Hierarchical Text Categorization and Its Application to Bioinformatics

by

Svetlana Kiritchenko

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

In a hierarchical categorization problem, categories are partially ordered to form a hierarchy. In this dissertation, we explore two main aspects of hierarchical categorization: learning algorithms and performance evaluation. We introduce the notion of consistent hierarchical classification that makes classification results more comprehensible and easily interpretable for end-users. Among the previously introduced hierarchical learning algorithms, only a local top-down approach produces consistent classification. The present work extends this algorithm to the general case of DAG class hierarchies and possible internal class assignments. In addition, a new global hierarchical approach aimed at performing consistent classification is proposed. This is a general framework of converting a conventional "flat" learning algorithm into a hierarchical one. An extensive set of experiments on real and synthetic data indicate that the proposed approach significantly outperforms the corresponding "flat" as well as the local top-down method. For evaluation purposes, we use a novel hierarchical evaluation measure that is superior to the existing hierarchical and non-hierarchical evaluation techniques according to a number of formal criteria.

Also, this dissertation presents the first endeavor of applying the hierarchical text categorization techniques to the tasks of bioinformatics. Three bioinformatics problems are addressed. The objective of the first task, indexing biomedical articles with Medical Subject Headings (MeSH), is to associate documents with biomedical concepts from the specialized vocabulary of MeSH. In the second application, we tackle a challenging problem of gene functional annotation from biomedical literature. Our experiments demonstrate a considerable advantage of hierarchical text categorization techniques over the "flat" method on these two tasks. In the third application, our goal is to enrich the analysis of plain experimental data with biological knowledge. In particular, we incorporate the functional information on genes directly into the clustering process of microarray data with the outcome of an improved biological relevance and value of clustering results.

Acknowledgements

First, I wish to express my sincere gratitude to my supervisors, Pr. Stan Matwin and Pr. Fazel Famili. I am indebted to Stan who has taught me what it means to be a researcher and has provided support, encouragement, and valuable criticism through all these years. I thank Fazel for introducing me to the exciting science of bioinformatics, for generously sharing his knowledge and expertise in this area, and for giving professional advice.

I would also like to thank the members of my committee, Pr. Nathalie Japkowicz, Pr. John Oommen, and Pr. Marcel Turcotte, for their thoughtful comments and insightful discussions that help me considerably improve this dissertation.

My very special thanks go to Pr. Richard Nock from the Université Antilles-Guyane who contributed several ideas on hierarchical learning. Working with Richard was a very stimulating and fun experience. In fact, this one-month collaboration became a turning point in my research.

Also, I had a privilege to work with many wonderful people from the National Research Council of Canada (NRC). In particular, I would like to thank the Integrated Reasoning group at the Institute for Information Technology and especially the BioMiner team for their professional and friendly support and help with my research. I have also greatly benefited from the motivating discussions with outstanding biologists at the Institute for Biological Sciences (IBS) and the Biotechnology Research Institute (BRI), especially Dr. Roy Walker, Brandon Smith, Dr. Maureen O'Connor, Dr. Anne Lenferink, and Dr. Edwin Wang.

Grateful acknowledgments are made for AmikaNow! Corporation, its president Dr. Suhayya Abu-Hakima and the whole team of great professionals for their interest in my work, support and cooperation.

My fellow students from the University of Ottawa, Fernanda Caropreso, Marina Sokolova, Magda Widlak, Vivi Nastase, Anna Kazantseva, Jelber Sayyad, Quintin Armour, helped me in many different ways providing much needed support, advice, and friendship.

Thanks are also due to the researchers from all over the world, who generously made their software available for the present research: Erin Allwein, Robert Schapire, and Yoram Singer (BoosTexter), Ross Quinlan (C4.5), Amanda Clare (multi-label and hierarchical extensions to C4.5), Pedro Domingos (MetaCost).

Last, but not least, I am very grateful to my family, my parents who always believed in me and encouraged through all the way, and my husband, Misha, who made everything possible to help me make it this far.

The financial support during the course of my graduate work was provided by Natural Sciences and Engineering Research Council of Canada (NSERC), Communications and Information Technology Ontario (CITO), the Government of Ontario, the University of Ottawa, AmikaNow! Corporation, and National Research Council of Canada (NRC).

Contents

List of Tables

List of Figures

Notation

Below is the notation used in this dissertation.

D:	A domain of textual documents.
d_i:	A textual document from a given domain of documents D.
C:	A set of predefined categories (classes).
$\|C\|$:	The cardinality of a set of predefined categories C.
c_i:	A category (class) from a given set of predefined categories C.
C_i:	A subset of a given set of predefined categories C.
$\hat{C}_i$:	A subset C_i extended with ancestor categories: $\hat{C}_i = \{\bigcup_{c_k \in C_i} Ancestors(c_k)\}$
$\mathcal{H}$:	A class hierarchy.
$Ancestors(p)$:	An ancestor set of category p.
$Offspring(p)$:	An offspring set of category p.
$distance(c_i, c_j)$:	The distance between categories $c_i \in C$ and $c_j \in C$ in a class hierarchy $\mathcal{H} = \langle C, \leq \rangle$.
P:	Precision.
R:	Recall.
F:	F-measure.
hP:	Hierarchical precision.
hR:	Hierarchical recall.
hF:	Hierarchical F-measure.

Chapter 1

Introduction

The present dissertation addresses the task of hierarchical text categorization, text categorization where categories are hierarchically organized. Specifically, we focus on two aspects of hierarchical text categorization: learning and performance evaluation. We also present several applications of the hierarchical text categorization techniques to the area of bioinformatics.

The first chapter starts with the discussion of general text categorization, then formally introduces the task of hierarchical text categorization, presents the motivation for this work, and summarizes our research contribution.

1.1 Text categorization

Text categorization is a process of labeling natural language texts with one or several categories from a predefined set[1][2]. Usually, the predefined categories are thematic, but there are applications where categories are formed by other criteria, *e.g.* genre classification, email classification by priority, etc. Text categorization is a case of supervised learning where the set of categories and examples of documents belonging to those categories are given. This research will not concern problems of unsupervised learning, called text clustering, where the categories are not known in advance.

Text categorization as a research area appeared in the 1960s [Maron, 1961], yet only 15 years ago it became a major field in information science due to the increased interest

[1] Text categorization is also known as text classification or topic spotting. In the present dissertation we use terms *text categorization* and *text classification* interchangeably. The same applies to terms *category* and *class*.

[2] A formal definition of text categorization will be given in Section 1.1.1

in its diverse applications such as document indexing with controlled vocabulary, filtering of irrelevant information, web page categorization, email management, detection of text genre, and many others.

Clearly, text categorization techniques are a necessity nowadays when most information is produced and stored digitally. Business and personal correspondence, scientific and entertaining articles, conference proceedings, patient data are just a few examples of electronic text collections. With the advent of World Wide Web (WWW) another massive repository of text information was created. These huge text data demand automatic means of efficient and effective storage and retrieval that can be provided by means of text categorization.

Example 1. In the biomedical domain there exists a rich public text repository, Medline. It is an online library of all article abstracts published in all major biomedical journals for the last 40 years. This resource contains vital information for life scientists, but its free-text form makes it hard to work with. Automatic methods of categorizing the texts into specified biological topics would help users browse the collection easily and find the relevant information more quickly. Alternatively, categorizing articles according to a user's interests would also be greatly beneficial to biologists.

1.1.1 Formal definitions

Definition (Text Categorization). *Text categorization is the task of assigning a Boolean value to each pair* $\langle d_j, c_i \rangle \in D \times C$*, where* D *is a domain of documents and* $C = \left\{c_1, \ldots, c_{|C|}\right\}$ *is a set of predefined categories.* [Sebastiani, 2002]

Often, text categorization tasks are distinguished by the cardinality of the category set $|C|$.

Definition (Binary/Multi-class Text Categorization). *The text categorization task is called binary if* $|C| = 2$*; it is called multi-class if* $|C| > 2$*.*

Binary text categorization occurs less frequently in practice; nevertheless, it is an important case since any multi-class problem can be converted to $|C|$ binary problems where each problem concerns of whether a document belongs to a category or does not.

Text categorization tasks can also be distinguished by the number of categories a single document can be assigned to.

Definition (Single-label/Multi-label Text Categorization). *A text categorization task is called single-label if each document must be assigned to exactly one category. A text categorization task is called multi-label if each document can be assigned to any number of categories from* 0 *to* $|C|$.

1.2 Hierarchical text categorization

Hierarchical text categorization deals with problems where categories (classes) are organized in the form of a hierarchy[3]. This research area has received less attention, yet it has many important applications. Many text collections are organized as hierarchies: web repositories, digital libraries, patent libraries, email folders, product catalogs, *etc.* In addition, bio-medicine has several important taxonomies: Gene Ontology (GO), Medical Subject Headings (MeSH), Unified Medical Language System (UMLS), *etc.* Learning in the presence of class hierarchies is becoming a necessity in many text categorization applications.

Example 2. Traditionally, many biomedical topics are organized hierarchically: for example, diseases and other health problems (ICD-10) [ICD-10, 1992], gene functions (GO) [Ashburner et al., 2000], or Enzyme Commission Codes (ECC) [ECC, 1992]. As a result, categorization of Medline articles (see the previous example) is often accompanied by a hierarchy of classes and, thus, represents one of the many real-world applications for hierarchical text categorization techniques.

Until the mid-1990s researchers mostly ignored the hierarchical structure of categories present in some domains. In 1997 Koller and Sahami carried out the first proper study of a hierarchical text categorization problem [Koller and Sahami, 1997]. They proposed a divide-and-conquer principle, the most intuitive for hierarchical text categorization. As humans go through a hierarchy level by level, an automatic system also first classifies a document into high-level categories and then proceeds iteratively dealing only with the children of the categories selected at the previous level. This study experimentally showed that hierarchical information can, in fact, be extremely beneficial for text categorization improving the classification performance over the "flat" technique. After this work a number of approaches to hierarchical text categorization have been proposed [Chakrabarti et al., 1997, McCallum et al., 1998, Mladenic and Grobelnik, 1998, Wang et al., 1999, Weigend et al., 1999, Dumais and Chen, 2000, Cheng et al.,

[3] A formal definition of hierarchical text categorization will be given in Section 1.2.1

2001, Sun and Lim, 2001, Blockeel et al., 2002, Ruiz and Srinivasan, 2002, Dekel et al., 2004, Tsochantaridis et al., 2004, Cai and Hofmann, 2004]. In the past couple of years, it has become an active research area. Our work contributes to this research by exploring two of the most important aspects of hierarchical text categorization: learning and evaluation. In addition, we present three real-world applications of hierarchical categorization techniques from the field of bioinformatics.

1.2.1 Formal definitions

Definition (Poset). *A finite partially ordered set (poset) is a structure* $\mathcal{H} = \langle C, \leq \rangle$*, where* C *is a finite set and* $\leq \subseteq C \times C$ *is a reflexive, anti-symmetric, transitive binary relation on* C*.* [Joslyn, 2004]

Given a relation $\leq$, we define a relation $<$ as $q < p$ if $q \leq p$ and $q \neq p$. For any two categories $p, q \in C$ such that $q < p$ and $\nexists r \in C : q < r < p$, we will call p a parent category of q and q a child category of p. For any category $p \in C$, its ancestor set $Ancestors(p) = \{q \in C : q \geq p\}$, and its offspring set $Offspring(p) = \{q \in C : q \leq p\}$ (note that both sets include class p). We call categories that have no children leaf categories and classes that have both parents and children intermediate (or internal) classes.

The notion of posets is more general than the notion of trees in that categories in a poset can have multiple parents.

Definition (Hierarchical Text Categorization). *Hierarchical Text Categorization task is a text categorization task with a given poset structure* $\mathcal{H} = \langle C, \leq \rangle$ *on category set* C*.*

For any poset $\mathcal{H} = \langle C, \leq \rangle$ that represents a hierarchy we assume the existence of the root (or top) category $Root(\mathcal{H})$ which is an ancestor of all other classes in the hierarchy: $\{Root(\mathcal{H})\} = \bigcap_{p \in C} Ancestors(p)$. The root category itself has no parent classes.

Generally, text hierarchies are of a broader-narrower type where a subcategory represents a subtype or a part of the parent category (for more details see Section 2.3). Category hierarchies are usually represented in the form of a directed acyclic graph (DAG)[4].

[4]The formal definition of a DAG will be given in Section 2.2

1.3 Motivation

1.3.1 Hierarchical text categorization

> *The art of ranking things in genera and species is of no small importance and very much assists our judgment as well as our memory. You know how much it matters in botany, not to mention animals and other substances, or again moral and notional entities as some call them. Order largely depends on it, and many good authors write in such a way that their whole account could be divided and subdivided according to a procedure related to genera and species. This helps one not merely to retain things, but also to find them. And those who have laid out all sorts of notions under certain headings or categories have done something very useful.*
>
> *Gottfried Wilhelm Leibniz, New Essays on Human Understanding (1704).*

As Leibniz pointed out, a hierarchical organization of entities or notions is very helpful for humans to retain, find and analyze things. Therefore, it is not surprising that people organize large collections of web pages, articles or emails in hierarchies of topics or systematize a large body of biological knowledge in hierarchies of concepts (aka ontologies). Such organization allows to focus on a specific level of details ignoring specialization of lower levels and generalization of upper levels. Now, the task of automatic categorization systems is to deal with category hierarchies in an effective and efficient way.

Theoretically, hierarchical text categorization can be easily substituted with "flat" categorization if we ignore the class structure and replace a hierarchy with a set of categories. We can consider either only leaf categories if intermediate level categories are not a concern, or all categories in a hierarchy, in any case, paying no attention to any category relationships. However, by doing this we would disregard relevant information. For most text categorization tasks the category hierarchies have been carefully composed by humans and represent our knowledge on the subject matter. This additional information can boost the performance of a classification system if we find the way to incorporate it in the learning process. Although this has not been proved formally, several previous studies have demonstrated the performance advantages of hierarchical approaches over the "flat" technique experimentally [McCallum et al., 1998, Dumais and Chen, 2000, Dekel

et al., 2004, Tsochantaridis et al., 2004]. In addition, hierarchical categorization is flexible, giving us a choice of the level of detail we want to deal with. We can limit the classification process to upper levels of a hierarchy getting accurate information on documents' general topics. On the other hand, we may want to acquire as much detail as possible classifying documents into lower level categories.

Although a number of studies on hierarchical text categorization have been proposed recently, we feel that there is still room for improvement. In particular, we believe that hierarchical classification requires a special treatment to reflect the semantics of category relations. Most of the category relations present in class hierarchies are transitive (*e.g.* "is-a", "part-of"). Thus, a logical way to pursue hierarchical classification is to assign not just a single category in the middle of a hierarchy graph, but a complete subset including the most specific category and all its ancestor nodes. This would lead to a more comprehensible view on the classification results. Conversely, the conventional way of labeling can result in an ambiguous outcome, especially in multi-label classification when some of the ancestor categories are assigned to an instance, but not the others. In this work, we formalize the concept of hierarchically consistent classification that realizes this idea (Section 2.3). To the best of our knowledge, no previous study addresses a hierarchical categorization task in a hierarchically consistent manner. Hence, we present two learning algorithms whose main goal is to produce hierarchically consistent labeling (Chapter 4).

The second issue of hierarchical categorization that requires special attention is performance evaluation. Most of the studies on hierarchical text categorization focus primarily on the learning part leaving evaluation to the conventional "flat" techniques. However, we believe that the "flat" evaluation measures are inappropriate for the hierarchical setting because they do not take into account the hierarchical relations among categories and, as a result, have little discriminating power. A few hierarchical measures that have been introduced previously are not fully satisfiable too (for details see Section 5.1). Therefore, we propose a new hierarchical evaluation measure that possesses all the intuitively desired properties and is simple while extremely discriminating (Chapter 5).

1.3.2 Hierarchical text categorization in bioinformatics

Hierarchical text categorization has several interesting applications in bioinformatics[5]. Most new discoveries in life sciences first appear in scientific journals. Journal articles

[5]Bioinformatics is a field that deals with the computational aspect of biology.

are free-form texts, which makes searching through them a non-trivial task. There have been some efforts on structuring biological knowledge and organizing it in specialized databases. For example, the Saccharomyces Genome Database (SGD) is designed to contain all information on molecular biology of yeast (*i.e.* gene names, functions, DNA sequences, *etc.*). To keep a database up-to-date, the facts of interest have to be extracted from current publications and entered in the database. This is mostly done manually, and, therefore, requires substantial resources. As a result, such databases are often incomplete, leaving journal articles sometimes be the only source of requisite information. Given a large volume of available literature, automatic text mining tools have become crucial for life scientists.

The advantages of text categorization have been already recognized for many problems in bioinformatics (see Section 3.5). The main difficulty of working with biomedical texts is the variety of terminology used. To organize the domain knowledge and to effectively communicate this knowledge biologists have been designing controlled vocabularies (or ontologies) that associate biological concepts with carefully chosen terms. The consistent use of terminology is also beneficial for automatic text analysis tools. However, many authors still prefer to use their own terms and expressions, not to mention the large volumes of existing literature using legacy terminology. Automatic methods of categorizing texts into standardized terminology can address the problem of diverse vocabulary. As controlled vocabularies are mostly designed as hierarchies, this task becomes a hierarchical text categorization task. In spite of its importance, this research area has received little attention. In fact, there has been no single study known to us that addresses the task from the hierarchical point of view. We would like to fill this gap and bring the benefits of hierarchical text categorization to bioinformatics.

Specifically, we address the task of indexing of biomedical articles with Medical Subject Headings (MeSH), which is a standard biomedical vocabulary used to index a large part of the Medline library. MeSH indexing is an essential part of Medline requiring a tremendous manual effort; hence, automatic techniques for this task are extremely valuable. Our second application tackles a more complex problem of annotating genes with functional categories of the Gene Ontology from the biomedical literature. Functional annotation of genes is one of the fundamental tasks of genomics, a subfield of biology. We address this problem as a hierarchical text categorization task and show that including hierarchical information on class relations benefits the classification process to a large extent. Finally, in our third application we incorporate the functional information on genes into the clustering process, which is one of the main steps in mi-

croarray data analysis. Our experiments demonstrate that this background knowledge significantly improves the cluster quality producing functionally coherent clusters that are more meaningful and practical for biologists. Overall, these three applications can be viewed as first steps towards creating and maintaining integrated biomedical knowledge bases.

1.4 Contributions

This work presents a comprehensive study of the learning and evaluation techniques for hierarchical text categorization. It identifies the weaknesses of the existing approaches and proposes new methods to overcome these limitations. In particular, the formal concept of hierarchically consistent classification is defined to fully reproduce the semantics of hierarchical category relations. Then, two hierarchical learning algorithms, an extended version of the local method pachinko machine and a novel global approach, that aim at consistent classification are proposed. Both algorithms represent general frameworks of applying a conventional learning method in the hierarchical settings. Instead of taking the approach of adapting a specific "flat" learning algorithm to hierarchical categorization, we design general hierarchical procedures suitable for any conventional multi-label learning techniques. An extensive set of experiments on real and synthetic data demonstrates a significant advantage in performance of these techniques over the conventional "flat" method. Moreover, the extent of the achieved improvement amplifies with the size of a class hierarchy.

Another important contribution of this work is a novel hierarchical evaluation measure that possesses a number of imperative qualities and is superior to the existing "flat" as well as hierarchical evaluation techniques. Specifically, the new measure is simple, requires no parameter tuning, gives credit to partially correct classification and discriminates errors by both distance and depth in a class hierarchy. It is also statistically consistent, yet more discriminating than standard "flat" measures according to the definitions proposed by Huang and Ling [Huang and Ling, 2005], which is the only way we know to systematically compare classifier performance measures.

This work also presents the first endeavor of applying the hierarchical text categorization techniques to tasks in bioinformatics. Three bioinformatics problems are addressed, namely biomedical article indexing with Medical Subject Headings, functional annotation of genes from biomedical literature, and gene expression analysis in the presence of background knowledge. These three tasks are among the high-priority daily activities

in the field. Thus, automatic means that can efficiently deal with these problems are in high demand as they can considerably reduce the manual effort.

In addition, these bioinformatics applications pose a real-life challenge for the hierarchical techniques. Previous studies on hierarchical text categorization used for evaluation fairly small (mostly 2-level, rarely 3-4-level) taxonomies. We, on the other hand, explore biomedical hierarchies that have up to 12 levels and over a thousand categories. The experiments of such a large scale bring invaluable experience to the research.

Some parts of the present research have appeared in the following publications:

- Svetlana Kiritchenko, Stan Matwin, Richard Nock, and A. Fazel Famili. Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization. Submitted, 2005.

- Svetlana Kiritchenko, Stan Matwin, and A. Fazel Famili. Functional Annotation of Genes Using Hierarchical Text Categorization. *Proc. of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology*, 2005.

- Svetlana Kiritchenko, Stan Matwin, and A. Fazel Famili. Hierarchical Text Categorization as a Tool of Associating Genes with Gene Ontology Codes. *Proc. of the Second European Workshop on Data Mining and Text Mining for Bioinformatics*, pp. 26-30, 2004.

Finally, we would like to note that although we mainly focus on text categorization, the proposed hierarchical techniques may be applied to categorization of entities of any kind, *e.g.* images, audio, video, etc.

