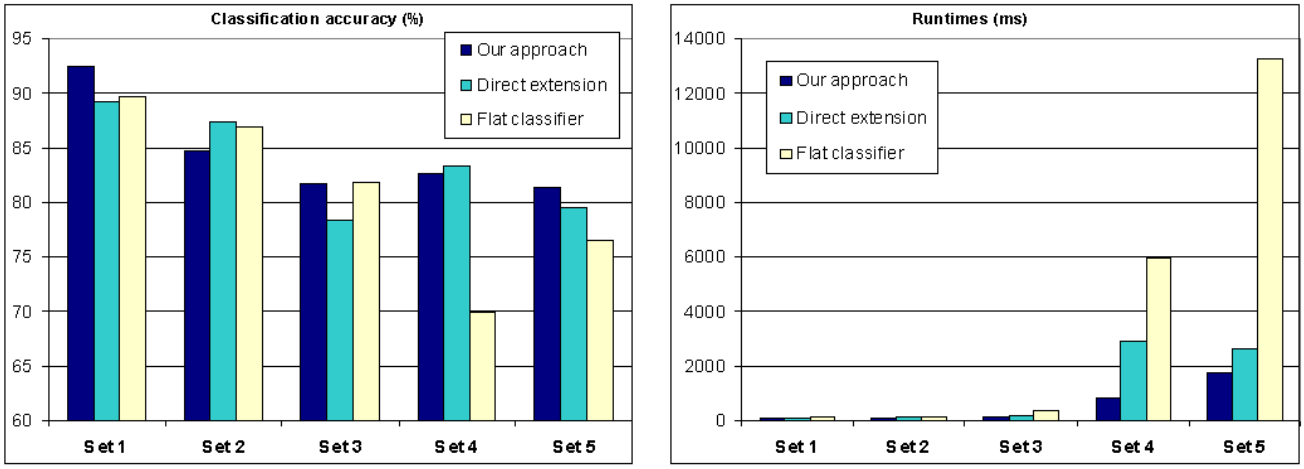

Figure 6: Accuracy and runtime for hierarchical classification employing a 1-versus-1 SVM with an extended class set (direct extension) and two subsequent 1-versus-1 SVMs (our approach). Additionally, we compare our approach without using a TDAG (flat classifier).

on both representations. We observed better accuracy in most cases and an enormous improvement in classification time, especially when working with large class systems (cf. Figure 6). In case of Set 4 providing 94 target classes the flat-classifier achieved 69.92% accuracy and took an average of 5.97 seconds for the classification of an object. The hierarchical approach achieved on the same data significantly higher accuracy (82.65%) and needed only 0.85 seconds per object. Thus, hierarchical classification was processed up to 7 times faster than flat classification. Note that this considerable speed up was achieved especially in the large TDAGs where the performance is much more critical than in smaller problems. Furthermore, the classification accuracy surpassed the accuracy observed for the other approaches in the majority of test sets.

The next experiment compares the use of a structured input space for classification. Therefore, we compared the accuracy achieved by employing only the text part, only the sequence part, a combined feature space that incorporates the features of both representations and our combined classifier. The combined classifier was evaluated with and without object-adjusted weights (cf. Table 2). In all cases, the classification of text was more accurate than that of sequence data. Furthermore, the combination without object-adjusted weights and the variant employing a combined feature space, were not capable to improve accuracy towards the text description in all cases. Thus, it would be more promising to restrict the classifier to employ text descriptions only. The variant that employs the object-adjusted weighting on the other hand increases accuracy in all 5 test beds up to 4 %. Thus, it was the only examined method that was able to dynamically decide which representation is suited best and draw advantages from all representations.