1.5 Thesis outline

The remainder of the thesis is organized as follows. In Chapter 2, we discuss some issues specific to hierarchical text categorization: hierarchical structures (DAGs and trees), category relationships ("is-a" and "part-of"), multi-class categorization, and multi-label categorization. We also formally introduce the notion of hierarchically consistent classification. Chapter 3 describes previous work on feature selection, learning algorithms, evaluation, and applications of hierarchical text categorization. In addition, an overview of previous work on text categorization in bioinformatics is given. Then, our novel hierarchical techniques are presented in detail. First, in Chapter 4 we introduce two new hierarchical learning approaches, local and global, that conform with the hierarchical

consistency requirement. Then, in Chapter 5 we present a novel hierarchical evaluation measure. We begin with a discussion of the weaknesses of the existing hierarchical and non-hierarchical evaluation measures, after which a new measure is proposed and its properties are examined. The experimental comparison demonstrating the superiority of the hierarchical learning algorithms over the "flat" approach on a variety of real and synthetic textual data is presented in Chapter 6. Chapter 7 describes three bioinformatics applications where hierarchical techniques appear to be very practical. Finally, Chapter 8 summarizes the presented research and discusses possible directions for future work.

Chapter 2

Specifics of hierarchical text categorization

This chapter defines some notions specific for hierarchical text categorization. In particular, we discuss the types of hierarchies (*e.g.* ontologies, taxonomies, thesauri), the structural aspect of hierarchical graphs (DAGs vs. trees), the types of category relations ("is-a" and "part-of"), multi-class and multi-label aspects of hierarchical text categorization. Furthermore, this chapter introduces the novel concept of hierarchically consistent classification, which is the central concept of the present dissertation.

2.1 Hierarchies: ontologies, taxonomies, thesauri

A hierarchy is a set of entities, usually terms or categories, with partially defined relationships between those entities (Section 1.2.1). Hierarchies have many areas of application. In information science, hierarchies are often used to represent controlled vocabularies in the form of taxonomies and thesauri. According to the definition by the National Library of Canada, controlled vocabularies establish standardized terminology for use in indexing and information retrieval. Taxonomy is a form of hierarchy representing a controlled vocabulary, while thesaurus is a more advanced controlled vocabulary that gives not only the relationships between the terms in the form of a hierarchy (narrower-broader terms), but also related and preferable terms. In addition, taxonomies are also used to represent a category structure as, for example, in web directories.

Another concept related to hierarchies is ontology. In ancient philosophy, ontology referred to the study of the nature and relation of being. Today, however, ontology is

defined as "a specification of a specialization" [Gruber, 1993]. It defines an information structure for communicating knowledge. Comparing to taxonomy, ontology does not only organize entities in a hierarchy, but also provides exact semantics for these entities. Moreover, ontologies often use richer semantic relationships among terms. The concept of ontology is especially widespread in life sciences (see for example, EcoCyc [Karp et al., 1999], GenProtEC [Riley, 1998], MYGD [Mewes et al., 1997], KEGG [Ogata et al., 1999], GO [Ashburner et al., 2000]).

In this research we deal with category hierarchies represented in the form of taxonomy (newsgroups and news articles) and ontology (Gene Ontology).

2.2 Structure

A hierarchy is a finite partially ordered category set (poset) (Section 1.2.1). For convenience, it is usually given in the form of a directed acyclic graph (DAG) (see definitions below). Each DAG determines a unique poset.

Definition (Graph). *A graph G is a pair (V, E), where V is a set of vertices (aka nodes), and E is a set of edges between the vertices $E = \{(u, v)|u, v \in V\}$.*

Definition (Directed Acyclic Graph). *A directed acyclic graph (DAG) is a graph where each edge has a direction and there are no cycles.*

A category in a hierarchy is represented as a vertex in the corresponding directed acyclic graph. The edges in the graph show the parent-child relations: if p is a parent of q, then there is an edge $(p, q) \in E$ leading from vertex p to vertex q. In general, we omit the direction of edges in a graph assuming it is top-down.

Definition (Path in a Directed Graph). *A path in a directed graph is a list of vertices of the graph where each vertex has an edge from it to the next vertex. The length of the path is the number of edges traversed.*

As stated in Section 1.2.1, we assume the existence of a root category in a hierarchy, which is the common ancestor of all classes in a hierarchy. It is usually represented as the top node in a graph. For each vertex in a graph, there exists a path from the root to the vertex.

Definition (Depth of a Vertex in a DAG). *The depth (or level) of a vertex in a DAG is the length of the shortest path from the root to the vertex.*

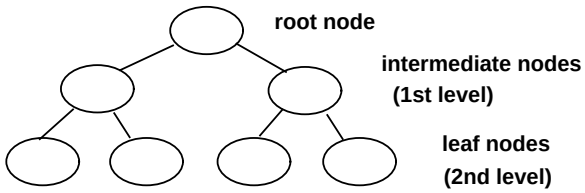


Figure 2.1: Example of a two-level hierarchy showing the root (top) node, intermediate nodes, and leaf nodes.

Definition (Depth of a Directed Acyclic Graph). *The depth of a directed acyclic graph is the maximal depth over all vertices in the graph.*

Sometimes, we call a DAG hierarchy of depth n an n-level hierarchy. For example, a two-level hierarchy has depth 2, in other words, having a root node, intermediate nodes and leaf nodes (Figure 2.1).

Definition (Average Out-degree of a Directed Graph). *The out-degree of a vertex v is the number of edges initiated in vertex v. The average out-degree of a directed graph is the average of the out-degree of all its vertices.*

There is a special case of DAG, called **tree**, where each vertex is allowed to have exactly one parent (except a root node that has no parent). Since a tree is a simpler kind of a hierarchy, most of the previous research on hierarchical text categorization dealt with trees only. We, on the other hand, consider methods that work with general DAGs and experiment with real-life DAG hierarchies, such as the Gene Ontology.

2.3 Category relationships

There exist several kinds of relationships between entities: "is-a", "part-of", "made-of", "cause-to", "similar-to", *etc.* However, in category hierarchies only two kinds can be usually found: "is-a" and "part-of". "Is-a" is a specification relationship. For example, cell adhesion is a kind of cell communication, so that the category "cell adhesion" is a (more specific) subcategory of the category "cell communication". The "is-a" relation is asymmetric (*e.g.* cell adhesion is a kind of cell communication, yet cell communication is not only cell adhesion) and transitive (cell adhesion is a kind of cell communication, cell communication is a cellular process; therefore, cell adhesion is a cellular process) [Ruiz and Srinivasan, 2002]. In an "is-a" hierarchy, if an object belongs to a category it also belongs to all its ancestor categories.

"Part-of" is a partitive relationship. For example, cell aging is a part of the process called cell death; thus, the category "cell aging" is a subcategory (a part) of the category "cell death". Similarly, the "part-of" relation is asymmetric and transitive. We will also consider that in a "part-of" hierarchy, if an object belongs to a category it also belongs to all its ancestor categories.

Most of the text hierarchies are "is-a". However, some biological hierarchies, for example Gene Ontology, contain "is-a" as well as "part-of" relationships.

As stated above, for most class hierarchies we can safely assume that an instance belonging to a category also belongs to all ancestor nodes of that category. Therefore, it is meaningless to assign an instance to a category and not assign it to the parent category, as conventionally done in hierarchical classification systems. For example, saying that a gene is responsible for "cell adhesion", but not for "cell communication" would be inappropriate. We want to express this semantics explicitly by requiring a classifier to assign all the relevant labels, including the ancestor labels, to a given instance. In this way, the assigned labels would clearly indicate the position of an instance in a category hierarchy. Thus, we introduce the notion of hierarchical consistency and expect any hierarchical classification algorithm to conform to the hierarchical consistency requirement defined as follows [Kiritchenko et al., 2005b].

Definition (Hierarchical Consistency). *A label set $C_i \subseteq C$ assigned to an instance $d_i \in D$ is called consistent with a given hierarchy if C_i includes complete ancestor sets for every label $c_k \in C_i$, i.e. if $c_k \in C_i$ and $c_j \in Ancestors(c_k)$, then $c_j \in C_i$*[1].

Definition (Hierarchical Consistency Requirement). *Any label assignments produced by a hierarchical classification system on a given hierarchical text categorization task has to be consistent with a corresponding class hierarchy.*

Fiqure 2.2 presents examples of consistent and inconsistent label assignments. The figure on the left gives an example of the consistent assignment since the set includes labels along with all their ancestors. The figure on the right shows an inconsistent assignment: label A is included while its parent label B is not. Such situations can occur if we apply a conventional text categorization method to a hierarchical task without any modifications.

[1]We assume that every instance belongs to the top class of the hierarchy; therefore, we always exclude the top node from any ancestor set since including it does not provide any additional information on the instance.

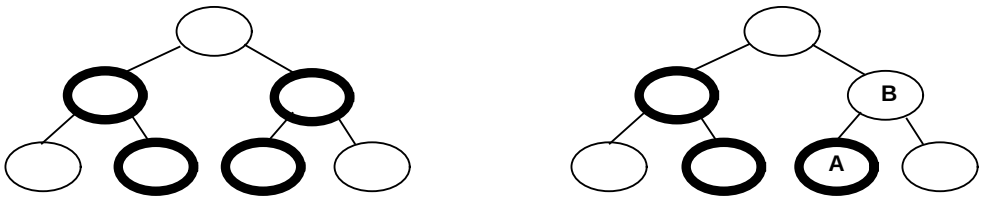


Figure 2.2: Examples of consistent and inconsistent label assignments. The ellipses in bold represent the categories assigned to an instance by a classifier. The left figure represents a consistent assignment since the labels are included in the set along with all their ancestors. The right figure shows an inconsistent assignment since node A is included in the set while its parent, node B is not.

2.4 Multi-class categorization

Unlike classical text categorization where binary applications are common (for example, spam filtering), hierarchical text categorization is always multi-class. Even when we divide a given task into smaller subtasks, rarely do we have only two classes. For example, the first level of the biological process hierarchy in Gene Ontology has 7 categories. They in turn contain from 6 to 41 children categories with the exception of one category, "biological process unknown", containing no children. Overall, the hierarchy consists of several thousands of categories.

Most of the state-of-the-art learning algorithms, such as Naive Bayes, decision trees, neural networks, are capable of dealing with multiple classes. However, there are algorithms, for example Support Vector Machines, that were designed to work only with binary problems. In such cases we can easily transform a given n-class problem into n binary problems: the i^{th} binary problem corresponds to a decision if an instance belongs to the i^{th} category or not. Yet, the computational complexity of the transformed problem would be higher since we now have to solve n problems instead of one.

Another way of dealing with multiple categories is pairwise comparison. We transform a given n-class problem into $\frac{n\cdot(n-1)}{2}$ binary problems each time comparing only a pair of categories. The final decision for each category is taken by a vote.

A more interesting and less computationally expensive solution for a multi-class problem is Error-Correcting Codes (ECOC) [Dietterich and Bakiri, 1995]. Each of the n categories is mapped into a unique k-bit string ($k < n$) called a codeword. Then, k binary classifiers are trained, one for each bit. When a new instance is classified, a category with the codeword closest to the bit string produced by the classifiers is selected.

2.5 Multi-label categorization

Most of the real-life applications of hierarchical text categorization are multi-label tasks, which means that an instance can belong to any number of categories from 0 to $|C|$. For example, a gene can be associated with several biological processes and, therefore, would be classified into several category nodes of Gene Ontology. Some of the learning algorithms can be readily applied to such problems. For example, with a Naive Bayes classifier we could assign to an instance not only the category with the highest predicted probability, but also all categories for which the probabilities are greater than a certain threshold. For other algorithms that can predict only one category at a time, a common solution is to transform an n-class problem into n binary problems (as described in the previous section).

The computational overhead of solving many binary problems instead of one multi-label problem forced several researchers to look for other solutions. One such solution is to adapt classical learning algorithms to predict not one category, but a set of categories simultaneously. As a result, there has been some work on modifying the decision tree learning algorithm by extending the expression for class entropy for multi-label categorization [Suzuki et al., 2001, Clare, 2003]. Wang and colleagues adapted an association rule based method to allow a set of categories be predicted by a single rule [Wang et al., 2001], while McCallum proposed a Bayesian approach where multiple labels for a document are represented by a mixture model [McCallum, 1999]. In this research, we employed AdaBoost.MH, a multi-class multi-label version of a very successful boosting algorithm AdaBoost [Schapire and Singer, 1999].

Chapter 3

Previous work

This chapter discusses the related work on three aspects of hierarchical text categorization: feature selection, learning algorithms, and performance evaluation. Then, it presents several applications of hierarchical text categorization to other domains such as email classification, personalization systems, question answering, etc. Finally, an overview of text learning related research in bioinformatics is given. By analyzing previous work on hierarchical text categorization and text learning applications in bioinformatics, we identify several issues that require further investigation and address those issues in the following chapters of the present dissertation.

3.1 Document representation and feature selection

The conventional method of representing texts is the *bag of words* approach where each word (or word stem[1]) from a vocabulary corresponds to a feature and a document corresponds to a feature vector. Sometimes, instead of single words, features can represent phrases or n-grams, sequences of n words with $n \geq 2$ [Mladenic and Grobelnik, 1998]. Intuitively, more complex features such as phrases should enrich the information space. For example, the phrase "world wide web" is much more informative for text categorization than those three words separately. However, in practice the use of n-grams leads to an overwhelming number of features and can actually deteriorate the performance [Caropreso et al., 2001].

After determining the features, we need to compute the feature values (or term

[1] A word stem is the morphological root of the word. The common name for both types of features (word and word stem) is term.

weights) for each document. It can be done in several ways. Binary features refer to the presence or absence of a term. Term frequencies present the number of times a term occurred in a document, sometimes normalized by the total number of term occurrences in a document. Another common method for computing term weights is TFIDF:

$$TFIDF = TF(t_k, d_j) \cdot log\frac{|D|}{DF(t_k)},$$

where $TF(t_k, d_j)$ defines the number of times term t_k occurs in document d_j, $DF(t_k)$ denotes the number of documents term t_k occurs in, and $|D|$ stands for the total number of training documents in the collection[2]. The intuition behind this formula is to give more weight to terms frequently occurring in a document but rarely in the whole collection, and, thus, having more discriminating power.

Generally, only a small portion of all possible features is used in a categorization system. The sizes of the vocabularies of modern text collections are estimated in hundreds of thousands. Processing of the feature vectors of such dimension requires extensive computational resources and can sometimes be infeasible for some learning algorithms. In addition, many of these features are uninformative or noisy, which can lead to the problem of overfitting. In machine learning, the traditional way of dealing with a vast number of features is feature selection.

Feature selection is a process of selecting the most informative features while discarding the noisy ones. There are two main types of feature selection methods: wrappers and filters. A wrapper approach chooses a feature subset that leads to the best classification performance for a given learning algorithm, while a filter approach chooses the most relevant features without consideration of the learning procedure. The latter is computationally easier and, therefore, has been more preferable in text categorization. A number of criteria for determining the feature relevancy for the text categorization task have been proposed in the literature. Some of them are presented in Table 3.1.

In hierarchical text categorization the need for feature selection is even more apparent since the hierarchical corpora are usually larger. Almost all studies on hierarchical text categorization are forced to select a feature subset in order to be able to run experiments on real-life data.

In hierarchical text categorization feature selection has been used in a global or local way.

[2]In the present dissertation we follow the general approach of defining the terms constituting a mathematical expression immediately after the expression.

Function	**Mathematical Form**
Document frequency	$P(t_k \mid c_i)$
Mutual information	$log \frac{P(t_k,c_i)}{P(t_k)\cdot P(c_i)}$
Information gain	$P(t_k)\sum_i P(c_i \mid t_k)\cdot log\frac{P(t_k,c_i)}{P(t_k)\cdot P(c_i)} + P(\bar{t_k})\sum_i P(c_i \mid \bar{t_k})\cdot log\frac{P(\bar{t_k},c_i)}{P(\bar{t_k})\cdot P(c_i)}$
Cross entropy	$P(t_k)\sum_i P(c_i \mid t_k)\cdot log\frac{P(t_k,c_i)}{P(t_k)\cdot P(c_i)}$
Weight of evidence	$\sum_i P(c_i)\cdot P(t_k)\cdot \left\lvert log\frac{P(c_i \mid t_k)(1-P(c_i))}{P(c_i)(1-P(c_i \mid t_k))}\right\rvert$
Odds ratio	$\frac{P(t_k \mid c_i)\cdot(1-P(t_k \mid \bar{c_i}))}{(1-P(t_k \mid c_i))\cdot P(t_k \mid \bar{c_i})}$
Chi-square	$\frac{\lvert D\rvert\cdot(P(t_k,c_i)\cdot P(\bar{t_k},\bar{c_i})-P(t_k,\bar{c_i})\cdot P(\bar{t_k},c_i))^2}{P(t_k)\cdot P(\bar{t_k})\cdot P(c_i)\cdot P(\bar{c_i})}$
Correlation coefficient	$\frac{(P(t_k,c_i)\cdot P(\bar{t_k},\bar{c_i})-P(\bar{t_k},c_i)\cdot P(t_k,\bar{c_i}))\cdot\sqrt{\lvert D\rvert}}{\sqrt{(P(t_k,c_i)+P(\bar{t_k},c_i))(P(t_k,\bar{c_i})+P(\bar{t_k},\bar{c_i}))(P(t_k,c_i)+P(t_k,\bar{c_i}))(P(\bar{t_k},c_i)+P(\bar{t_k},\bar{c_i}))}}$
Fisher index	$\frac{\sum_l \lvert c_l\rvert\cdot(\frac{1}{\lvert c_l\rvert}\sum_{d_j\in c_l} TF(t_k,d_j)-\frac{1}{\lvert c_i\rvert}\sum_{d_j\in c_i} TF(t_k,d_j))^2}{\sum_l(\frac{1}{\lvert c_l\rvert}\sum_{d_j\in c_l}(TF(t_k,d_j)-\frac{1}{\lvert c_l\rvert}\sum_{d_j\in c_l} TF(t_k,d_j))^2)}$

Table 3.1: Main functions for determining feature relevancy in the feature selection process. The functions specify the relevancy of term t_k to category $c_i \in C$ in the probabilistic form. The probabilities are estimated by counting the corresponding occurrences in the training data. For example, $P(t_k)$ denotes the probability that a random document d contains term t_k and is estimated as the proportion of documents containing t_k in the training set. $|D|$ denotes the total number of documents, $\bar{t_k}$ represents the absence of term t_k, and $\bar{c_i}$ represents all categories in C other than c_i. In the last formula, Fisher index, $TF(t_k, d_j)$ stands for the number of occurrences of term t_k in document d_j and $|c_i|$ states the number of documents in category c_i.

Definition (Global Feature Selection in Hierarchical Text Categorization). *In hierarchical text categorization feature selection method is called global if it selects features that distinguish among all categories in a given hierarchy.*

Definition (Local Feature Selection in Hierarchical Text Categorization). *In hierarchical text categorization feature selection method is called local if it selects relevant features for each subproblem separately, where a subproblem corresponds to an internal node of a class hierarchy.*

A global approach to feature selection is akin to traditional feature selection in "flat" text categorization (Figure 3.1a). A local approach, on the other hand, treats every

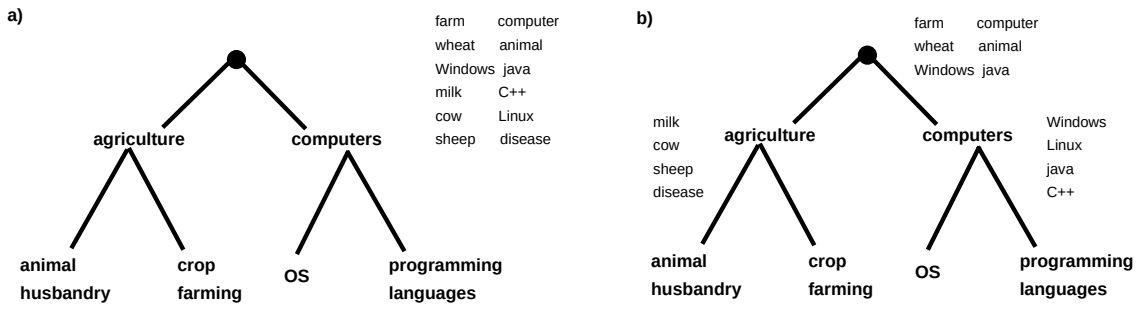


Figure 3.1: Global (a) and local (b) feature selection. In global feature selection one set of features is used to discriminate among all categories in a hierarchy. In local feature selection each internal node of a hierarchy has its own set of features.

internal category as a separate classification subtask and selects features for each subtask independently. Relevant features for a subtask are the ones that discriminate the children categories of the corresponding internal node (Figure 3.1b).

3.1.1 Hierarchical global feature selection

A global approach is normally employed in systems where learning is also done in a global manner, *i.e.* only one classifier is constructed to distinguish among all categories in a hierarchy [Frommholz, 2001, Itskevitch, 2001]. However, Wibowo and Williams use global feature selection while splitting the initial task into subtasks for learning a classifier [Wibowo and Williams, 2002a]. To manage an overwhelming number of potential global features, they propose a simple and fast method of selecting a fixed, small number of terms from the beginning of each document. McCallum et al., on the other hand, use the global approach to classification while selecting the features locally at first and then taking the union of the feature subsets [McCallum et al., 1998]. They choose the most relevant features based on mutual information (see Table 3.1).