Our last experiment examines the capability of the system to cope with incomplete data objects. Therefore, we trained the classifiers on both data sources and tested them by classifying the sequence part of the test instances only. For the majority of test beds it turned out that the accuracy approximately reached the level achieved by classifying the sequence data alone (see last line of Table 2). In the case of Set 5 the classification accuracy of 73.41% even exceeded the values observed for sequences only (71.09%). Thus, the system is able to handle incomplete data. Let us note that this capability gets more and more important with an increasing number of representations, since it is getting very demanding to train classifiers that can handle the remaining representations best possible with increasing numbers of representations. Furthermore, when incorporating several representations, the remaining representations are more likely to compensate the missing information.

5 Conclusions

In this paper, we proposed a prototype for classifying data objects into taxonomic directed acyclic graphs and applied it to biological entities in molecular biological ontologies. Our method addresses the following problems: First, biological instances often consist of multiple representations such as sequence, text, etc. The classifi-

Method	Set 1	Set 2	Set 3	Set 4	Set 5
classification on text only	90.82	80.5	80.71	82.12	80.55
classification on sequence only	89.4	80.3	77.96	75.22	71.09
modelling multi-represented objects with one combined feature space	88.6	80.56	74.76	77.87	77.89
seperate feature spaces combined by average function	87.92	78.61	72.97	76.68	75.35
object adjusted weighting combination	92.52	84.71	81.65	82.65	81.37
training on text and sequence, classification on sequence only	89.32	80.66	76.44	69.56	73.41

Table 2: Classification Accuracy (in %) for text descriptions, sequence data and varying combination methods.

cation process within our prototype is able to integrate all possible representations of an instance and can also handle the frequently occurring case when one or more representations are missing. Second, our prototype handles multi-classified objects, the occurrence of multiple inheritance and leaf nodes on different abstraction levels.

A thorough experimental evaluation of our prototype based on a versatile test bed for classifying proteins from SWISS-PROT into Gene Ontology is presented. Based on this test bed, we demonstrated that our method is capable to classify new entries with high accuracy and an efficiency adequate for real world applications.

For future work, we plan to extend our system to new data sources and ontologies, employing various different representations. Since these new data sources are likely to provide already linked instances, we plan to extend the system to exploit the general relationships supported by ontologies. Furthermore, we plan to develop a method for interleaved calculation of a general voting vector. For example, after the voting vector of the first data source is calculated, only those binary SVMs of other data sources might be used that still have a chance to change the result. Thus, the efficiency could be further improved.

References

[1] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. Bhat, H. Weissig, I. Shindyalov, and B. P.E. "The Protein Data Bank". *Nucleic Acid Research*, 28:235–242, 2000.

[2] B. Boeckmann, A. Bairoch, R. Apweiler, M.-C. Blatter, A. Estreicher, E. Gasteiger, M. Martin, K. Michoud, C. O'Donovan, I. Phan, S. Pilbout, and M. Schneider. "The SWISS-PROT Protein Knowledgebase and its Supplement TrEMBL in 2003". *Nucleic Acid Research*, 31:365–370, 2003.

[3] F. Bry and P. Kröger. "A Computational Biology Database Digest: Data, Data Analysis, and Data Management". *Distributed and Parallel Databases*, 13:7–42, 2003.

[4] T. G. O. Consortium. "Gene Ontology: Tool for the Unification of Biology". *Nature Genetics*, 25:25–29, 2000.

[5] C. Cortes and V. Vapnik. "Support-Vector Networks". *Machine Learning*, 20(3):273–297, 1995.

[6] M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. "Learning to Construct Knowledge Bases from the World Wide Web". *Artificial Intelligence*, 118(1/2):69–113, 1999.

[7] S. D'Allesion, K. Murray, and R. Schiaffino. "The Effect of Hierarchical Classifiers in Text Categorization". In *Proc. 3rd Conf. on Empirical Methods in Natural Language Processing (EMNLP'98)*, 1998.

[8] M. Deshpande and G. Karypis. "Evaluation of Techniques for Classifying Biological Sequences". In *Proc. of Pacific-Asia Conf. on Knowledge Discovery and Data Mining (PAKDD'02)*, pages 417–431, 2002.