3.1.2 Hierarchical local feature selection

A local approach to feature selection, unlike the global one, takes advantage of dividing a large initial problem into subproblems. The benefits of doing this are two-fold. First, the number of features needed to discriminate among a small number of categories (for a subtask) is typically much less than what would be needed for an initial problem containing hundreds of categories. Second, feature subsets for each subproblem are selected independently, so that a small number of features, the most relevant for a

particular subproblem, can be chosen. The subproblems can differ substantially and, therefore, should be categorized by different terms. Like in the example by Koller and Sahami, words "farm" and "computer" are good indicators for topics "agriculture" and "computers", but these words are unlikely to be helpful to distinguish between "animal husbandry" and "crop farming" because "agriculture" is likely to appear in documents of both kinds and "computers" most probably will not appear in any documents [Koller and Sahami, 1997]. As a result, the authors propose to use local feature selection, *i.e.* choosing relevant features at each node of a hierarchy. In their work they use expected cross-entropy (see Table 3.1) and show that their hierarchical classifier reaches the best performance with only a few words (in their case, 10) and its performance is superior to the performance of a "flat" classifier [Koller and Sahami, 1997].

Before Koller and Sahami's classical paper, there were a few works mentioning hierarchical categorization. In the study by Ng et al., the authors compared three methods: correlation, chi-square, and term frequency, selecting features at each node of the hierarchy [Ng et al., 1997]. Yet, hierarchical categorization was not the primary focus of the paper, and the authors did not compare the performance of the hierarchical approach with the performance of the corresponding "flat" approach. Wiener and colleagues proposed to create a hierarchy over a set of categories, combining similar topics into one high-level category [Wiener et al., 1995]. They employed this new hierarchy to perform local feature selection with Latent Semantic Indexing (LSI)[3] and top-down classification approach with neural networks. Their results showed the improved performance of the hierarchical model over the "flat" model. Nevertheless, Koller and Sahami's paper was the first work whose goal was to explore hierarchical text categorization, categorization with a given category hierarchy, and therefore, is often cited as the pioneering work in this area.

Following that work, a number of studies applied local feature selection for hierarchical text categorization using different feature relevancy criteria such as Fisher index [Chakrabarti et al., 1997, Cheng et al., 2001] or information gain [Dumais and Chen, 2000]. In the paper by Chuang et al. feature selection with the help of manually made positive and negative feature vectors is compared to TFIDF-based selection with the former significantly outperforming the latter [Chuang et al., 2000]. Ruiz and Srinivasan compare correlation, mutual information, and odds ratio, finding that these three meth-

[3]Latent Semantic Indexing is a feature selection technique that reduces the dimensionality of the feature vector space by combining co-occurring terms into joint features. The dimensionality reduction is achieved by applying a singular value decomposition to the matrix formed by the original document vectors.

ods perform the same if mutual information is enhanced with discarding low frequency terms [Ruiz and Srinivasan, 2002]. On the other hand, Mladenic and Grobelnik compare 6 different feature selection methods, namely information gain, cross entropy, mutual information, weight of evidence, odds ratio, and term frequency and find that odds ratio perform the best [Mladenic and Grobelnik, 1998]. Also, they confirm Koller and Sahami's conclusions that the best results are achieved for relatively small feature subsets.

Feature selection methods specific for hierarchical categorization have also been proposed. In the work by D'Alessio et al. features are chosen locally based on their frequency: a feature is considered relevant for a category if its occurrence in the category is much higher than in the parent category. In other words, since a parent category is formed as a union of its children, a feature is relevant if it occurs in the category more frequently than in the sibling categories altogether [D'Alessio et al., 2000].

The comparison between the local and global approaches to feature selection has been performed by Weigend and colleagues [Weigend et al., 1999]. Two methods, chi-square and latent semantic indexing (LSI) are used in combination with hierarchical neural networks. Surprisingly, the results are mixed. On Reuters-22173 dataset the local feature selection with LSI outperforms the corresponding global approach, whereas the local approach with chi-square performs better only on the subset of low frequency categories.

3.2 Learning algorithms

Until the mid-1990s machine learning researchers mostly ignored the hierarchical category structure present in some text categorization applications by turning a hierarchy into a flat set of categories. With the appearance of World Wide Web and other large collections of documents the need in hierarchical classification has increased. As a result, a number of studies on hierarchical text categorization have appeared in literature.

Hierarchical categorization methods can be divided in two types [Wang et al., 1999, Itskevitch, 2001, Sun and Lim, 2001]: global (or big-bang) and local (or top-down level-based).

Definition (Global Hierarchical Text Categorization). *In hierarchical text categorization, a learning approach is called global if it builds only one classifier to discriminate all categories in a hierarchy.*

A global approach differs from "flat" categorization in that it somehow takes into

account the relationships between the categories in a hierarchy.

Definition (Local Hierarchical Text Categorization). *In hierarchical text categorization, a learning approach is called local if it builds separate classifiers for internal nodes of a hierarchy.*

A local approach usually proceeds in a top-down fashion first picking the most relevant categories of the top level and then recursively making the choice among the low-level categories, children of the relevant top-level categories. A local approach seems natural for hierarchical classification since it reflects the way that humans usually perform such tasks. Discriminating among a few categories, children of one internal node, is much easier than discriminating among hundreds of categories. The same is also true for automatic systems. In machine learning it is well known experimentally that, in general, the more categories are present, the more difficult the task is, and therefore, the lower classification accuracy can be reached[4]. Also, intuition tells us that classifying into high-level categories is easier than discriminating among all categories not only because the number of categories is smaller but also because they are more distinctive. Then, after correct categorization into one of the high level categories is done, the number of possible low-level categories becomes smaller if we consider only the children of the correct high-level categories. This intuition has been confirmed experimentally by Wibowo and Williams [Wibowo and Williams, 1999]. In their study, the number of misclassification errors at the parent level is 11 times smaller than the number of misclassification errors at the children level.

However, each of the approaches has its weaknesses [Sun and Lim, 2001]. The global approach is computationally heavy. It cannot exploit different sets of features at different hierarchical levels. Finally, it is not flexible; a classifier has to be re-trained each time the hierarchical structure changes. The local approach, while computationally more efficient, has to make several correct decisions in a row to correctly classify one example, and errors made at top levels are usually not recoverable. Also, at low levels the categories become smaller, so the number of training examples can be insufficient to learn a reliable classifier. Below, we give a brief overview of both global and local learning algorithms that appear in the literature.

[4]For example, a random classifier has a 50% chance to correctly classify an instance in a binary problem and only $1/|C|$ for a multi-class problem ($|C| > 2$).

3.2.1 Global approaches to hierarchical text learning

Despite its computational overhead, the global approach has been employed in several text categorization systems. An association rule based method has been proposed by Wang and colleagues [Wang et al., 1999]. Given a hierarchy of categories and a hierarchy of terms, they produce a set of rules in the form $X \rightarrow C$, where X is a set of terms and C is a category. Based on algorithms for mining association rules, they first generate all rules $X \rightarrow C$ that satisfy the minimum support specified by a user. Then, they sort the rules according to some ranking criteria that take into account the distance in the hierarchy between the correct and assigned category. Finally, they select the minimal number of top rules that gives the minimum error. In the subsequent study they extend this approach to accommodate multi-label categorization [Wang et al., 2001]. In addition, they modify their ranking criteria. Their new metric takes into account the category set similarity and is based on the number of documents shared by two category sets (see Section 3.3). Itskevitch proposes a slightly different approach for creating association rule hierarchical classifier [Itskevitch, 2001]. She produces a tree-like structure of frequent patterns found in data and prune the rules based on specified criteria during and after the tree construction.

Some researchers adapt decision tree learning algorithms to accommodate for multi-label classification and to take into account a given class hierarchy [Blockeel et al., 2002, Clare, 2003]. In their work, Blockeel and colleagues employ predictive clustering trees [Blockeel et al., 2002]. Predictive clustering trees are decision trees that can predict multiple target attributes at once. They are obtained in a standard top-down fashion by recursively partitioning data into clusters such that the intra-cluster variance is minimized. Intra-cluster variance is defined as the sum of squared distances between the members of the cluster and its center (a point closest to all members of the cluster). In their work the distances between data points are calculated as the weighted shortest path distance between the corresponding categories in a hierarchical tree, *i.e.* the sum of the weights of the edges on the shortest path between the class nodes (see Figure 3.5a). The weights decrease exponentially with depth.

Clare adapts classical decision tree algorithm C4.5 [Clare, 2003]. She allows several most frequent classes to be associated with a leaf. In addition, she changes the entropy formula combining the class entropy with the classification depth as follows:

$$entropy = -\sum_i (P(c_i)logP(c_i) + (1 - P(c_i))log(1 - P(c_i)) - \alpha(c_i)log(tree\_size(c_i))) \,,$$

where $tree\_size(c_i)$ is the number of nodes in a subtree rooted in c_i and

$$\alpha(c_i) = \begin{cases} 0, \text{if } P(c_i) = 0, \\ \text{a user defined constant, otherwise.} \end{cases}$$

With the parameter α a user can trade off between the homogeneity of predictive nodes (which would reflect classification precision) and the specificity level of predictions in a hierarchical tree.

More recently, large margin classifiers have been extended to be applicable in the hierarchical settings [Dekel et al., 2004]. In this work, each node in a class hierarchy is associated with a prototype vector, and an instance is classified to the class with the most similar prototype. The general idea of prototype-style classifiers has been implemented in several well-known learning algorithms, *e.g.* in the Rocchio method [Hull, 1994]. Dekel and colleagues combine this prototype-style approach with the large margin principle and put it in the context of hierarchical categorization. The hierarchical settings impose the requirement of adjacent categories in a hierarchical tree to have similar prototypes. The learning algorithm works in an online manner updating the current prototype vectors at each round when a new training instance is given. The new set of prototypes is kept close to the current set while maintaining the large margins between the correct and each of the incorrect labels for a given instance. Formally, at each round the new prototypes are obtained as the solutions of a constrained optimization problem:

$$min_{\{p^v\}} \frac{1}{2} \sum_{v \in C} \|p^v - p_i^v\|$$

s.t. $\sum_{v \in Ancestors(c_i)} (p^v \cdot d_i) - \sum_{u \in Ancestors(c_j)} (p^u \cdot d_i) \geq \sqrt{distance(c_i, c_j)}$,
where p^v are new prototype vectors, p_i^v are current prototype vectors, $(d_i, c_i) \in D \times C$ is a given training instance, $c_j \in C$ is any class label, and $distance(c_i, c_j)$ is the number of edges between labels c_i and c_j in the hierarchical tree.

Another approach based on the maximum margin idea has also appeared in the literature [Tsochantaridis et al., 2004, Cai and Hofmann, 2004]. This approach builds on the work by Crammer and Singer [Crammer and Singer, 2001] that adapts the classical binary learning algorithm Support Vector Machines (SVM) to the multi-class settings. Consequently, Tsochantaridis et al. extend the multi-class SVM to the case of hierarchical classification. In Crammer and Singer's approach to multi-class SVM, the goal is to learn a discriminant function $F : D \times C \rightarrow \Re$, which is then maximized over the category set

C to predict a class for a given document $d \in D$:

$$f(d; w) = argmax_{c \in C} F(d, c; w),$$

where $w = (w_1, \ldots, w_{|C|})$ is a parameter vector. To account for the hierarchical settings, Tsochantaridis et al. assume F to be linear in a combined feature representation of inputs and outputs $\Phi(d, c)$ that encodes the relationships between classes in a hierarchy:

$$F(d, c; w) \equiv \langle w, \Phi(d, c) \rangle = \sum_{v \in Ancestors(c)} \lambda_v(c) \langle w_v, d \rangle ,$$

where $\langle ., . \rangle$ denotes the inner product between two vectors, and $\Lambda(c) = (\lambda_1(c), \ldots, \lambda_{|C|}(c))$ is an attribute vector defined as follows:

$$\lambda_v(c) = \begin{cases} z_v, \text{if } v \in Ancestors(c), \\ 0, \text{otherwise.} \end{cases}$$

The z_v are non-negative weights, which in the simplest case can be set to 1. Such a discriminant function takes account of all nodes along the path from a root to a specific leaf. In addition, the authors generalize the SVM optimization problem:

$$min_{w,\xi} \frac{1}{2} \|w\|^2 + \frac{Const}{|D|} \sum_{i=1}^{|D|} \xi_i, \quad s.t. \ \forall i, \xi_i \geq 0$$

$$\forall i, \forall c_j \in C \setminus c : \langle w, \delta\Phi_i(c_j) \rangle \geq 1 - \xi_i,$$

where $\delta\Phi_i(c_j) \equiv \Phi(d, c) - \Phi(d, c_j)$ and $|D|$ stands for the total number of training documents. This optimization problem is extended to the case of arbitrary loss functions Δ in two ways. The first method re-scales the slack variables ξ_i according to the loss incurred in each of the linear constraints:

$$\forall i, \forall c_j \in C \setminus c : \langle w, \delta\Phi_i(c_j) \rangle \geq 1 - \frac{\xi_i}{\Delta(c, c_j)}.$$

The second method re-scales the margin itself:

$$\forall i, \forall c_j \in C \setminus c : \langle w, \delta\Phi_i(c_j) \rangle \geq \Delta(c, c_j) - \xi_i.$$

The loss function $\Delta(c, c_j)$ reflects the class relationships in a hierarchy and is defined as

the height of the deepest common ancestor of c and c_j in a hierarchal graph [Tsochantaridis et al., 2004] or through the distance from c to c_j in a hierarchical tree [Cai and Hofmann, 2004].

Shrinkage is another global approach. Here a probabilistic classifier that discriminates between all leaf categories is built while its parameters (such as $P(t_k|c_i)$) are estimated based on the whole hierarchy. Since the low-level categories are usually small, the number of training examples is insufficient to make robust parameter estimates. To overcome this problem, smoothing of the parameter estimates of a child category with the estimates of its ancestors has been proposed [Greiner et al., 1997, McCallum et al., 1998, Toutanova et al., 2001, Gaussier et al., 2002]. Smoothing can be achieved by a linear interpolation of estimates of all hierarchy nodes from the child to the root:

$$P(t_k|c_i) = \sum_j \lambda_{i,j} \cdot P(t_k|v_j),$$

where sum is taken over all nodes v on the path from c_i to the root and $\sum_j \lambda_{i,j} = 1$. In the work by Greiner et al. the weights $\lambda_{i,j}$ are set to be inversely proportional to the variances of the corresponding estimates, where the variance in an internal node is the variance of the estimates among the children of the node [Greiner et al., 1997]. McCallum and colleagues, on the other hand, use the weights that maximize the likelihood of data [McCallum et al., 1998]. The weights can be found by the Expectation-Maximization (EM) algorithm [Dempster et al., 1977].

Toutanova and colleagues extend the work by McCallum et al. in that they use EM to compute the interpolation weights λs as well as the node conditional word distributions at inner nodes $P(t_k|v)$ to maximize the likelihood of data [Toutanova et al., 2001]. That allows them to do selective shrinkage, *i.e.* choosing more general words from the upper levels and very specialized ones from the leaf nodes. The same approach to the estimation of conditional word probabilities $P(t_k|c_i)$ is taken by Gaussier and colleagues [Gaussier et al., 2002]. In addition, they consider soft assignment of categories to documents and estimate $P(d|c_i)$ by maximizing the log-likelihood of a document (again with EM):

$$L(d) = \sum_k log \left(\sum_i P(c_i) P(d|c_i) \sum_j P(v_j|c_i) P(t_k|v_j) \right)$$

Completely different approach has been proposed by Frommholz [Frommholz, 2001]. Starting from the posterior probability estimates for each class provided by a "flat"

classifier, he calculates the final score of a category c_i based on the posterior probabilities of other categories and their proximity to c_i:

$$P_H(c|d) = \sum_i P(c_i|d) \cdot P(c_i) \cdot CS(c, c_i)$$

The set of class proximities $CS(c_i, c_j)$ can be given by a user or can be estimated in the following way. For two neighboring categories the class proximity is the proportion of shared terms. For two distant categories the class proximity is the sum over all paths between the nodes in a hierarchical tree of the products of class proximities of the neighboring categories on a path.

Overall, none of the previously introduced hierarchical global algorithms produces hierarchically consistent classifiers. As we argue in Section 2.3, hierarchical consistency is an integral element of hierarchical categorization. Therefore, we propose a novel hierarchical global approach that performs consistent classification (Chapter 4). Moreover, unlike previous global hierarchical methods that typically focus on a specific learning algorithm, *e.g.* association rules, decision trees, etc., the new approach is a general hierarchical framework in which any conventional "flat" multi-label learning algorithm could be adapted to the hierarchical setting.

3.2.2 Local approaches to hierarchical text learning

While many global approaches to hierarchical text categorization have been proposed in literature, local approaches are, nevertheless, more popular due to their computational benefits. In general, we can divide local approaches in two groups: pachinko machine (sequential Boolean decisions) and probabilistic (multiplicative decisions). In pachinko machine a decision which path in a tree to take is made sequentially at each level of a hierarchy, while in a probabilistic method all paths are considered simultaneously, their probabilities are calculated as the product of individual probabilities of categories on a path and the most probable path is picked as a solution.

The name, pachinko machine, comes from a Japanese gambling machine which is a mixture between a slot machine and pinball (see Figure 3.2). It looks like a vertical pinball where small steel balls thrown by a player fall down through a maze of nail-like pins. A player can control only the speed of releasing the balls; after that it is a game of chance. Most of the balls just fall down while some fall into special holes, which activates a slot machine with three spinning reels. A player wins if the same three pictures appear

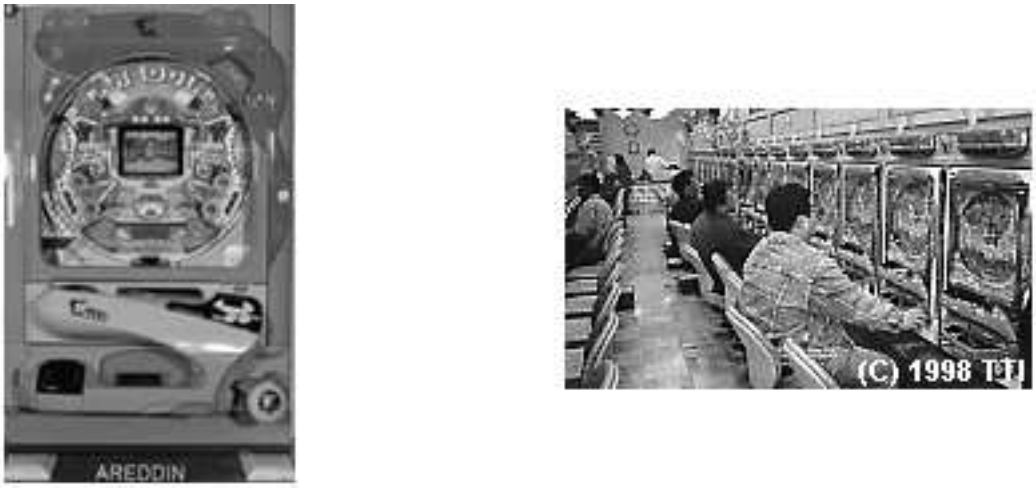


Figure 3.2: Pachinko machine (taken from http://slotsdirect.com).

in the slot machine.

As in a gambling counterpart, in a hierarchical pachinko machine the classification decisions are made in a top-down fashion iteratively and irreversibly. At each level the classifier selects one (or several, in case of multi-label categorization) most probable category(ies) for a test example and then proceeds down the hierarchy inspecting only the children of the selected nodes. This method has been widely used with different learning algorithms: linear [Ng et al., 1997, D'Alessio et al., 2000, Wibowo and Williams, 2002a], probabilistic [Koller and Sahami, 1997, Wibowo and Williams, 2002a], decision rules [Ipeirotis et al., 2001], neural networks [Ruiz and Srinivasan, 2002], and SVM [Sun and Lim, 2001, Wibowo and Williams, 2002a].

A probabilistic hierarchical local approach makes probabilistic decisions at each level of the hierarchy and then selects the leaf categories on the most probable paths. This method is more computationally expensive than pachinko machine since we have to learn classifiers at every node. Therefore, there have been just a few studies on this technique: it has been used in combination with probabilistic classifiers [Chakrabarti et al., 1997] and neural networks [Wiener et al., 1995, Weigend et al., 1999]. Dumais and Chen compared the two local approaches, pachinko machine and probabilistic, and found no difference in performance [Dumais and Chen, 2000]. Since pachinko machine is much more efficient, it is usually the choice in local approaches. Moreover, Mitchell [Mitchell, 1998] has proved that if we use the same probabilistic classifier, *e.g.* Naive Bayes, for hierarchical and "flat" approaches, the probability terms are estimated using a maximum

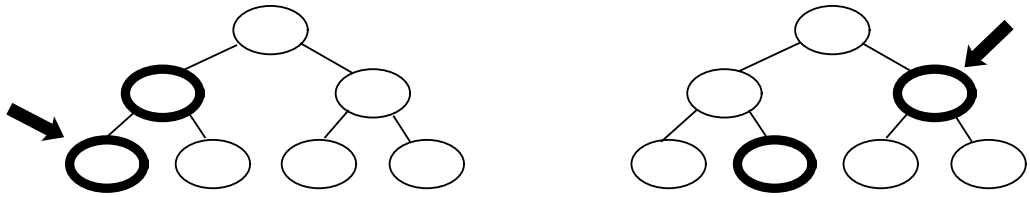


Figure 3.3: Error-correcting hierarchical method by Wibowo and Williams (2002). Ellipses in bold represent the categories assigned to the instance by a classifier at each level of the hierarchy; the ellipse with an arrow pointing to it is chosen as the final decision.

likelihood estimator, and the documents are represented using the same feature sets at each level of the hierarchy, then the probabilistic hierarchical classifier is equivalent to the standard non-hierarchical classifier.

An alternative local approach is Error-Correcting Codes (ECOC) [Ghani, 2000]. In this work the author makes some of the code bits to represent the high-level categories, but it results in reduced column separation and, thus, worse classification performance.

There has been some effort in the research community to cope with one of the major problems of the local hierarchical approach: high-level error recovering. Cheng and colleagues propose two methods [Cheng et al., 2001]. The first one is a pachinko machine with the possibility to return to the high-confidence ancestor node a few levels back if the classification probability drops below a given threshold. The second approach uses three different classification algorithms, two of which are standard pachinko machines with different feature sets and the third one is a classifier that dynamically skips some levels in the hierarchy. The decision is then made by the majority vote. Skipping the levels in a sense is a compromise between the hierarchical and "flat" approaches that can reduce the high-level errors. In the paper by Wibowo and Williams [Wibowo and Williams, 2002b] several methods are proposed, where "flat" classification is performed at each level of a class hierarchy and the most probable category is chosen if its parent is also the most probable category at the level above; otherwise the best parent category is selected (see Figure 3.3). Shifting the choice of a category higher in the hierarchy gives us a better chance of correct classification since, in general, classification at higher levels is more accurate.

Most of previous research on hierarchical local categorization focus on text collections with class hierarchies represented as trees and all documents assigned to leaf categories. However, a large number of real-world hierarchical text categorization tasks involve class hierarchies formed as true directed acyclic graphs (*e.g.* Gene Ontology), where documents

		true labels	
		c_i	$\bar{c}_i$
classified as	c_i	TP_i	FP_i
	$\bar{c}_i$	FN_i	TN_i

Table 3.2: The contingency matrix for category $c_i \in C$. $\bar{c}_i$ denotes all categories in C other than c_i. TP_i (true positives) is the number of documents correctly classified in c_i, TN_i (true negatives) is the number of documents correctly classified as not belonging to c_i, FP_i (false positives) is the number of documents incorrectly classified in c_i, and FN_i (false negatives) is the number of documents incorrectly classified as not belonging to c_i.

may belong to any category, internal or leaf. For that reason, in the present work we extend the original hierarchical local approach pachinko machine to the general case of DAG class hierarchies and internal class assignments (Chapter 4).

3.3 Performance evaluation

In this section we review the common "flat" and hierarchical evaluation measures. Later, in Chapter 5 we argue that the existing measures are not particularly suitable for evaluation of a hierarchical text categorization system. Therefore, we propose our own hierarchical measure that addresses the discussed issues (Chapter 5).

The performance of a text categorization system is usually measured in terms of its effectiveness, *i.e.* its ability to produce correct classification. Some researchers also compare the performance in terms of efficiency (for example, running time). Since text collections are typically large, the efficiency of a system is very important in real-life settings, yet the consistent comparisons can be tricky due to differences in computational resources, algorithm implementation, *etc.*

There exist several evaluation metrics to assess the effectiveness of a categorization system. The most common are accuracy, error rate, precision, recall, and combined measures of precision and recall. Given the contingency matrix in Table 3.2, these measures are defined as follows. For any category $c_i \in C$,

- Accuracy gives the percentage of correct decisions made by a classifier:

$$A_i = \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i}$$

- Its counterpart, error rate, gives the percentage of incorrect decisions:

$$E_i = 1 - A_i = \frac{FP_i + FN_i}{TP_i + TN_i + FP_i + FN_i}$$

- Precision is the percentage of correctly classified documents out of all documents classified to category c_i:

$$P_i = \frac{TP_i}{TP_i + FP_i}$$

- Recall is the percentage of correctly classified documents out of all documents in category c_i:

$$R_i = \frac{TP_i}{TP_i + FN_i}$$

Precision and recall are normally averaged over all categories in one of the following ways:

- macroaverage: averaging individual categories' precision/recall over all categories:

$$P = \frac{\sum_i P_i}{|C|} = \frac{\sum_i \frac{TP_i}{TP_i+FP_i}}{|C|} \qquad R = \frac{\sum_i R_i}{|C|} = \frac{\sum_i \frac{TP_i}{TP_i+FN_i}}{|C|},$$

  where $|C|$ is the total number of categories;

- microaverage: summing individual cells in contingency matrices, *i.e.* TP, FP, FN, TN, and then calculating precision/recall for a global matrix:

$$P = \frac{TP}{TP+FP} = \frac{\sum_i TP_i}{\sum_i (TP_i+FP_i)} \qquad R = \frac{TP}{TP+FN} = \frac{\sum_i TP_i}{\sum_i (TP_i+FN_i)}$$

Microaveraging puts more stress on the performance of high frequency categories while macroaveraging weights the performance of all categories equally.

Neither precision nor recall is a good measure in isolation. For example, a trivial classifier that classifies every document into a given class would get perfect recall (100%), but its precision would be very low. In general, high values of recall can be obtained at the cost of low precision and vice versa. As a result, a number of combination measures have been used in text categorization research:

- F-measure:
$$F_\beta = \frac{(\beta^2 + 1) \cdot P \cdot R}{(\beta^2 \cdot P + R)}, \beta \in [0, +\infty)$$
This measure has been proposed by van Rijsbergen [van Rijsbergen, 1979] and is widely used. It represents the weighted harmonic mean of precision and recall, which acts as a conservative average: for close values of precision and recall the F-measure is near the average of the two numbers, while for rather distant values the F-measure is shifted towards the smaller number. The standard value for β is 1 that gives precision and recall equal weights.

- 11-point average precision: the average precision for recall values 0, 0.1, ..., 0.9, 1.

- Breakeven point: the value at which precision is equal to recall. This measure has been proposed by Lewis [Lewis, 1992]. Recently, some significant drawbacks of this measure have been discovered [Sebastiani, 2002]. First, often no parameter settings would lead to equal values of precision and recall. In this situation the breakeven has to be interpolated, which yields artificial and unreliable results. Second, equal values of precision and recall are not necessarily the best and/or desirable point in the performance. Therefore, this measure is rarely used nowadays.

Most studies on text categorization assume equal costs of different kinds of error: a false positive (a document incorrectly assigned to a category) is considered as bad as a false negative (a document incorrectly not assigned to a category). Yet, some applications require non-unified misclassification costs. For example, classifying a cancerous tumor as non-malignant and, as a result, not treating the patient right is a more severe error than classifying a non-cancerous tumor as malignant. Hierarchical text categorization calls for cost-sensitive learning since a hierarchical structure itself presents built-in costs. For example, misclassifying a document in a sibling or parent of a correct category is intuitively preferable to misclassifying it to a very distant node. However, most work on hierarchical text categorization do not take into account the costs and evaluate systems based on standard measures, accuracy and precision/recall [Chakrabarti et al., 1997, Koller and Sahami, 1997, Dumais and Chen, 2000].

Only a few researchers noted that hierarchical categorization systems cannot be evaluated in a standard, "flat" way and, therefore, require special metrics. The first who pointed out the need of non-unified error costs were Wang and colleagues. In their work they assume that a set of misclassification costs for all category pairs $B(c_i, c_j)$ is given, and they optimize their association rule based classifier to minimize the overall

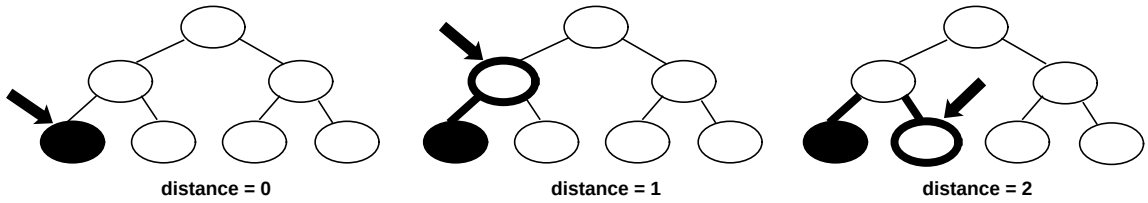


Figure 3.4: Distance-based hierarchical measure. The solid ellipse represents the real category of a test instance; the ellipse in bold (with an arrow pointing to it) represents the category assigned to the instance by a classifier; edges in bold represent the shortest (undirected) path between the real and assigned categories or, in other words, the distance-based error.

cost [Wang et al., 1999]. As a possible choice for misclassification costs they propose to use the distance between nodes representing categories in a hierarchical tree (see Figure 3.4). The distance between a correct and assigned category, $distance(c_i, c_j)$, is defined as the length of the shortest (undirected) path from node c_i to node c_j in a hierarchical graph. The same distance-based evaluation measure is used by Dekel et al. [Dekel et al., 2004].

In their subsequent work Wang and colleagues replace distance-based costs with similarity-based costs [Wang et al., 2001]. The similarity of two categories is measured as the similarity of the sets of documents belonging to the categories. Multi-label categorization is assumed, and the misclassification error between two sets of categories, assigned to a document by a classifier (C_i') and the real categories (C_i), is calculated as the normalized difference of the coverage of the two sets:

$$E(C_i, C_i') = \frac{|Cover(C_i') - Cover(C_i)| + |Cover(C_i) - Cover(C_i')|}{|Cover(C_i) \cup Cover(C_i')|},$$

where coverage of a category set $Cover(C_i)$ include all documents that are labeled with the categories from the set or with the categories whose ancestors are in the set.

Itskevitch extends Wang and colleagues' work introducing probabilistic hierarchical measure based on the distance between categories in a hierarchical tree [Itskevitch, 2001]. Since a document can be classified into several categories (multi-label categorization), the probabilistic score of a classifier on a document d_i is summed over all categories assigned to d_i:

$$A(d_i) = \sum_j A_j(d_i),$$

where $A_j(d_i)$ is calculated based on prediction probabilities and distances between correct

and assigned categories:

$$A_j(d_i) = \begin{cases} -\frac{P(c_j|d_i)}{distance(c_i,c_j)} & \text{for } distance(c_i,c_j) = 1,2,\ldots,D_{acc} \\ P(c_j|d_i)log_2\ distance(c_i,c_j) & \text{for } distance(c_i,c_j) = D_{acc}+1,\ldots,D_{max} \end{cases}$$

Here D_{max} denotes the maximum distance between two categories in a hierarchy and D_{acc} denotes the maximum acceptable distance. Distances in a hierarchy are increased by 1 to avoid zero division. Overall score values range from -1 to $log_2\ D_{max}$ with -1 being the best value.

Blockeel et al. also use a measure based on the distance between categories, but consider the weights for the edges of a hierarchy tree (see Figure 3.5a) [Blockeel et al., 2002]. The weights decrease exponentially with depth. Cai and Hofmann propose a more general measure based on a given set of user defined misclassification costs: $cost_1(v) \geq 0$ is the cost of assigning an item $d_i \in Offspring(v)$ to $w \notin Offspring(v)$, $cost_2(v) \geq 0$ is the cost of assigning an item $d_i \notin Offspring(v)$ to $w \in Offspring(v)$ [Cai and Hofmann, 2004]. Then, the overall loss is computed as the sum of these costs for nodes in the symmetric difference of the ancestor sets of the real and predicted category:

$$\Delta(c_i,c_j) = \sum_{v \in Ancestors(c_i), v \notin Ancestors(c_j)} cost_1(v) + \sum_{v \notin Ancestors(c_i), v \in Ancestors(c_j)} cost_2(v).$$

For uniform costs, this formula reduces to the standard distance-based error.

Another measure, weighted penalty, is also suggested by Blockeel et al. [Blockeel et al., 2002]. Here the nodes of the tree have weights (deeper nodes have smaller weights), and the distance between two nodes is calculated as the weight of their deepest common ancestor (see Figure 3.5b). Tsochantaridis et al. use a variant of this measure with weights defined as the height of a node in a hierarchical tree [Tsochantaridis et al., 2004]. Semantic similarity [Lord et al., 2003] can be seen as an extension of such a measure for the general case of a DAG hierarchy. The semantic similarity measure is based on the minimal weight over all common ancestors of the two nodes. This measure has been specifically designed for the Gene Ontology. The weights are calculated as the probabilities of a category (or any of its descendant nodes) occurrence in given data.

Sun and Lim propose their own category similarity and distance-based measures for hierarchical text categorization [Sun and Lim, 2001]. Their category similarity CS measure is based on the content of documents comprising the categories and is computed as the cosine similarity between the feature vectors of two categories. The distance be-

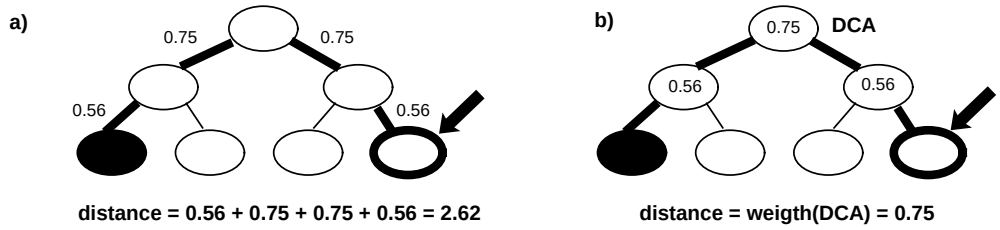


Figure 3.5: Weighted distance-based hierarchical measure. The solid ellipse represents the real category of a test instance; the ellipse in bold (with an arrow pointing to it) represents the category assigned to the instance by a classifier; edges in bold represent the shortest path between the real and assigned categories. In figure a) edges have weights that decrease with depth. The distance-based error is the sum of all weights on the shortest path between the real and assigned categories. In figure b) nodes have weights that decrease with depth. The distance-based error is the weight of the deepest common ancestor of the real and assigned categories.

tween two categories is the shortest distance between corresponding nodes in a hierarchy, similar to the one by Wang et al. Then, based on these measures, category similarity and distance between categories, they modify the standard measures, precision/recall and accuracy/error. They consider misclassification as being partly correct depending on how close the real and assigned categories are and add contributions from FP and FN to correct decisions (TP):

$$A_i = \frac{TP_i+TN_i+FPCon_i+FNCon_i}{TP_i+TN_i+FP_i+FN_i} \qquad E_i = \frac{FP_i+FN_i-FPCon_i-FNCon_i}{TP_i+TN_i+FP_i+FN_i}$$

$$P_i = \frac{max(0,TP_i+FPCon_i+FNCon_i)}{TP_i+FP_i+FNCon_i} \qquad R_i = \frac{max(0,TP_i+FPCon_i+FNCon_i)}{TP_i+FN_i+FPCon_i}$$

For category similarity based measures, the contribution of document d_j from FP(FN) to class c_i is defined as

$$Con(d_j, c_i) = \frac{\sum_k (CS(c_i, c_k) - ACS)}{1 - ACS},$$

where summation is done over all document categories (for FPCon) or over all categories assigned to a document (for FNCon) and the average category similarity ACS is defined as $ACS = \frac{2 \cdot \sum_{i=1}^{n} \sum_{k=i+1}^{n} CS(c_i,c_k)}{n \cdot (n-1)}$.

For distance based measures, the contribution of document d_j from FP(FN) to class c_i is defined as

$$Con(d_j, c_i) = \sum_k \left(1 - \frac{distance(c_i, c_k)}{D_{acc}}\right),$$

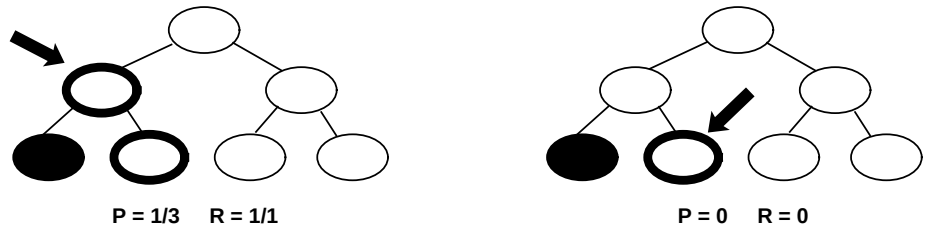


Figure 3.6: Hierarchical measure proposed by Ipeirotis et al. (2001). The solid ellipse represents the real category of a test instance; the ellipse in bold with an arrow pointing to it represents the category assigned to the instance by a classifier. Both real and assigned category sets are expanded to include all the descendant nodes. Then precision is calculated as the percentage of the overlap of the two sets in the total number of the assigned categories including the descendants, and recall is the percentage of the overlap in the total number of real categories including the descendants.

where summation is done over all document categories (for FPCon) or over all categories assigned to a document (for FNCon). Both types of contributions can be positive and negative and are restricted to the range $[-1, 1]$.

An unusual approach to evaluation has been proposed by Ipeirotis and colleagues [Ipeirotis et al., 2001]. Instead of measuring distance or content similarity of two category sets, correct C_i and predicted C'_i, they measure the overlap in subtrees induced by the category sets (see Figure 3.6):

$$P = \frac{|Expanded(C_i) \cap Expanded(C'_i)|}{|Expanded(C'_i)|}$$

$$R = \frac{|Expanded(C_i) \cap Expanded(C'_i)|}{|Expanded(C_i)|},$$

where $Expanded(C_i) = \{c_i \in C | c_i \in C_i \text{ or } \exists c_j \in C_i : c_j \in Ancestors(c_i)\}$.

Evidently, a variety of hierarchical evaluation measures has been proposed to date. These measures try to move away from the conventional "flat" techniques and capture some of the specifics of hierarchical classification. Unfortunately, there are no well established criteria on what aspects of hierarchical information are the most significant and therefore, should be addressed in an evaluation measure. This is reflected in the diversity of the proposed techniques. In present work (Chapter 5), we overcome this problem by formulating the natural, intuitive characteristics we expect a hierarchical evaluation measure to exhibit. We compare the existing hierarchical and non-hierarchical measures

based on those characteristics and show that none of them surpasses all the criteria. Consequently, we propose a new hierarchical measure that possesses all the required properties. Furthermore, we compare our new measure with the conventional "flat" measures based on the concepts of consistency and discriminancy [Huang and Ling, 2005], the first technique to systematically compare classifier performance measures.

3.4 Applications

Most of the research ideas described in previous sections were tested on classical text corpora Reuters-21578 (Reuters-22173)[5] and 20-newsgroups [Lang, 1995] and different subsets of real-life web directories Yahoo! and DMOZ[6]. Other real-life text collections with built-in hierarchies include US Patent database[7], ACM Digital Library[8], LookSmart Web directory[9], and biomedical articles OHSUMED [Hersh et al., 1994].

In addition to this extensive set of text data, hierarchical text categorization has been successfully applied to other domains. Email classification is a subdomain of text categorization where topic hierarchies are often present [Agrawal et al., 2000, Itskevitch, 2001, Zhdanova and Shishkin, 2002]. Many users organize (or would like to organize) their incoming email into a hierarchy of folders. Also, an automatic forwarding system can have the forwarding addresses organized in a hierarchy, for example by department.

On the web there exist various topic hierarchies in the form of web directories that can be used not only to classify web pages but also to classify web databases. There are many web-accessible databases, such as Medline or ZDNet Product Review[10], whose content is only accessible through search interfaces. For these resources, a special kind of categorization, database categorization is needed [Ipeirotis et al., 2001].

Another interesting application of hierarchical text categorization is a user interface for search engines [Chen and Dumais, 2000, Lin et al., 2002]. Instead of returning a long flat list of documents that match a user's query, these systems return the search results organized into an existing topic hierarchy. Such result presentation greatly helps users to quickly find the information they need [Chen and Dumais, 2000]. This is achieved by simply classifying the results returned by a search engine into a predefined topic

[5] http://www.daviddlewis.com/resources/testcollections/reuters21578/
[6] http://www.dmoz.com/
[7] http://www.uspto.gov/
[8] http://portal.acm.org/dl.cfm
[9] http://www.looksmart.com/
[10] http://reviews-zdnet.com.com/

hierarchy [Chen and Dumais, 2000]. However, a more efficient approach is to perform search in two steps: first, the categories that match a user's query are identified and then, documents only from these categories that match the query are presented to the user [Lin et al., 2002]. The same approach can be applied to personalization systems where the search results are filtered and re-ranked according to a user's preferences organized in a topic hierarchy, and then presented in a hierarchical structure [Pretschner and Gauch, 1999, Chen et al., 2002].

Hierarchical text categorization can be useful for question answering [Li and Roth, 2002]. Before an automatic system attempts to answer a free form factual question from a given large collection of texts, it may be useful to first categorize the question into one of the semantic types from a given type hierarchy. For example, given the question "Who was the first female astronaut?" we know that the required answer is a person (or group of people) because of "who" and more specifically, that it is a woman. These semantic categories provide some constraints on the types of answers that should be found at later stages of the question answering process.

3.5 Text learning in bioinformatics

In this work, in addition to testing our new ideas on hierarchical learning and evaluation on classical text corpora, we explore the application of hierarchical text categorization to molecular biology. Computational methods have been applied in life sciences for several decades now. Moreover, a new discipline, bioinformatics, has emerged in the mid-1980s. Roughly speaking, bioinformatics is a merger of computer science and biology. It deals with storing, analyzing, and visualizing any kind of biological data.

More recently, an advantage of text analysis in bioinformatics has been recognized. A vast amount of biological and medical knowledge has been accumulated in scientific journal publications. The largest public corpus of biological literature is the database of the National Library of Medicine (NLM), Medline. It has an online access through PubMed[11], a service by the National Center for Biotechnology Information (NCBI) at the National Institutes of Health (NIH). PubMed includes bibliographic citations and author abstracts from more than 4,600 biomedical journals published in the United States and 70 other countries. It contains over 15 million citations dating back to the 1950s. New citations are added on a weekly basis.