[9] R. Duin. "The Combining Classifier: To Train Or Not To Train?". In *Proc. 16th Int. Conf. on Pattern Recognition (ICPR'02), Quebec City, Canada)*, pages 765–770, 2002.

[10] S. Dumais and H. Chen. "Hierarchical Classification of Web Content". In *Proc. 23rd Int. Conf. on Research and Development in Information Retrieval (SIGIR'00)*, pages 256–263, 2000.

[11] R. Durbin, S. Eddy, A. Krogh, and G. Mitchinson. *Biological Sequence Analysis*. Cambridge University Press, 1998.

[12] E.-H. Han and G. Karypis. "Centroid-Based Document Classification: Analysis and Experimental Results". In *Proc. 4th European Conf. on Principles of Data Mining and Knowledge Discovery (PKDD'00),*

Lyon, France, volume 1910 of *Lecture Notes in Computer Science*, pages 424–431. Springer, 2000.

[13] T. Joachims. "Text Categorization with Suport Vector Machines: Learning with Many Relevant Features". In *Proc. 10th European Conference on Machine Learning (ECML'98), Chemnitz, Germany*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer, 1998.

[14] J. Kittler, M. Hatef, R. Duin, and J. Matas. "On Combining Classifiers". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[15] D. Koller and M. Sahami. Hierarchically classifying documents using very few words. In *Proc. 14th Int. Conf. on Machine Learning (ICML'97), Nashville, TN*, pages 170–178, 1997.

[16] D. Kudenko and H. Hirsh. "Feature Generation for Sequence Categorization". In *Proc. of the 15th Nat. Conf. on Artificial Intelligence (AAAI'98)*, pages 733–738, 1998.

[17] L. Kuncheva, J. Bezdek, and R. Duin. "Decision Templates for Multiple Classifier Fusion: an Experimental Comparison". *Pattern Recognition*, 34:299–314, 2001.

[18] L. Larkey. Some issues in the automatic classification of u.s. patents. In *Learning for Text Categorization. Papers from the 1998 Workshop*, pages 87–90. AAAI Press, 1998.

[19] N. Lesh, M. Zaki, and M. Ogihara. "Mining Features for Sequence Classification". In *Proc. of the 5th Int. Conf. on Knowledge Discovery and Data Mining (SIGKDD'99)*, 1999.

[20] A. McCallum, R. Rosenfeld, T. Mitchell, and A. Ng. "Improving Text Classification by Shrinkage in a Hierarchy of Classes ". In *Proc. 15th Int. Conf. on Machine Learning (ICML'98), Madison, WI*, pages 359–367, 1998.

[21] D. Mount. *Bioinformatics: Sequence and Genome Analysis*. CSHL Press, 2001.

[22] J. Platt, N. Cristianini, and J. Shawe-Taylor. "Large Margin DAGs for Multiclass Classification". In *Advances in Neural Information Processing Systems 12 (NIPS Conference, Denver, CO, 1999)*, pages 547–553. MIT Press, 2000.

[23] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice-Hall Inc., 1971.

[24] G. Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[25] S. Vaithyanathan, J. Mao, and B. Dom. "Hierarchical Bayes for Text Classification". In *Proc. of Int. Workshop on Text and Web Mining (PRICAI'00), Melbourne, Australia*, pages 36–43, 2000.

[26] J. Wang, Q. Ma, D. Shasha, and C. Wu. "New techniques for extracting features from protein sequences". *IBM Systems Journal*, 40(2), 2001.

[27] K. Wang, S. Zhou, and S. Liew. "Building Hierarchical Classifiers Using Class Proximity". In *Proc. 25th Int. Conf. on Very Large Databases (VLDB'99), Edinburgh, Scotland*, pages 363–374, 1999.

[28] Y. Yang. An evaluation of statistical approaches to text categorization. Technical Report CMU-CS-97-127, Carnegie Mellon University, April 1997.