[11] http://www.ncbi.nih.gov/entrez/query.fcgi

This literature is essential for life scientists at all stages of their research, from planning and performing experiments to analyzing and putting in the context the results. Manually processing such a large corpus is tedious and very time-consuming; therefore, automatic methods of text analysis are needed. As a result, data mining, machine learning, and natural language processing (NLP) techniques have been applied to a wide range of problems in bioinformatics: functional annotation of genes and proteins, finding gene-gene or protein-protein interactions, knowledge discovery, to name a few [Yandell and Majoros, 2002,de Bruijn and Martin, 2002]. The recently developed BLIMP (Biomedical LIterature and text Mining Publications)[12] service collects all research publications in the field of biomedical literature mining and represents a valuable source of information on this topic. Below we describe a few classical text-related bioinformatics tasks and some of the solutions proposed so far by AI researchers.

3.5.1 Information retrieval

Information retrieval (IR) in bioinformatics has the same goal as general information retrieval: to select documents relevant to a user's request from a large collection of biological and medical texts. In some cases, an IR system is also required to rank the retrieved documents according to their degree of relevancy. A user's request can be formulated as a query, *i.e.* words that should or should not appear in the retrieved papers, or a set of papers that are known to be relevant, *i.e.* the training set. NCBI provides an engine, PubMed, to retrieve documents from Medline according to a user's query. While this mechanism allows users to reduce significantly the number of documents for subsequent manual processing, formulating a good query requires some skill and experience. The main challenge in query composition is that authors use diverse terminology and nomenclatures to describe biological entities. This problem can be addressed with machine learning techniques that can automatically distinguish the words that are highly representative of a given topic from the irrelevant terms by observing example documents in the training set [Dobrokhotov et al., 2003].

Information retrieval is a fundamental part of any life science research. In addition to helping users efficiently search for relevant information, it also can help database curators to select the most promising documents for reviewing. There has been some effort to structure the accumulated biological knowledge. The resulting databases, such as SGD [Dolinski et al., 2003] and Flybase [Consortium, 2002], are currently manually

[12]http://blimp.cs.queensu.ca/

replenished by professional curators who examine journal articles and extract new facts to annotate the entries of the database. Automatically selecting the articles that are worth reviewing will be highly beneficial for curators.

To generate more interest in this problem and evaluate the existing approaches, several international competitions were organized. The well-known Text Retrieval Conference (TREC)[13], whose goal is to encourage and support research in general information retrieval by providing the infrastructure for large-scale evaluation of the current IR techniques, has been organizing a special Genomics Track for the past few years. At the first TREC Genomics competition in 2003, the main IR task was the following: from a given set of Medline abstracts, retrieve documents that focus on the basic biology of a given gene or its protein products, *i.e.* location, structure, genetics, and functions, from a given organism [Hersh and Bhupatiraju, 2003]. In 2004, the focus of the Genomics Track shifted towards the conventional searching, *i.e.* the search topics were developed from the information needs of real biologists [Hersh et al., 2004]. Similarly, in the TREC 2005 competition, the search topics were collected from biologists, yet were more structured than the mostly free-form topics of the 2004 track [Hersh et al., 2005]. The idea was to provide better defined, while still realistic, queries to make easier for the IR systems the use of other resources, such as ontologies or databases. Another competition, the KDD Challenge Cup 2002[14] had one of its tasks to rank the documents according to their relevancy to the FlyBase database [Yeh et al., 2003]. A document was considered relevant if it contained experimental evidence for gene products (RNA and proteins). This competition showed that fully automated systems based on standard machine learning approaches are still inferior to manually devised pattern-matching rules.

3.5.2 Summarization

Summarization in general is the task of producing a short summary of a given document or a set of documents. The summary should convey the general idea of the document and its length should be just a fraction of the length of the original document. The simplest version of summaries is keywords, which usually does not require a deep natural language analysis. In bioinformatics summarization can be a very useful tool presenting a user short summaries of large sets of articles. Andrade and Bork [Andrade and Bork, 2000] present a statistical approach to extracting keywords for a set of Medline articles

[13] http://trec.nist.gov/

[14] http://www.biostat.wisc.edu/~craven/kddcup/tasks.html

describing a particular disease from the OMIM (Online Mendelian Inheritance in Man) database [McKusick, 1994]. Another statistical method by Shatkay and colleagues produces a list of keywords characterizing the literature on a specific gene [Shatkay et al., 2000]. Keywords for a set of articles represent a set of words highly relevant to a given topic, *e.g.* a disease or a gene. Such keywords allow a user to quickly assess the content of a large set of papers of interest.

3.5.3 Named entity recognition

The goal of this task is to recognize biological entities, such as genes and proteins, in the text. The straightforward solution to this task is to use a dictionary of these terms. However, this solution is limited since not always such a dictionary exists, and even when it does exist it is not complete. New genes/proteins are being discovered and named on a daily basis. In addition, the lack in naming conventions leads to very inconsistent use of gene/protein names. For example, most genes have several names and symbols to represent them; one symbol can be used to represent different genes; some gene symbols coincide with common English words, such as a, is, her, can, that are considered stop words in text learning and usually removed at the preprocessing step.

A number of automatic and semi-automatic techniques have been applied to this problem. For more standardized subdomains, like protein names, there exist some character clues, such as capital letters and numerical and special symbols, that help recognize the biological terms. These clues along with some hand-crafted domain-specific rules can produce a very accurate prediction system [Fukuda et al., 1998]. For other, less standardized subdomains more complex algorithms are required. Proux and colleagues combine a part-of-speech (POS) tagger with lexical and contextual hand-crafted rules and domain-specific dictionaries to detect gene names from sentences extracted from the Flybase database with 94.4% recall and 91.4% precision [Proux et al., 1998].

To avoid manual construction of domain-specific rules for every subtask at hand, some researchers apply machine learning techniques to learn the contextual rules automatically. Collier et al. adapt Hidden Markov Models (HMM) to predict biological named entities based on the sequences of two adjacent words (bi-grams) and their characteristics such as presence of digits, capital letters and so on [Collier et al., 2000]. Statistical supervised methods are used in the work by Nobata et al. to recognize biological terms [Nobata et al., 1999]. While these techniques are much more scalable and generalizable, their accuracy (F-score of $0.3 \ldots 0.7$) is not yet comparable to that of hand-crafted rules. Bunescu et

al. compare several machine learning methods, such as rule learners, Support Vector Machines and HMM, with hand-crafted rules on the dataset of human gene names and show that automatic methods can outperform the manually created classifiers [Bunescu et al., 2003]. However, in this study the authors use rules initially made for genes of other organisms, and applying them on human genes makes the comparison unfair since the name conventions differ significantly from one organism to another.

3.5.4 Entity relationship detection

This task consists of identifying groups of biological entities and their relations, such as protein-protein interactions. The most common approach to this problem is finding entities that frequently co-occur in the same sentence or abstract [Stapley and Benoit, 2000, Jenssen et al., 2001, Stephens et al., 2001, Cooper, 2003]. The experimental studies give evidence that co-occurrence usually correctly reflects meaningful relationships between entities. As usual, there is a trade-off between precision and recall: looking for co-occurrences at the sentence level typically increases precision while looking for co-occurrences at the abstract level increases recall [Cooper, 2003]. To increase precision researchers employed statistical measures to estimate if the co-occurrence frequencies are significant enough to represent a meaningful relationship [Stapley and Benoit, 2000, Stephens et al., 2001, Cooper, 2003]. These methods are based on the assumption of redundancy in the biological texts: the true related entities would appear together in several documents.

Another approach to increase the precision of finding related entities is to go deeper into the sentence analysis. Researchers apply shallow or deep sentence parsing and look for hand-crafted linguistic templates built around specific verbs representing interactions [Sekimizu et al., 1998, Blaschke et al., 1999, Proux et al., 2000, Thomas et al., 2000]. Subjects and objects of such verbs, if entities of interest, give a pair of related entities with the type of interaction indicated by the verb. Bunescu et al. design a completely automatic method of learning rules to recognize protein-protein interactions that can achieve higher precision than hand-crafted rules at cost of lower recall or higher recall at cost of lower precision [Bunescu et al., 2003]. Since automatic rule learning requires deep semantic analysis, which is computationally expensive, Nédellec et al. propose to select the relevant fragments of texts first, and then apply extensive NLP techniques only on them [Nédellec et al., 2001]. They classify sentences as containing potentially relevant information with standard machine learning algorithms, C4.5 and Naive Bayes. Given

the large sparseness of the facts to be extracted in the biomedical domain, this approach can significantly reduce the amount of data that need to be processed at the next steps, semantic-conceptual analysis and pattern matching.

3.5.5 Functional annotation

One of the main tasks in molecular biology is to identify the functions of genes and their products. Functions of thousands of gene products for many organisms have been identified and are described in the literature. However, the vocabulary used to describe the functions is not standardized and automatically retrieving the gene functions from literature is not obvious. There have been several attempts to standardize the vocabulary (for example, Gene Ontology (GO) [Ashburner et al., 2000]). Yet, mixed terminology is still present in articles, especially in older ones.

The text-related approach to functional annotation in general corresponds to classifying articles describing a particular gene into one or several functional categories. Then, the discovered categories are assigned to the gene. The straightforward technique of classifying Medline abstracts into GO terms using standard machine learning algorithms (maximum entropy, Naive Bayes, and nearest neighbor) has been proposed by Raychaudhuri et al. and showed promising results [Raychaudhuri et al., 2002]. After each article associated with a gene has been classified into GO terms, the gene function is chosen based on a weighted voting scheme. In this work, the experiments are performed only on 21 chosen genes. Catona and colleagues ran an extended set of experiments covering all 6295 human proteins annotated by that time in the Swiss-Prot database[15] [Catona et al., 2004]. Similar approach is employed in the Euclid system [Tamames et al., 1998]. They use a simple keyword-based classifier to assign functions to sequences from the Swiss-Prot database based on the detailed free-text functional annotations provided by human experts.

Functional annotation from biomedical literature has also been the focus of the BioCreAtIvE - Critical Assessment of Information Extraction systems in Biology (2004)[16] competition (task 2). The following task has been presented: for a given document and a given protein the participating systems have to identify a Gene Ontology code that corresponds to the protein's functionality described in the document. In addition, they

[15]Swiss-Prot database is an annotated database of protein sequences maintained by the Swiss Institute for Bioinformatics (SIB) and the European Bioinformatics Institute (EBI) (http://www.ebi.ac.uk/swissprot/).

[16]http://www.pdg.cnb.uam.es/BioLINK/BioCreative.eval.html

have to provide a text segment that supports such an annotation. Because of the second part, most of the participants employ NLP approaches to this task. Some systems try to match Gene Ontology terms with words occurring in a document and return the sentence with maximum matching score [Couto et al., 2005, Ehrler et al., 2005]. Other two systems match not only Gene Ontology terms, but also protein names, along with as many their lexical variants and synonyms as possible [Krallinger et al., 2005, Krymolowski et al., 2004]. Chiang and Yu enrich this simple matching with shallow parsing and manually constructed phrasal patterns [Chiang and Yu, 2004]. Three systems present machine learning approaches to the task. Rice et al. employ Support Vector Machines on carefully constructed feature vectors representing given documents [Rice et al., 2005]. Another system learns two statistical models. The first one identifies what Gene Ontology terms are relevant to a given document while the second one decides which of these GO terms should be returned as annotations for a given protein [Ray and Craven, 2005]. All of the approaches mentioned above employ the hierarchical information present in the Gene Ontology at most to the extent of populating training data of some node with the examples from its descendant nodes. Other than that, they address the task in the "flat" manner classifying each document into one or several GO codes from a set of available codes. Only one study recognizes this problem as a hierarchical one [Verspoor et al., 2005]. This work presents a system that first identifies the relevant GO terms for a given document through a direct matching or presence of words that often co-occur with GO terms. Then, it employs the hierarchical approach called Gene Ontology Categorizer [Joslyn, 2004]. This approach finds the best set of GO nodes that summarizes the relevant terms in the GO graph.

Several researchers address the problem of verifying if a given set of genes shares a function [Shatkay et al., 2000, Raychaudhuri et al., 2003, Raychaudhuri and Altman, 2003]. This task is especially important in microarray analysis (see Section 3.5.6). The usual practice in microarray analysis is to cluster genes by their expression profiles. If genes in a cluster share functionality, the cluster is considered interesting and is a good candidate for the follow-up studies. Shared functionality verification is based on the assumption that genes share functionality if the articles describing these genes share the content. So, the suggested approach works as follows [Raychaudhuri and Altman, 2003]. Suppose, we are given a set of genes $G = \{g_j\}, j = 1 \dots N$. We look at the sets of articles D_i similar in content to a document d_i that describes a particular gene $g_i \in G$. Next, for each gene $g_i \in G$ we calculate how many of these documents D_i also refer to other genes in the set G. Let's say that the number of such documents is n_i. Then, if

the numbers $n_i, i = 1 \ldots N$ are significantly larger than what would be expected for a random group of genes, then the group of interest is considered functionally coherent. Masys et al. apply a similar idea to biomedical terms assigned to documents describing a given set of genes [Masys et al., 2001]. In the Medline database articles are indexed with terms from biological ontologies such as Medical Subject Headings (MeSH) and Enzyme Commission (EC) codes. Masys et al. look for the MeSH and EC terms assigned to documents that describe genes from a given set. Figure 3.7 shows an example of such MeSH term analysis. Shown are the disease-related MeSH terms for genes predictive of Acute Lymphoblastic Leukemia as was identified in the comparative study of two types of leukemia by Golub et al. [Golub et al., 1999]. The terms form a sub-hierarchy of the original MeSH ontology. Numbers in parentheses beside each term represent the total number of matching gene and Medline citation records associated with the term and are linked to the information on the corresponding genes and Medline publications. Numbers in curly brackets are P-value estimates representing likelihood that this number of term matches would occur by chance. Such analysis can show not only whether a set of genes shares functionality, but also what functionality (expressed in MeSH/EC terms) it shares.

King et al. exploit the observation that some functions are related to each other and if a gene is assigned one function it should probably be assigned the other too [King et al., 2003]. They model the relations between GO terms with standard machine learning algorithms: decision trees and Bayesian networks.

There has also been some work on predicting gene/protein functions by sequence similarity analysis. Often proteins that have similar amino acid sequences share a function. Renner and Aszódi employ this dependency by scanning several gene and protein databases to retrieve the annotations for genes/proteins with sequences similar to those of a novel gene [Renner and Aszódi, 2000]. Then, they group the annotations with similar keywords into clusters to reflect different functions that the protein has.

3.5.6 Gene expression analysis

With the invention of microarrays gene expression analysis has gone to a new level. Evidently, genes and their products in any living organism interact with each other in complex ways. However, with traditional methods in molecular biology, the expression level of only one gene at a time could be measured; therefore, the whole picture was hard to obtain. DNA microarrays is a technology that allows measuring the expression level

```
Virus Diseases (2) {<.001}
Neoplasms (10) {<.001}
   Neoplasms by Histologic Type (8) {<.001}
      Leukemia (8) {<.001}
         Leukemia, Lymphocytic (6) {<.001}
            Leukemia, B-Cell (2) {<.001}
               Leukemia, B-Cell, Acute (2) {<.001}
            Leukemia, Lymphocytic, Acute (2) {<.001}
               Leukemia, B-Cell, Acute (2) {<.001}
            Leukemia, T-Cell (2) {<.001}
   Precancerous Conditions (2) {<.001}
      Preleukemia (2) {<.001}
Nervous System Diseases (4) {<.001}
   Autoimmune Diseases of the Nervous System (2) {<.001}
      Demyelinating Autoimmune Diseases, CNS (2) {<.001}
         Multiple Sclerosis (2) {<.001}
   Demyelinating Diseases (2) {<.001}
      Demyelinating Autoimmune Diseases, CNS (2) {<.001}
         Multiple Sclerosis (2) {<.001}
Female Genital Diseases and Pregnancy Complications (1) {<.001}
   Genital Diseases, Female (1) {<.001}
      Infertility (1) {<.001}
         Infertility, Female (1) {<.001}
Hemic and Lymphatic Diseases (2) {<.01}
   Hematologic Diseases (2) {<.001}
      Preleukemia (2) {<.001}
Neonatal Diseases and Abnormalities (5) {<.001}
   Hereditary Diseases (3) {<.001}
      Werner Syndrome (1) {<.001}
   Infant, Newborn, Diseases (2) {<.001}
      Severe Combined Immunodeficiency (2) {<.001}
Immunologic Diseases (8) {<.001}
   Autoimmune Diseases (2) {<.001}
      Autoimmune Diseases of the Nervous System (2) {<.001}
         Demyelinating Autoimmune Diseases, CNS (2) {<.001}
            Multiple Sclerosis (2) {<.001}
   Immunologic Deficiency Syndromes (6) {<.001}
      Common Variable Immunodeficiency (2) {<.001}
      Severe Combined Immunodeficiency (2) {<.001}
Pathological Conditions, Signs and Symptoms (1) {<.01}
   Pathologic Processes (1) {<.001}
      Disease Attributes (1) {<.001}
         Acute Disease (1) {<.001}
```

Figure 3.7: An example of concept hierarchy matches by Masys et al. (2001). Shown are disease-related MeSH terms for Acute Lymphoblastic Leukemia predictive genes identified by Golub et al. [Golub et al., 1999]. Numbers in parentheses represent the total number of matching records. Numbers in curly brackets are P-value estimates representing likelihood that this number of term matches would occur by chance.

of thousands of genes simultaneously. The purpose of such experiments is to learn the behavior of genes under different conditions (*e.g.* cell starvation, heat shock), in different tissue samples (*e.g.* different cancer types), or at different time points during a biological process (*e.g.* cell cycle). Gene expression analysis then facilitates function prediction for poorly characterized genes, discovery of potential drug targets, and differentiation between types of diseases.

While new microarray data are obtained on a daily basis, their analysis remains a complex task. Experiments of such scale are very difficult to analyze manually. Normally, bioinformaticists apply various techniques, such as clustering, to produce a manageable number of groups of genes with similar expression profiles. Eisen et al. experimentally showed that gene clusters tend to correspond to functional categories and, therefore, can shed light on functions of novel or poorly characterized genes [Eisen et al., 1998].

There exist a wide variety of unsupervised clustering techniques that can be applied to gene expression analysis [Shamir and Sharan, 2002]. The algorithms can be divided into hierarchical and centroid algorithms. Hierarchical algorithms produce a hierarchy of clusters where each cluster consists of its children clusters. The number of children is typically two. The hierarchical algorithms are agglomerative if they work in a bottom-up manner starting with clusters of single objects and recursively merging them into larger clusters. Divisive hierarchical algorithms, on the contrary, work in a top-down manner starting with one cluster containing all objects and recursively dividing it into smaller clusters.

Centroid algorithms generally require the specification of the number of clusters k to produce. They start with k random or specifically chosen points called centroids and group the objects around these points. Then they iteratively adjust the positions of the centroids and re-calculate the cluster membership for all objects until a certain criterion is optimized. K-means and Self-Organizing Maps (SOM) are the most common centroid algorithms.

Most clustering algorithms require the specification of a distance between a pair of objects. Distance measures can emphasize only positive correlations between objects, both positive and negative correlations, or more complex relationships making use of mutual information. Some of the common distance measures are presented in Table 3.3.

The most commonly used measure is Euclidean distance. It is a simple and intuitive measure that represents the shortest distance between two points (see Figure 3.8). It has a number of variations that differ from the original formula by attribute weights w_i. In standard Euclidean distance all attributes have equal weights of 1. However,

Distance Measure	**Mathematical Form**
Euclidean	$D(x,y) = \sqrt{\sum w_i(x_i - y_i)^2}$
Euclidean (inverse of variance)	$D(x,y) = \sqrt{\sum \frac{1}{\sigma_i^2}(x_i - y_i)^2}$
Euclidean (inverse of max deviation)	$D(x,y) = \sqrt{\sum \frac{1}{max(x_i - y_i)^2}(x_i - y_i)^2}$
Euclidean (Clark)	$D(x,y) = \sqrt{\sum \frac{1}{(x_i + y_i)^2}(x_i - y_i)^2}$
Minkowski	$D(x,y) = (\sum w_i \lvert x_i - y_i \rvert^r)^{\frac{1}{r}}$
Manhattan	$D(x,y) = \sum w_i \lvert x_i - y_i \rvert$
Manhattan (attribute mean)	$D(x,y) = \sum \frac{1}{\bar{X}_i} \lvert x_i - y_i \rvert$
Chebychev	$D(x,y) = max \lvert x_i - y_i \rvert$
Pearson correlation	$r_{xy} = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$

Table 3.3: Main functions for measuring distance in the clustering process. Here $\bar{x}$ denotes the mean value of vector x; $\bar{X}_i$ denotes the mean of the i^{th} attribute of all data points.

these weights can be set to the inverse of variance, the inverse of maximal deviation or other values including user-defined. Minkowski distance is a generalization of Euclidean distance where the exponent can be set to any user-defined value.

Manhattan distance represents the sum of the absolute distances between attribute values of two vectors (see Figure 3.8). It also has several variants that differ in the way of defining the attribute weights w_i. Chebychev distance is similar to Manhattan distance but it calculates the maximum absolute distance of two vectors (see Figure 3.8) instead of the sum of absolute distances.

Another widely used measure is Pearson correlation. This measure shows correlation between attribute values of two vectors with value of 1 for identical vectors and -1 for perfect opposites. To get the distance between two vectors we can use $1 - r$ to capture only positive correlations between vectors or $1 - |r|$ to capture both positive and negative correlations.

After genes have been grouped into clusters, each cluster has to be analyzed, in most cases manually, to find subsets of genes with similar biological properties, *i.e.* genes that share not only expression profiles, but also functionality. Such subsets of genes would

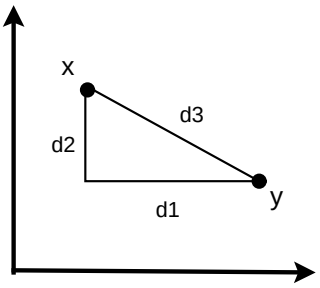


Figure 3.8: Distance measures used in clustering. $d3$ gives the Euclidean distance between vectors x and y, $d1 + d2$ gives the Manhattan distance, and $d1 (d1 > d2)$ gives the Chebychev distance.

represent the biologically meaningful chunks of information and will be the objects of further studies. There have been some attempts on automating this process as well. Speer and colleagues suggest a technique to automatically cluster given groups of genes by their functional categories in the Gene Ontology [Speer et al., 2004a]. For this, they apply a regular clustering algorithm with the distance function defined on the functional space of the Gene Ontology. The distance between two GO annotations can be defined as the distance between the two nodes in the GO graph, *i.e.* the length of the path between the two nodes. In the case of multiple paths, the average or the minimal path is used. The semantic similarity [Resnik, 1995, Lord et al., 2003] and its variants [Lin, 1998, Jiang and Conrath, 1998] have been suggested as information-theoretic alternative methods to measure the similarity in the ontology. To calculate these measures, a set of weights is required that are defined as the probabilities $p(c)$ of encountering nodes $c' \in Offspring(c)$ in data. These weights reflect the amount of information that a node carries. The semantic similarity of two nodes c_i and c_j is the negative logarithm of their common ancestor with the minimal weight:

$$sim(c_i, c_j) = max_{c \in S(c_i, c_j)}[-log(p(c))],$$

where $S(c_i, c_j) = Ancestors(c_i) \cap Ancestors(c_j)$ [Resnik, 1995, Lord et al., 2003]. Lin's variant of this measure additionally takes into account the weights of the query terms c_i and c_j [Lin, 1998]:

$$sim(c_i, c_j) = \frac{2 \cdot max_{c \in S(c_i, c_j)}[log(p(c))]}{log(p(c_i)) + log(p(c_j))}$$

Jiang and Conrath propose the distance measure based on the same idea [Jiang and

Conrath, 1998]:

$$d(c_i, c_j) = 2 \cdot max_{c \in S(c_i, c_j)}[log(p(c))] - [log(p(c_i)) + log(p(c_j))].$$

Since a gene can have multiple annotations, the average (or maximum/minimum) similarity/distance over all pairs of annotations of two genes is taken.

Background knowledge can also come useful as a judging criterion of a cluster quality: clusters composed of genes participating in the same or closely related biological processes are the first candidates for human analysis and possibly follow-up studies. Conventional cluster quality measures, such as Silhouette index [Rousseeuw, 1987] and Davies-Bouldin index [Davies and Bouldin, 1979], would evaluate cluster compactness and distinctness in the functional space of the Gene Ontology if one of the functional distances defined above is employed with these indices [Speer et al., 2005]. Another cluster quality measure, cluster stability [Famili et al., 2004], does not depend on the notion of distance between genes and, therefore, can be applied here without any modifications. The same techniques can be used to find the optimal clustering parameters, *e.g.* the number of clusters, on given data [Bolshakova et al., 2005] or compare the results of different clustering techniques. Alternatively, the mutual information between cluster membership and gene function may play a role of a cluster quality measure [Gibbons and Roth, 2002]. If the functional information is not readily available in databases, we can turn to the biomedical literature. Several techniques determining the functional coherence of a group of genes based on the similarity of the literature describing these genes have been proposed recently (see Section 3.5.5).

Another interesting way of adding up the background knowledge is incorporating it into the clustering process directly. Wang et al. experimentally show that high correlation in gene expression profiles often implies strong functional similarity [Wang et al., 2004]. However, microarray data are generally very noisy. As a result, cluster boundaries might be arbitrary to some degree. Using another information source, such as functional annotations, can help resolve these ambiguities and result in high-quality, biologically meaningful clusters. For example, we can combine conventional distances defined on expression data with functional distances defined on the Gene Ontology [Speer et al., 2004b]. Hanisch et al. employ this idea to co-cluster gene expression data with biological networks, namely metabolic reactions [Hanisch et al., 2002]. Another approach is proposed by Liu and colleagues [Liu et al., 2004]. This work is focused on building a tree of clusters grouping genes by their similarity in expression on a subset of experimental

conditions. A child cluster has one more condition added comparing to its parent cluster. In addition, the cluster tree is built in such a way that the cluster relationships reflect the functional relationships of genes in the Gene Ontology. Again, if the background knowledge is not available directly, the biomedical literature can be used instead. Glenisson et al. exploit the above mentioned idea of distance combination, integrating gene expression data with free-form textual information on the analyzed genes [Glenisson et al., 2003]. Raychaudhuri and colleagues also combine gene expression analysis and text analysis to search for sets of genes with similar expression patterns and shared functionality [Raychaudhuri et al., 2003]. They separate a group of genes with a linear projection in gene expression data. Then, iteratively, starting from a given (possibly random) group of genes, they refine the group to find the best group (local maximum) in terms of functional similarity. To determine if the group of genes shares a common function they use the biomedical literature and a statistical method described in Section 3.5.5.

Another direction in gene expression analysis is applying supervised learning methods to classify expression patterns in a predefined set of classes [Brown et al., 1999, Hvidsten et al., 2003]. For example, Hvidsten and colleagues design a rule-based system to classify gene expression profiles into Gene Ontology codes to infer the functional properties of the corresponding genes [Hvidsten et al., 2003]. Also, supervised learning techniques can be applied to identify genes whose expressions are highly characteristic of the studied conditions/tissues [Furey et al., 2000, Do and Poulet, 2003, Long and Vega, 2003]. In this problem, genes become attributes, and conditions become classes. Then, any machine learning algorithm can be applied to learn the subsets of genes that tell apart the conditions. However, gene expression data pose a tremendous challenge for learning algorithms as the data are generally characterized by a high level of noise and, more importantly, an enormous number of attributes and an inadequate amount of training samples. In a typical microarray study, expressions of a few thousands of genes (attributes) are measured in only a few tens or hundreds of samples. Therefore, Support Vector Machines is a good candidate for this classification problem since it is very robust in noisy environments and efficient in high-dimensional feature spaces [Furey et al., 2000, Do and Poulet, 2003]. Alternatively, feature selection techniques can be brought into play to reduce the dimensionality of the feature space yet preserving most of the information present in the original data.

3.5.7 Creating/maintaining knowledge databases

All the tasks mentioned above are essential steps in a global task of automatic or semi-automatic creation and maintenance of biological databases [Craven and Kumlien, 1999]. One of the central goals of bioinformatics is to structure the biological knowledge accumulated to date and store it in databases, so that it can be easily accessible by automatic systems. Nowadays, most of the work on database creation and maintenance is done manually, which requires significant resources in terms of experts' time. To do this automatically, we have to produce a system capable of doing several tasks, for example named entity recognition and entity relationship detection, or named entity recognition and functional annotation. Another possibility is to assist experts in their annotation work. For instance, Ohta and colleagues use information retrieval and summarization modules along with a domain-specific dictionary construction to assist biologists in maintaining the Transcription Factor database (TFDB) [Ohta et al., 1997].

3.5.8 Knowledge discovery

The ultimate goal of any science is scientific discovery, and there have been some work that shows that certain discoveries in biology and medicine can be done automatically. In the mid-80s, based on the literature analysis only, Swanson generated several novel hypotheses related to the connection between disease syndromes and chemical substances that were later confirmed experimentally. For example, he found a connection between migraine and magnesium [Swanson, 1988] and between Raynaud's syndrome and fatty acids in fish oil [Swanson, 1986]. After this pioneering work several researchers continued the studies in this direction. The general idea is to produce a graph where nodes represent biological entities and edges represent the relations between the entities. Each relation found in the literature becomes an edge in the graph. If two nodes are not connected directly, but connected indirectly through other nodes, then they are possibly related to each other, but this connection has not been discovered yet. Some works on protein-protein interaction detection can be viewed as knowledge discovery processes since they produce graphs of related proteins and can discover the relations between proteins not explicitly present in literature [Blaschke et al., 1999, Stapley and Benoit, 2000, Jenssen et al., 2001, Cooper, 2003]. Sehgal, Qui, and Srinivasan extend this to the network of topics, where a topic can be any search specification (basically any PubMed query) and is characterized by a set of documents related to the topic [Sehgal et al., 2003]. Two topics are considered related if they share a number of documents or their documents are

assigned similar MeSH terms. Their system is able to replicate some of the Swanson's discoveries.

Burhans and colleagues started a project to create a system that can reason as a human biologist [Burhans et al., 2003]. They represent the existing knowledge in the form of a semantic network that can then be used to induce new hypotheses. However, the process of translating the knowledge in a semantic network is now done mostly by hand, which narrows significantly the applicability of the system in more general settings. Unlike these researchers, Srihari et al. aim at automatic building of a probabilistic network from information extracted from literature with some promise indicated by preliminary experiments [Srihari et al., 2003].

Much of the text-related research in bioinformatics deals with hierarchically organized data. For example, gene function nomenclature has been standardized as the Gene Ontology hierarchy. Keywords describing an article or a group of related articles often form a hierarchy as, for instance, MeSH terms used to annotate Medline articles. Nevertheless, hierarchical text categorization techniques have been rarely in use in this field. Our research is aimed to change this situation and enhance the existing bioinformatics techniques with additional knowledge of hierarchical relationships present among categories (see Chapter 7).

Chapter 4

Hierarchical learning algorithms

In Section 2.3 we have introduced the notion of hierarchical consistency and the hierarchical consistency requirement. We stated that for better understandability and interpretability of classification results, a hierarchical classification system should produce labeling consistent with a given category hierarchy. As has been shown in Section 3.2, there exist two main approaches to hierarchical text categorization: global and local. Local approaches naturally produce consistent classification since a category can be assigned to an instance only if this instance has been already classified into a parent of the category. As we mentioned in Section 3.2.2, local approaches, especially pachinko machine, have been the focus of several previous studies. However, most of these studies worked on a special type of text collections where class hierarchies were designed as trees and all instances belonged to leaf classes. Moreover, these studies utilized the conventional "flat" evaluation measures that do not reward partially correct classification. As a result, the pachinko machine method always classifies instances to the lowest level categories. In this work, we extend pachinko machine to handle cases where class hierarchies are represented as directed acyclic graphs (DAGs) and internal category assignments are accepted and rewarded.

Unlike local approaches, a global hierarchical algorithm has to be specifically designed to produce consistent classification. We introduce one such global approach [Kiritchenko et al., 2005b]. This is a general framework of converting a conventional "flat" learning algorithm into a hierarchical one. In our experiments we used AdaBoost.MH [Schapire and Singer, 1999] as the underlying learning approach. However, any conventional method capable of performing multi-label classification can be used within this framework. We selected AdaBoost.MH because of its robustness and high performance. The main idea

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$
unseen instance $x \in D$
hierarchy $\mathcal{H} = \langle C, \leq \rangle$

push(Queue, Root($\mathcal{H}$))

while(Queue is not empty) do:
  c = pop(Queue)
  if not $Leaf(c, \mathcal{H})$ then
    $\hat{S} = \{(d_i, \hat{C}_i)$: $\exists (d_i, C_i) \in S$, $\forall \hat{c_j} \in \hat{C}_i$ $\exists c_j \in C_i$
      $\hat{c_j} \in Ancestors(c_j) \wedge \hat{c_j} \in Children(c)\}$
    $predictor$ = learn_classifier($\hat{S}$)
    $local\_labels$ = classify_instance($predictor$, x)
    for each $c_i \in local\_labels$ do:
      add($Labels$, c_i)
      push(Queue, c_i)

Return $Labels$.

Figure 4.1: Generalized hierarchical local approach for tree hierarchies.

of the algorithm is to transform an initial (single-label) task into a multi-label task by expanding the label set of each example with the corresponding ancestor labels. By expanding label sets we ensure that intermediate classes contain all examples from their offspring nodes, in other words, that the initial labeling of training data is consistent with a class hierarchy. Then, we apply a regular learning method such as AdaBoost.MH on the multi-label data. Finally, re-labeling of inconsistently classified instances is performed to satisfy the hierarchical consistency requirement. The method is simple and effective and can be applied to any text categorization task with a class hierarchy represented as a DAG.

4.1 Generalized hierarchical local approach

We present an extended version of the local hierarchical pachinko machine approach. This extended version is capable of handling general cases where hierarchies are represented as DAGs and instances can be classified into intermediate categories. We start with a description of the simplified algorithm applicable only to hierarchical trees. Then, we identify the problems that emerge when a class hierarchy is represented as a DAG. We propose possible solutions to these problems and finally, we present the generalized version of the local hierarchical approach.

The simplified version of the algorithm is presented in Figure 4.1. In a hierarchical

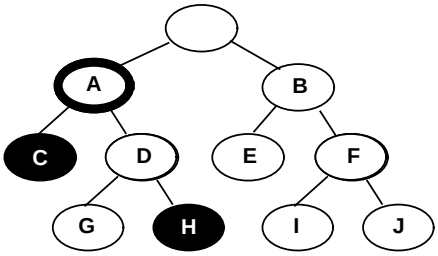


Figure 4.2: Re-labeling of the training data for the first classification subtask, at the root node, in the hierarchical local approach. Solid ellipses represent the real categories of a training instance d_i: $C_i = \{C, H\}$. The ellipse in bold represents the category used as a true label of d_i in the classification subtask: $\hat{C}_i = \{A\}$; $\hat{S} = \{(d_i, \{A\})\}$. Node A is a child category of the root node as well as the ancestor of the instance's real categories C and H.

text categorization task, we are given a training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where each document $d_i \in D$ belongs to a subset of classes (*i.e.* covers several topics) $C_i \subseteq C$. The class set C forms a hierarchy $\mathcal{H} = \langle C, \leq \rangle$. The algorithm first builds a classifier for the categories of the first level of the hierarchy. Any conventional machine learning method can be employed here. However, if a given classification task is multi-label, which is a common case in hierarchical text categorization, the learning method has to be able to perform multi-label classification. This classifier then assigns the most relevant categories to a test instance $x \in D$. If none of the categories seems relevant, the categorization process is stopped. Otherwise, in the next step the learning process is repeated for all categories assigned to the instance classifying it into categories on deeper levels of the hierarchy. Note that in order to learn a classifier for an internal class $c \in C$, we need to transform the original training set S into $\hat{S}$. For this, we pick out the training instances that belong to at least one offspring of category c and change their labels into corresponding categories, children of c (see an example in Figure 4.2). More formally, each instance $(d_i, C_i) \in S$, for which there exists $c_j \in C_i$, an offspring of category c, is replaced with instance $(d_i, \hat{C}_i)$, where $\hat{C}_i$ is composed of all categories, children of c and ancestors of categories from the original label set C_i: $\hat{S} = \{(d_i, \hat{C}_i)\colon \exists (d_i, C_i) \in S, \forall \hat{c}_j \in \hat{C}_i \; \exists c_j \in C_i \; \hat{c}_j \in Ancestors(c_j) \wedge \hat{c}_j \in Children(c)\}$.

This algorithm is basically the standard version of pachinko machine with the exception of the added ability to stop the categorization process at an intermediate level of the hierarchy. This novelty allows us to cope with cases where instances can be assigned to any node of a class hierarchy, including non-leaf nodes. It is also consistent with our new hierarchical evaluation measure, which is introduced in Chapter 5.

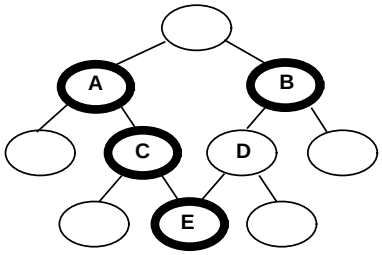


Figure 4.3: Inconsistent labeling as a result of applying the standard hierarchical local approach to a DAG hierarchy. Ellipses in bold represent the categories assigned to an instance by a hierarchical local classifier. Class E has been assigned by a classifier learned for node C. At the same time, its second parent D has not been assigned as a decision of a classifier learned for node B.

The algorithm is guaranteed to produce labeling consistent with a given class hierarchy, because we cannot assign an instance to a class without first assigning it to its parent. In fact, it does not matter in which order we explore the nodes of a hierarchy, breadth-first or depth-first, as long as a node is explored after its parent node. In the case of directed acyclic graphs (DAGs), however, the order becomes important. If we blindly apply the same algorithm to a DAG hierarchy, we can encounter a situation similar to the one shown in Figure 4.3. This figure shows an inconsistent labeling as a result of the wrong classification order. Suppose, we first learn a classifier for the top node of the hierarchy and assign an instance to both categories A and B. Then, we learn a classifier for node A and assign the instance to C. If in the next step we learn a classifier for node C and assign the instance to E, then we can end up with an inconsistent labeling in the case when the classifier in node B does not assign the instance to class D. To keep the consistency requirement fulfilled, we have to impose more restrictions on the classification order: a node can be explored only after *all* its parent nodes have been explored. In addition, we have to keep track of all visited nodes to avoid recurring exploring of nodes that can be reached by different paths. The full version of the generalized hierarchical local approach can be found in Figure 4.4.

4.2 New hierarchical global approach

In this section we present a global hierarchical framework that produces a classifier consistent with a given category hierarchy [Kiritchenko et al., 2005b]. We introduce this framework with a state-of-the-art learning algorithm AdaBoost.MH [Schapire and

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$
unseen instance $x \in D$
hierarchy $\mathcal{H} = \langle C, \leq \rangle$

for each $c \in C$ do:
  done[c] = $false$
  rejected[c] = $false$

push(Queue, Root($\mathcal{H}$))

start: while(Queue is not empty) do:
  c = pop(Queue)

  if done[c] then go to start
  if rejected[c] then go to start

  for each $p \in Parents(c, \mathcal{H})$ do:
    if not done[p] then
      push(Queue, c)
      go to start

  done[c] = $true$

  add($Labels$, c)

  if not $Leaf(c, \mathcal{H})$ then
    $\hat{S} = \{(d_i, \hat{C}_i)$: $\exists (d_i, C_i) \in S$, $\forall \hat{c_j} \in \hat{C}_i$ $\exists c_j \in C_i$
      $\hat{c_j} \in Ancestors(c_j) \wedge \hat{c_j} \in Children(c)\}$
    $predictor$ = learn_classifier($\hat{S}$)
    $local\_labels$ = classify_instance($predictor$, x)

    for each $c_i \in Children(c, \mathcal{H})$ do:
      if $c_i \in local\_labels$ then
        push(Queue, c_i)
      else
        for each $f \in Offspring(c_i, \mathcal{H})$ do:
          rejected[f] = $true$

Return $Labels$.

Figure 4.4: Generalized hierarchical local approach for DAG hierarchies.

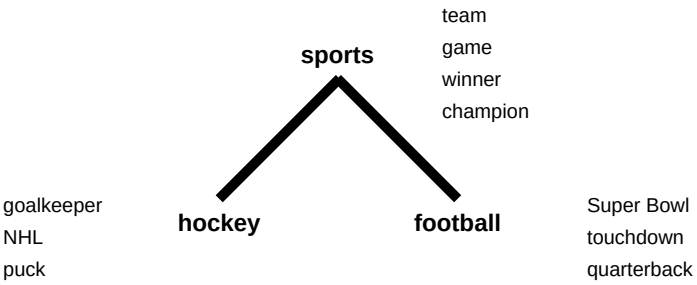


Figure 4.5: Hierarchically shared attributes. The categories "hockey" and "football" have their own specific vocabulary while sharing some common terms that also appear in their parent category "sports".

Singer, 1999] as an underlying learning method. In addition, we propose three novel techniques for selecting thresholds for AdaBoost.MH in truly multi-label settings.

The new hierarchical global approach is simple and effective. Before learning a classification model, we make given (inconsistent) data consistent with a corresponding class hierarchy. For this, we expand the label set of each training example with the corresponding ancestor labels. As a result, in the modified dataset each intermediate category contains training examples originally assigned to this category and examples originally assigned to the descendant nodes of the category in a hierarchical graph. This data modification forces a learning algorithm to focus on high level categories by providing a large number of training examples for those categories. The correct classification of unseen instances into high level categories is very important in hierarchical categorization since high level categories define the most general topics for documents. For example, if we classify a news article about an art exhibition into category "sports" (if "arts" and "sports" are among the top level categories), it would be completely wrong. On the other hand, a mistake made for lower levels, *e.g.* classification of a document on minor hockey into category "professional hockey", would not be so dramatical.

We expect the presented strategy to be successful in the hierarchical settings because a hierarchical structure is typically designed to reflect the semantic closeness of categories. Therefore, we anticipate that related categories share some attributes. In the text categorization context, that means shared vocabulary. For example, categories "hockey" and "football" have their own specific vocabulary, such as "goalkeeper" or "NHL" for "hockey" and "Super Bowl" or "touchdown" for "football" (Figure 4.5). At the same time, these two categories likely share some common terms, such as "team" or "game", that also appear in their parent category "sports". Our method allows a

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$
unseen instance $x \in D$
hierarchy $\mathcal{H} = \langle C, \leq \rangle$

$\hat{S} = \{(d_i, \hat{C}_i)\colon (d_i, C_i) \in S, \hat{C}_i = \{\bigcup_{c_k \in C_i} Ancestors(c_k)\}\}$
$predictor$ = learn_classifier($\hat{S}$)

$Labels$ = classify_instance($predictor$, x)

for each $c_i \in Labels$ do:
  if $\exists c_k : c_k \in Ancestors(c_i) \wedge c_k \notin Labels$ then
    $Labels$ = re-label_consistently(x, $Ancestors(c_i)$)

return $Labels$

Figure 4.6: Hierarchical global approach.

learning algorithm to explore such common attributes in order to improve classification, especially for high level categories.

Overall, the algorithm consists of three steps:

1. Transformation of training data, making them consistent with a given class hierarchy;
2. Application of a regular learning algorithm on the multi-label dataset;
3. Re-labeling of inconsistently classified test instances.

The pseudo-code for the hierarchical global approach is presented in Figure 4.6. In the first step, we replace each example (d_i, C_i), $d_i \in D$, $C_i \subseteq C$, with (d_i, $\hat{C}_i$), where $\hat{C}_i = \{\bigcup_{c_k \in C_i} Ancestors(c_k)\}$. Figure 4.7 shows an example of such replacement. Initially, only categories E and F are assigned to a training instance. This labeling is inconsistent with the class hierarchy since the ancestor categories A, B, C, and D of the initial labels E and F are not assigned to the instance. We correct this inconsistency and add labels A, B, C, and D to the label set of the example.

After that, we apply a regular learning algorithm, in our case AdaBoost.MH, on the modified multi-label dataset. As we mentioned earlier, this is a general framework, which is not restricted to AdaBoost.MH only. Any learning method capable of dealing with multi-label settings can be used here instead of AdaBoost.MH.

Since we train a classifier on consistent data, we expect the classifier to label test instances consistently as well. However, it is not guaranteed. Some of the test instances can end up with inconsistent labels. For example, AdaBoost.MH cannot preserve the consistency since it manipulates the weights for each category independently simulating in a way the "one vs. all" approach. Therefore, a test instance can be classified inconsistently if the confidence score of some class A passes a given threshold while the

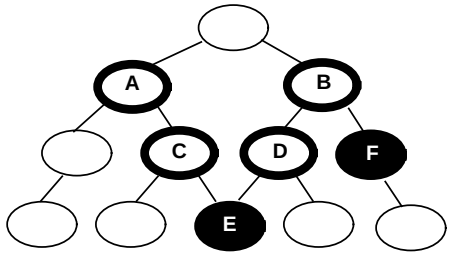


Figure 4.7: Re-labeling of the training data in the hierarchical global approach. The solid ellipses represent the initial categories a training instance belongs to; the ellipses in bold show the categories added to the label set of the instance by the hierarchical algorithm.

confidence score of one of its ancestor classes does not. For such instances we need to do the third post-processing step. At this step we re-label the instances in a consistent manner by considering the confidence in the predictions for class A and all its ancestor classes. One possible procedure here is to calculate the average of these confidences. If the average is greater than a threshold, we label the instance with class A and all its ancestor classes; if the average is lower than the threshold, we do not assign class A to the instance. This procedure acts as a kind of weighted voting. Each ancestor class votes with its own confidence score. Large positive scores would indicate high certainty in the assigning the class, while negative values would vote against this class assignment.

4.2.1 AdaBoost.MH, a boosting algorithm for multi-class multi-label classification

Boosting is a learning technique that combines many simple "weak" hypotheses into one highly accurate predictor. This class of techniques is called ensemble learning and consists of boosting, bagging, cross-validated committees and others. Each of these methods produce several hypotheses $h_1, \ldots, h_L$. Then, the final decision is made by a committee of these hypotheses, typically by weighted voting:

$$h(d) = \sum_i w_i h_i(d).$$

The intuition behind ensemble methods is that the decision of several classifiers can be more accurate than the decision of one of them if they make uncorrelated errors. Like in an old saying, "two heads are better than one". If a classifier makes a mistake on an instance, there is a chance that other classifiers in the committee make a correct decision

in this case, therefore, improving the overall accuracy. This is particularly true when the classifiers make errors independently of each other. In general, if individual predictors are at least slightly better than random and the errors they make are independent, then the voting decision should be more accurate than either of the predictors. However, if the individual classifiers have error rates greater than 50%, then the committee's error will only increase.

More formally, there are at least three reasons why ensemble methods can produce committees more accurate than a single classifier [Dietterich, 1997]. First, a hypothesis space can be so large and/or an available training set can be so small, that there are several plausible hypotheses that satisfy the training data. In this case, an ensemble of these hypotheses is a natural choice for the final classifier. Second, a learning algorithm usually is not capable of exhaustively searching the entire hypothesis space. For example, finding the smallest decision tree consistent with training data is NP-hard, so a decision tree learning algorithm uses some heuristics to guide the search. Therefore, there can be situations where the learning algorithm is unable to find the optimal solution. Since individual classifiers are usually trained on different samples of training data, an ensemble of these classifiers can produce a better solution than a single classifier. Third, in some situations a hypothesis space may not contain the true function. For example, a decision tree learning algorithm cannot produce a decision hyperplane that is not parallel to coordinate axes; it can only approximate such a hyperplane. Individual classifiers trained on different samples of training data will produce different approximations, therefore leading to a better overall approximation as a committee's decision.

The ensemble methods differ in ways that they learn hypotheses and later on combine them. In boosting, the choice of one hypothesis influences the choice of other hypotheses and the weights assigned to them. Such ensembles are called additive models. A very successful boosting algorithm, called AdaBoost, was proposed by Freund and Schapire [Freund and Schapire, 1996] and was improved and extended for the multi-class multi-label case by Schapire and Singer [Schapire and Singer, 1999]. Later, a specialized version of AdaBoost for text categorization was implemented in software BoosTexter [Schapire and Singer, 2000]. We will first describe the multi-class multi-label variant of AdaBoost, called AdaBoost.MH, and then its specific implementation in BoosTexter, which we use in our experiments[1].

As has already been mentioned, AdaBoost.MH [Schapire and Singer, 1999] is designed

[1]BoosTexter software is available free for non-commercial research or educational purposes from http://www.research.att.com/∼schapire/BoosTexter/

for multi-class multi-label problems. Formally, let D denote the domain of instances and C be a finite set of class labels, $|C| = k$. Let $S = ((d_1, C_1), \ldots, (d_m, C_m))$ be a set of training examples, where $d_i \in D$ and $C_i \subseteq C$, *i.e.* each instance can belong to multiple classes. The task is to predict all and only the correct labels for each instance $(d, C_d) \in D \times C$. To achieve this task, AdaBoost.MH aims to minimize the Hamming loss, which is the symmetric difference between the real C_d and predicted $h(d)$ label sets on a given distribution of examples:

$$Hamming\_Loss_P(H) = \frac{1}{k} E_{(d,C_d)\sim P}\{|h(d)\Delta C_d|\}$$

where $H : D \to 2^C$ is a learned hypothesis, P is a given distribution of examples, $E\{f\}$ is the expectation value of a function f, and Δ is symmetric difference.

We can reformulate this multi-class multi-label problem as k binary problems where the i^{th} problem corresponds to correctly predicting whether an instance $(d, C_d) \in D \times C$ belongs to the i^{th} class $\ell \in C$ or not. We can view C as a vector of binary labels ($\ell \in C$ is included in C_d or not) and $h(d)$ as a vector of k binary predictions. Then, the Hamming loss will be the average error rate of h on these k binary problems.

Let's denote

$$C_i[\ell] = \begin{cases} +1, \text{if } \ell \in C_i, \\ -1, \text{if } \ell \notin C_i. \end{cases}$$

for all $C_i \subseteq C$ and $\ell \in C$.

Now, we assume access to a "weak" learning algorithm that takes as input a set of training examples $S = ((d_1, C_1), \ldots, (d_m, C_m))$ and a distribution P over $\{1, \ldots, m\}$. It returns a "weak" hypothesis $h : D \times C \to \Re$. These hypotheses provide not only class predictions, but also self-rated confidences of the predictions. We call the values $h(d, \ell)$ the confidence scores. We interpret the sign of $h(d, \ell)$ as essentially a prediction, i.e positive (negative) values mean that h assigns (does not assign) instance d to class ℓ. The magnitude of $|h(d, \ell)|$ corresponds to the reliability of the prediction. Put another way, we cut off the predictions at zero threshold: all categories with confidences greater than or equal to zero are assigned to an instance while the categories with negative confidences are not. In Section 4.2.2 we present methods of finding more practical thresholds leading to better performance.

Overall, the AdaBoost.MH algorithm works as follows (see Figure 4.8). First, each training example (d_i, C_i) is replaced with k examples $((d_i, \ell), C_i[\ell]), \ell \in C$ and the distribution $P(i, \ell)$ is maintained over examples i and labels ℓ. Initially, the distribution is

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$

Initialize distribution $P_1(i, \ell) = 1/(mk)$.
For $t = 1, \ldots, T$:

- Train weak learner using distribution P_t.
- Get weak hypothesis $h_t : D \times C \to \Re$.
- Choose $\alpha_t \in \Re$.
- Update:

$$P_{t+1}(i, \ell) = \frac{P_t(i, \ell) exp(-\alpha_t C_i[\ell] h_t(d_i, \ell))}{Z_t}$$

where

$$C_i[\ell] = \begin{cases} +1, \text{if } \ell \in C_i, \\ -1, \text{if } \ell \notin C_i, \end{cases}$$

and Z_t is a normalization factor (chosen so that P_{t+1} will be a distribution).

The final hypothesis:

$$H(d, \ell) = \sum_{t=1}^{T} \alpha_t h_t(d, \ell)$$

is used to classify an unseen instance $d \in D$ with threshold $\bar{t} = 0$:

$$(d, \ell) = \begin{cases} true, \text{if } H(d, \ell) \geq 0, \\ false, \text{if } H(d, \ell) < 0. \end{cases}$$

Figure 4.8: AdaBoost.MH [Schapire and Singer, 1999].

uniform. Then, on each iteration t a new "weak" hypothesis $h : D \times C \to \Re$ is learned on a current distribution $P_t(i, \ell)$ over the training examples. After that, the distribution is modified to increase the weight of the incorrectly classified training examples and decrease the weight of the correctly classified examples. As a result, in the next round a "weak" learner is forced to focus on examples that are hardest to classify.

Schapire and Singer [Schapire and Singer, 1999] proved the upper bound on the Hamming loss of the final hypothesis H:

$$Hamming\_Loss(H) \leq \prod_{t=1}^{T} Z_t,$$

where Z_t is the normalization factor computed in round t

$$Z_t = \sum_{i=1}^{m} \sum_{\ell \in C} P_t(i, \ell) exp(-\alpha_t C_i[\ell] h_t(d_i, \ell)).$$

In each round we want to minimize the Hamming loss of the predictions. Thus, we need to choose α_t and design a "weak" learning algorithm in such a way as to minimize Z_t.

AdaBoost.MH is a general-purpose algorithm that can be combined with any "weak" learning method. In practice, however, it is usually employed with decision trees or decision stumps[2]. BoosTexter software [Schapire and Singer, 2000], designed specifically for text categorization, offers one such implementation. In BoosTexter, a "weak" learner is a decision stump. It tests if a term w is present or absent in a document:

$$h(d,\ell) = \begin{cases} q_{0\ell}, \text{if term } w \text{ does not occur in } d, \\ q_{1\ell}, \text{if term } w \text{ occurs in } d, \end{cases}$$

where $q_{j\ell}$ are real numbers representing the confidences of the predictions.

Let's denote X_0 the subset of training documents that do not contain term w and X_1 the subset of training documents that contain term w: $X_0 = \{d \in D : w \text{ does not occur in } d\}$, $X_1 = \{d \in D : w \text{ occurs in } d\}$. It was shown [Schapire and Singer, 1999] that in order to minimize the normalization factor Z_t, we should set $\alpha_t = 1$ and $q_{j\ell} = \frac{1}{2} ln \left(\frac{W_{+1}^{j\ell}}{W_{-1}^{j\ell}} \right)$. The weights $W_{+1}^{j\ell}$ ($W_{-1}^{j\ell}$) represent the total weight (with respect to the current distribution P_t) of documents in subset X_j that are (are not) labeled with category ℓ:

$$W_b^{j\ell} = \sum_{d_i \in X_j, C_i[\ell]=b} P_t(i,\ell), j \in \{0,1\}, b \in \{+1,-1\}.$$

With such settings, we have

$$Z_t = 2 \sum_{j \in \{0,1\}} \sum_{\ell \in C} \sqrt{W_{+1}^{j\ell} W_{-1}^{j\ell}}.$$

At each iteration we choose the term w that minimizes Z_t.

Since the values $W_{+1}^{j\ell}$ and $W_{-1}^{j\ell}$ can be in practice very small or even 0, BoosTexter applies a smoothing technique:

$$q_{j\ell} = \frac{1}{2} ln \left(\frac{W_{+1}^{j\ell} + \epsilon}{W_{-1}^{j\ell} + \epsilon} \right),$$

where ϵ is some small positive value.

AdaBoost has been extensively studied by many researchers in the past 10 years. In

[2] A decision stump is a one-level decision tree.

many experiments AdaBoost showed excellent results significantly improving the performance of single learners (*e.g.* decision trees) as well as beating other ensemble methods (*e.g.* bagging). Schapire and colleagues explain this phenomena from the point of view of PAC learning theory [Schapire et al., 1998]. They provide the boundary on the generalization error of an ensemble showing that the success of AdaBoost is due to its ability to increase the margins on the training data[3]. However, this bound is not tight enough to fully explain the impressive performance of the algorithm. Friedman, Hastie, and Tibshirani give another possible explanation from the statistical point of view [Friedman et al., 2000]. They show that AdaBoost can be interpreted as a stage-wise estimation procedure for fitting an additive logistic regression model $F(x) = \sum_{i=1}^{M} k_m f_m(x)$. Therefore, unlike other (randomized) ensemble methods that only reduce variance to compensate the instability of base learners, AdaBoost reduces both bias and variance by jointly fitting individual hypotheses in an additive predictor.

Moreover, AdaBoost has a very important property: resistance to overfitting. Often, when a predictor becomes very complex in its attempt to decrease the training error, the classification model overfits the data, and the test error increases. In AdaBoost, on the other hand, as more "weak" hypotheses are added, the test error decreases and then levels off. This is possibly due to the fact that with each successive step the changes to the overall function become smaller since only training points along the decision boundary (the points that were incorrectly classified at the previous step) are involved in the learning of a new base classifier. Also, AdaBoost fits M functions sequentially, step-wise, therefore, reducing the variance of an ensemble comparing to the variance of jointly fitted M functions [Friedman et al., 2000].

4.2.2 Finding high-quality thresholds for multi-label AdaBoost.MH

The classification decisions of AdaBoost.MH are made based on the final hypothesis: $H(d, \ell) = \sum_{t=1}^{T} \alpha_t h_t(d, \ell)$, which is a real-valued function. For single-label classification, the classification decision is simply the top-ranked class, the class with the highest confidence score. In a multi-label case, however, we have to select a threshold to cut off class labels for a given instance. One such possible threshold is *zero*: any positive confidence score indicates that the class should be assigned to an instance, any negative score indi-

[3]The margin is the quantity $C_i[\ell]H(d, \ell)$ that characterizes the amount by which d is correctly classified.

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$
hierarchy $\mathcal{H} = \langle C, \leq \rangle$

predictor = Learn_AdaBoost(S)
Confidences[ ] = Classify(*predictor*, S)

thresholds = {Confidences[d_j][c_i] | j = 1,. . . ,m, i = 1,. . . ,k}
Sort(thresholds)

$f\_max = 0$
For each $t \in$ thresholds
    f = Calculate_F_1(t, S, Confidences[ ])
    if ($f > f\_max$)
        $f\_max = f$; $t\_max = t$

Return t_max.

Figure 4.9: Finding best *single* threshold for AdaBoost.MH.

cates that the class should not be assigned to an instance. However, we are often able to find better thresholds with procedures described below.

The first one is a simple straight-forward procedure for selecting the best *single* threshold for given data. The algorithm is presented in Figure 4.9. First, we train AdaBoost.MH on an available training set S and get the confidence predictions on the same set S. Then, we put the confidences in a decreasing order and try them one by one as possible thresholds. For each such threshold we compute the evaluation measure that we try to optimize, *e.g.* the F_1 measure, on the training data and pick the threshold that gives the best result. Since it is well-known that optimizing the learning parameters on training data often leads to overfitting, we also run a similar procedure with a hold-out validation set. For this, we learn a classifier on the training data, but get the confidences and compute F_1 measure on the hold-out set.

Since a hierarchical text categorization task usually involves a large number of classes, it is unlikely that the best *single* threshold well represents all the best individual class thresholds. Therefore, we test a slightly more complex procedure that finds the best individual thresholds for every *subtree* of the top node of a class hierarchy[4]. So, we start with zero thresholds for all classes in a hierarchy: $T[c_i] = 0, c_i \in C$ (Figure 4.10). We learn a classifier and get the confidence values for all training instances and all categories. Then, we separately find the best single thresholds for each subtree in turn. Specifically, for each subtree $Subtree_r$ we try every confidence value assigned to categories from this subtree as a possible threshold t: $T[c_i] = t, c_i \in Subtree_r$. We calculate the F_1 measure

[4]This procedure is defined only for class hierarchies represented as trees. In a general case, where a class hierarchy is a DAG, this procedure is not applicable.

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$
hierarchy $\mathcal{H} = \langle C, \leq \rangle$

For each $c_i \in C$
$T[c_i] = 0$

$predictor$ = Learn_AdaBoost(S)
Confidences[ ] = Classify($predictor$, S)

For each $c_r \in Children(Root(\mathcal{H}))$
$Subtree_r = \{c_i \in C | c_i \in Offspring(c_r)\}$
thresholds = {Confidences[d_j][c_i] | j = 1,. . . ,m, i = 1,. . . ,k, $c_i \in Subtree_r$}
Sort(thresholds)

$f\_max = 0$
For each $t \in$ thresholds
For each $c_i \in Subtree_r$
$T[c_i] = t$
f = Calculate_F_1(T[ ], S, Confidences[ ])
if ($f > f\_max$)
$f\_max = f$; $t\_max = t$

For each $c_i \in Subtree_r$
$T[c_i] = t\_max$

Return T[ ].

Figure 4.10: Finding best *subtree* thresholds for AdaBoost.MH.

on the whole training set using thresholds $T[c_i], c_i \in C$. Finally, we pick the value that gives the best result and update $T[c_i]$ for all classes from the subtree: $c_i \in Subtree_r$.

In addition, we compile a procedure that finds the best individual *class* thresholds (Figure 4.11). This procedure provides the most flexible set of thresholds. As in the previous routine, we start with zero thresholds for all classes in a hierarchy: $T[c_i] = 0, c_i \in C$. We learn a classifier and get the confidence values for all training instances and all categories. Then, we separately find the best single thresholds for each class in turn. Specifically, for each class c_r we test every confidence value assigned to the class as a possible threshold t: $T[c_r] = t$. We calculate the F_1 measure on the whole training set using thresholds $T[c_i], c_i \in C$. Finally, we pick the value that gives the best result and update the class threshold $T[c_r]$.

We conduct a series of experiments to investigate the goodness of the proposed thresholding techniques for multi-label AdaBoost.MH. The first set of experiments is designed to study the performance of different thresholding strategies in the simplest setting, single-label non-hierarchical. We compare the proposed thresholding techniques with the best strategy on single-label tasks, which is to assign the *single most confident pre-*

Given: training set $S = ((d_1, C_1), \ldots, (d_m, C_m))$, where $d_i \in D, C_i \subseteq C$
hierarchy $\mathcal{H} = \langle C, \leq \rangle$

For each $c_i \in C$
$T[c_i] = 0$

predictor = Learn_AdaBoost(S)
Confidences[ ] = Classify(*predictor*, S)

For each $c_r \in C$
thresholds = {Confidences[d_j][c_r] | j = 1,. . . ,m}
Sort(thresholds)

$f\_max = 0$
For each $t \in$ thresholds
$T[c_r] = t$
f = Calculate_F_1(T[ ], S, Confidences[ ])
if ($f > f\_max$)
$f\_max = f$; $t\_max = t$

$T[c_r] = t\_max$

Return T[ ].

Figure 4.11: Finding best individual *class* thresholds for AdaBoost.MH.

diction. For these experiments, we use 26 datasets from the UCI repository [Hettich et al., 1998] described in Table 4.1[5]. In addition, we experiment with the "flatten" version of our synthetic data (ignoring hierarchical relations)[6]. We evaluate the performance with standard F-measure. For each UCI dataset, 10 times 10-fold cross-validation experiments are performed; 100 runs are performed on randomly generated synthetic data (with parameters set to the following: number of levels is 3, out-degree is 2).

The results are presented in Figure 4.12 and Tables 4.2 and 4.3. Figure 4.12 shows the performance of the 4 thresholding strategies: *single (most confident) prediction*, *zero threshold*, best *single threshold*, and best individual *class thresholds*[7]. The plots on the left show the performance of these algorithms on one of the UCI datasets, Autos, and the plots on the right demonstrate the performance on the synthetic data. Evidently, the best *single* and *class* thresholding methods have a tendency to overfit the data. To explain this phenomenon, we study the behavior of the best found thresholds over the boosting iterations (bottom row of Fig. 4.12). At the beginning of the learning process, AdaBoost.MH consistently underestimates its confidence in class prediction, confidence scores tend to be negative, and so are the best thresholds. While the number

[5]The UCI datasets were chosen to contain at least 5 attributes and at least 100 examples.
[6]For description of the synthetic data see Section 6.1.
[7]The best *subtree thresholding* strategy is not applicable to non-hierarchical data.

dataset	**number of attributes**	**number of categories**	**number of examples**
anneal	38	6	898
audiology	69	24	226
autos	26	7	205
breast-cancer	9	2	286
colic	28	2	368
credit-a	15	2	690
credit-g	20	2	1000
diabetes	8	2	768
glass	9	7	214
heart-c	13	5	303
heart-h	13	5	294
heart-statlog	13	2	270
hepatitis	19	2	155
hypothyroid	29	4	3772
ionosphere	34	2	351
kr-vs-kp	36	2	3196
lymph	18	4	148
mushroom	22	2	8124
primary-tumor	17	22	339
segment	19	7	2310
sick	29	2	3772
sonar	60	2	208
splice	61	3	3190
vehicle	18	4	846
vowel	13	11	990
waveform-5000	40	3	5000

Table 4.1: UCI datasets used in the experiments.

of iterations increases, the best thresholds increase as well coming towards zero. After a while, AdaBoost.MH becomes very confident in the prediction on training data; as a result, the best thresholds turn into "large" positive values and can overfit the data. To avoid this effect, we can use a hold-out set instead of training data to search for best thresholds. Fig. 4.12 (on the right) shows the performance of the *single* and *class* thresholding on hold-out set on the synthetic data. Alternatively, if obtaining additional data is a problem (as it is the case in many real-life situations), a simple smoothing technique also works very well: instead of the best threshold, we use the average of the best and the closest smaller confidence score. For this, we sort all confidence scores that we get on training data in the decreasing order (t_1, t_2, ..., t_n). Then, we find t_k that gives the best value for the F-measure (the best threshold). With the number of boosting iterations, the separation between the positive and negative confidence scores tends to

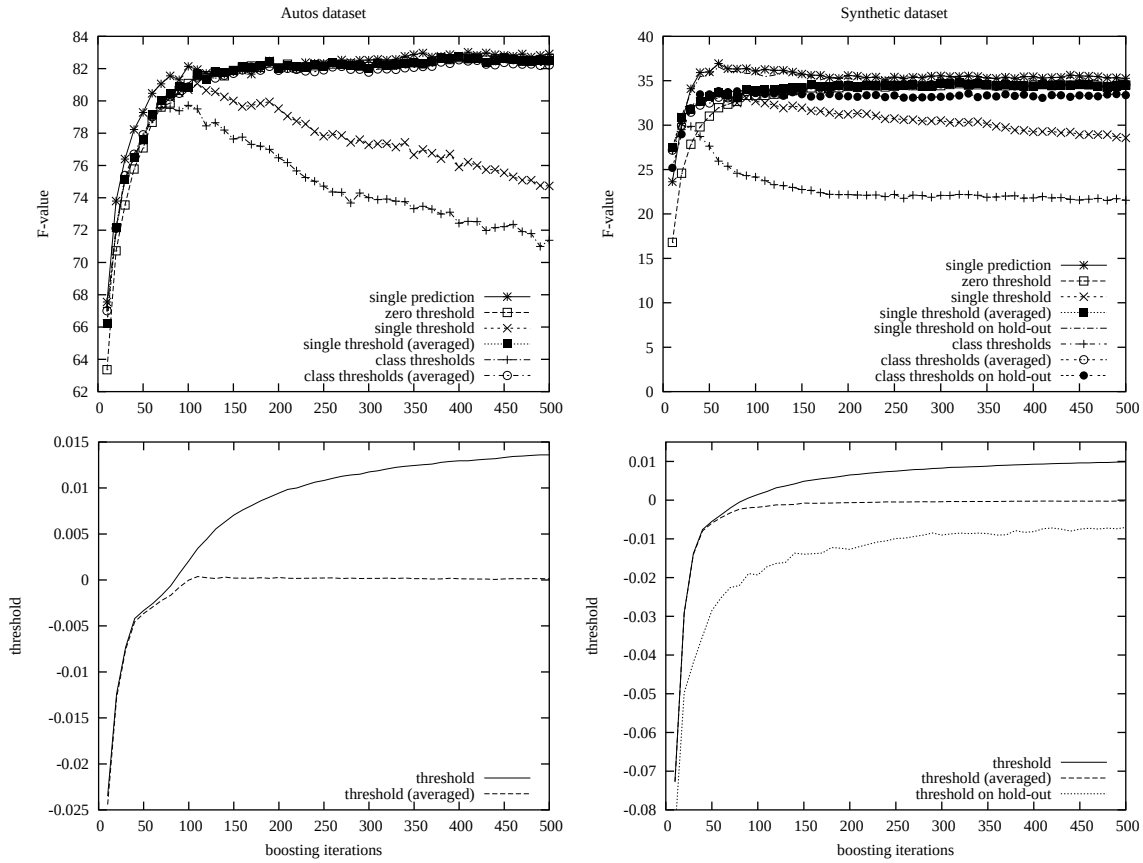


Figure 4.12: AdaBoost.MH with different thresholding strategies on single-label non-hierarchical data.

grow. Thus, the best score t_k is usually a large positive value while the next score t_{k+1} is a negative value. By taking an average $(t_k + t_{k+1})/2$, we keep our threshold close to zero, therefore, avoiding overfitting. The results (bottom row of plots in Fig. 4.12) confirm our hypothesis showing that the averaged thresholds indeed stay very close to zero. This technique has a dramatic effect on the performance. Both *single* and *class averaged* thresholding methods produce results significantly better than non-averaged techniques reaching the performance of thresholding on hold-out data or even better. Another interesting observation is that all proposed thresholding techniques (smoothed and non-smoothed) considerably outperform the thresholding at *zero* at the beginning of the boosting process, when the number of iterations is small and the confidence values are mostly negative. After a while, all techniques, except non-smoothed ones, show similar performance and get close to the best possible line, the performance of the *single most confident prediction*.

dataset	zero threshold	single threshold		class thresholds	
		non-averaged	averaged	non-averaged	averaged
anneal	98.49	98.64	98.67+++	98.75+++	98.80+++
audiology	77.40	77.68	77.98+++	77.47	77.62
autos	72.64	73.61	73.96	73.90+++	73.75
breast-cancer	70.23	71.81+++	71.80+++	72.29+++	71.91+++
colic	82.15	82.27	82.32	81.88	82.09
credit-a	85.45	85.73+++	85.52	85.58	85.34
credit-g	73.67	75.26+++	75.39+++	75.25+++	75.34+++
diabetes	75.24	76.90+++	76.88+++	76.58+++	76.59+++
glass	68.60	69.84+++	68.85+++	68.24	68.18
heart-c	82.93	82.81	82.32	81.76—	81.62
heart-h	82.00	82.35	81.89	81.42	80.99
heart-statlog	80.67	80.92	81.38	80.09	80.61
hepatitis	82.88	82.89	82.4	82.53	81.67
hypothyroid	99.54	99.53	99.55	99.54	99.56
ionosphere	92.90	92.77	93.05	92.14—	92.47
kr-vs-kp	95.17	95.60+++	95.59+++	96.81+++	96.76+++
lymph	82.65	82.68	83.03	82.39	83.20
mushroom	99.95	99.98+++	99.97+++	100.00+++	100.00+++
primary-tumor	47.06	47.83	47.59	46.21	45.90
segment	94.04	94.29+++	94.25+++	94.39+++	94.33+++
sick	97.35	97.39	97.35	97.32	97.34
sonar	80.06	79.38	79.99	78.96—	79.80
splice	93.00	93.14+++	93.01	93.16+++	93.06
vehicle	68.25	71.94+++	71.69+++	71.78+++	72.03+++
vowel	70.20	72.51+++	72.59+++	73.79+++	73.15+++
waveform-5000	81.80	82.40+++	82.55+++	82.54+++	82.65+++
total +++		12	12	12	10
total —		0	0	3	0

Table 4.2: The performance (F-measure) of AdaBoost.MH with different thresholding strategies on UCI data after 25 iterations. "+++"/"—" indicate that AdaBoost.MH with a corresponding thresholding algorithm performs better/worse than AdaBoost.MH with zero thresholding and the differences are statistically significant according to the paired t-test with 99% confidence.

		single threshold		**class thresholds**	
dataset	**zero threshold**	**non-averaged**	**averaged**	**non-averaged**	**averaged**
anneal	99.58	99.50	99.59	98.99—	99.55
audiology	82.62	79.56—	82.12	75.51—	81.69—
autos	82.32	79.68—	81.86	76.69—	81.70
breast-cancer	69.05	70.31+++	70.50+++	70.86+++	71.09+++
colic	81.71	81.65	81.74	81.49	81.72
credit-a	84.12	84.22	83.99	84.05	83.99
credit-g	73.69	74.56+++	74.95+++	74.63+++	74.98+++
diabetes	74.81	75.53+++	75.19+++	75.43+++	74.84
glass	71.87	70.23—	71.48	66.66—	71.25
heart-c	78.43	78.42	79.02	78.02	78.48
heart-h	78.94	79.11	79.22	78.06—	78.68
heart-statlog	78.37	78.23	78.14	78.10	78.17
hepatitis	83.18	78.33—	83.70	77.97—	83.64
hypothyroid	99.50	99.45	99.48	99.34—	99.48
ionosphere	93.10	87.71—	93.05	86.98—	92.93
kr-vs-kp	96.82	96.83	96.87	97.01+++	97.00+++
lymph	82.14	81.82	81.95	80.94	81.76
mushroom	100.00	100.00	100.00	100.00	100.00
primary-tumor	45.65	46.44	45.86	45.54	45.37
segment	97.01	97.04	97.01	96.77—	97.00
sick	97.64	97.66	97.69	97.66	97.68
sonar	83.28	60.01—	82.88	59.45—	82.88
splice	94.49	94.46	94.50	94.46	94.51
vehicle	75.71	76.34+++	76.45+++	76.07	76.48+++
vowel	84.90	85.11	85.30+++	84.23—	85.11
waveform-5000	83.89	84.23+++	84.44+++	84.23+++	84.48+++
total +++		5	6	5	5
total —		6	0	11	1

Table 4.3: The performance (F-measure) of AdaBoost.MH with different thresholding strategies on UCI data after 200 iterations. "+++"/"—" indicate that AdaBoost.MH with a corresponding thresholding algorithm performs better/worse than AdaBoost.MH with zero thresholding and the differences are statistically significant according to the paired t-test with 99% confidence.

Similar results are obtained on most of the UCI datasets. Table 4.2 shows the results of the *zero*, *single*, and *class* thresholding algorithms as well as the *averaging* version of the *single* and *class* thresholding algorithms on 26 UCI datasets after 25 boosting iterations. Both averaged and non-averaged versions of the *single* thresholding strategy significantly outperform *zero* thresholding on 12 datasets. On all other data the three algorithms show similar results. Most of the UCI datasets are quite small and simple, so apparently, 25 iterations is enough for AdaBoost.MH to learn confident models. As a result, we cannot gain much from choosing a threshold other than zero. The *class* thresholding strategies, both non-averaged and averaged, demonstrate a slightly worse performance outperforming *zero* thresholding on 12 and 10 datasets respectively. However, they were able to produce significantly better results than *single* thresholding on 3 datasets, namely "kr-vs-kp", "mushroom", and "vowel". Table 4.3 shows the performance of the same algorithms after 200 boosting iterations. On 5 (the hardest) datasets we still see significant improvement of *single* thresholding over *zero* thresholding. However, it is more than twice as small as it was after 25 iterations. Also, on 6 datasets there is a significant loss in F-measure. At the same time, the smoothed version does not show any loss while demonstrating improvement on 6 datasets. Similar situation holds for *class* thresholding and its smoothed version, yet the performance of the *class* thresholding is getting quite worse than the performance of *single* thresholding.

In the second set of experiments, we compare the proposed thresholding techniques in the hierarchical settings. We used our synthetic data as well as the 20 newsgroups and Reuters-21578 datasets (for description of these datasets see Section 6.1). AdaBoost.MH is replaced with its hierarchical global version, and the performance is evaluated with our hierarchical hF-measure (see Chapter 5). In both settings, hierarchical and non-hierarchical, the algorithms behave very similarly. Figure 4.13 shows the performance of the proposed algorithms in the hierarchical settings on the synthetic data[8]. Again the non-smoothed versions of the algorithms start to overfit the data after about 100 iterations when the best thresholds outgrow zero. One noticeable difference in these plots is that the *single most confident prediction* does not work well in the hierarchical settings since most of the examples have more than one class label.

We also run similar experiments on real data, namely 20 newsgroups and Reuters-21578. Figure 4.14 shows how much gain we can get on real data with *single* thresholding

[8]The *subtree* thresholding methods are not shown to keep the plots readable. Those methods show very similar performance to the corresponding *single* thresholding methods since the class hierarchy of the synthetic dataset has only two subtrees.

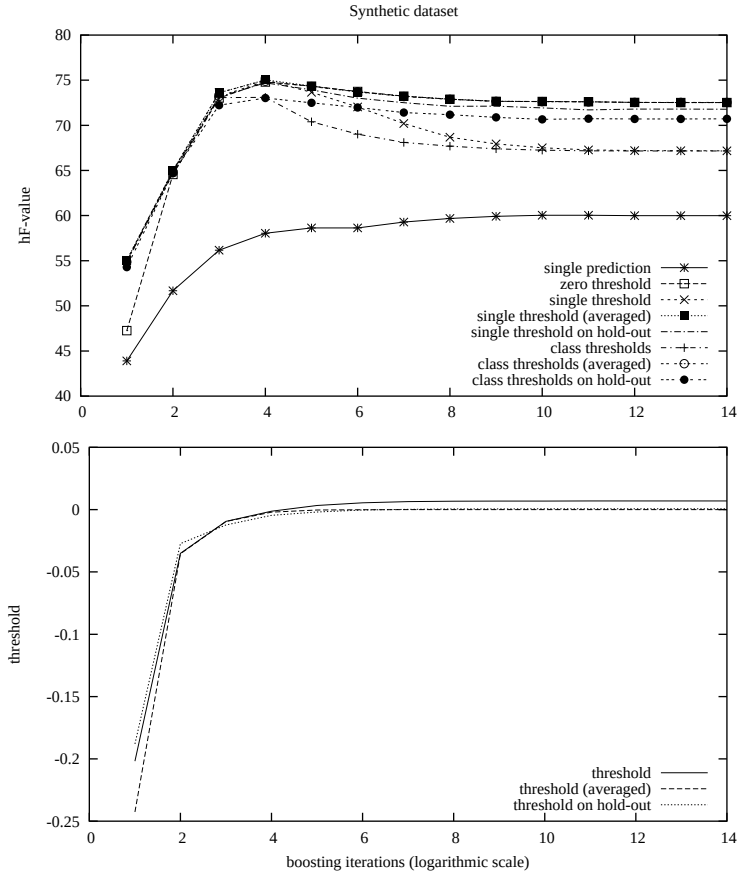


Figure 4.13: Hierarchical AdaBoost.MH with different thresholding strategies on multi-label hierarchical data.

as oppose to *zero* thresholding. On real, large-scale data with large class hierarchies and thousands of relevant attributes, AdaBoost.MH needs several hundreds or even a few thousands of iterations to learn high confidence models. This means that with time constraints when the number of iterations is limited, our thresholding techniques can be very beneficial. In our experiments, on both datasets the *single* thresholding method outperforms *zero* thresholding even after 500 iterations (the differences are statistically significant with 99% confidence). The gain is the largest at the beginning of boosting learning. As the number of iterations grows, the difference in performances of two algorithms decreases. At the same time, the rate of performance improvement for AdaBoost.MH also decreases with the number of iterations. As a result, the savings from the *single* thresholding technique in terms of boosting iterations increase as boosting progresses. Overall, AdaBoost.MH with *single* thresholding requires up to 30% fewer

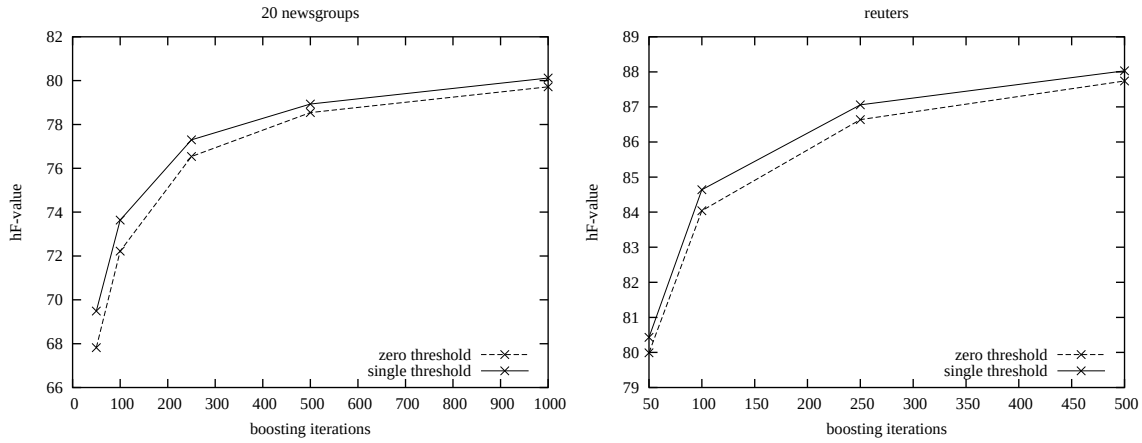


Figure 4.14: Hierarchical AdaBoost.MH with the best *single threshold* and with *zero threshold* on "20 newsgroups" and "reuters" data.

iterations than AdaBoost.MH with *zero* threshold does to reach the same level of accurateness.

These results show that the proposed thresholding techniques (with smoothing or using hold-out data) can effectively replace the *single prediction* method when needed, i.e. in multi-label settings. Among the three strategies, *single*, *subtree* and *class* thresholding, there is no clear winner, so *single* thresholding seems the most attractive because of its simplicity. Also, this technique often produces better results than the *zero* thresholding method, especially when the number of boosting iterations is small. Therefore, we use the *averaged single* thresholding approach in the following experiments since we do not always have enough data for hold-out sets.

4.3 Other global hierarchical approaches

We have experimented with a few other hierarchical algorithms: hierarchical decision trees, ECOC, and cost-sensitive learning. However, the preliminary results have not shown any promise; thus, we have decided not to pursue these topics any further. We present a brief description of the undertaken approaches for completeness (for more details see Appendix A).

Hierarchical decision trees

The objective of this approach was to modify the entropy/gain ratio splitting criteria to incorporate the hierarchical information. In short, we have tried to favor splits with categories that are close in a hierarchical graph, in a way simulating the hierarchical local approach. In particular, we modified the entropy formula to give more weight to sibling categories. We also tested the hierarchical evaluation measure as a new splitting criterion. Finally, we experimented with Gini index and different misclassification costs. Overall, we were not able to get a significant improvement in F-measure. The resulting decision trees were usually larger, which probably led to overfitting.

ECOC

Error-Correcting Output Codes (ECOC) proved to be a robust scheme for multi-class categorization. Generally, the codes are generated independently to get maximal row and column separation[9]. To enable the hierarchical information, we added bits to represent the class dependencies and to allow more/less separation between sibling and non-sibling categories. The additional bits resulted in smaller column separation and therefore, in less accurate predictions.

Cost-sensitive learning

Cost-sensitive learning concerns the classification tasks where the costs of misclassification errors are not uniform[10]. For example, the cost of deleting an important email as spam is usually much larger than the cost of letting a spam message through. A few algorithms have been proposed to deal with such problems. We experimented with two such algorithms: C5.0 and MetaCost.

C5.0 is a commercial release of the classical decision tree learning algorithm C4.5 [Quinlan, 1993]. It has been designed to handle large datasets faster and more efficiently. It also has additional functionalities, such as incorporated boosting, variable misclassification costs, sampling, and others. The variable misclassification costs component allows C5.0 to construct classifiers that minimize expected misclassification costs rather than error rates.

[9]Separation is defined as the number of bits in which the codes differ (Hamming distance).

[10]In general, cost-sensitive learning also deals with the costs of tests, such as the costs of observing features. In this work, we do not take these costs into account.

MetaCost [Domingos, 1999] is a method for making any learning algorithm cost-sensitive. This is achieved by applying a cost-minimizing procedure around a base learner. The algorithm uses bagging to estimate the class probabilities on training examples, relabels the training examples with the estimated optimal class that minimizes misclassification costs and applies a base learner on the relabeled training set.

Class hierarchies can naturally be converted to cost matrices: the larger the distance between two classes in a hierarchy, the bigger their misclassification cost should be in a cost matrix. To test if such transformation can be useful for hierarchical text categorization, we ran both algorithms on the 20 newsgroups dataset with varied costs. The performance of C5.0 with costs was just slightly better than its performance with uniform costs. We suspect that this is due to poorly calibrated prediction probabilities produced by the decision tree learning algorithm. At the same time, MetaCost produced results much worse than those of the base non-cost-sensitive learner.

4.4 Summary

In this chapter we have presented two hierarchical learning approaches that are capable of performing hierarchically consistent classification. The first approach is a generalized version of the classical hierarchical local algorithm, pachinko machine. We extend the local approach to the general case of DAG class hierarchies and possible internal class assignments. The second algorithm is a novel hierarchical global framework that builds a single classifier for all categories in a hierarchy. Since in present research both algorithms are applied with AdaBoost.MH as a base learner, we have described this state-of-the-art boosting technique and introduced several novel methods for selecting high-quality thresholds for AdaBoost.MH in the multi-label setting.

Chapter 5

Hierarchical evaluation measure

In this chapter we discuss performance evaluation measures for hierarchical classification and introduce natural, desired properties that these measures ought to satisfy. We define a novel hierarchical evaluation measure, and show that, unlike the conventional "flat" as well as the existing hierarchical measures, the new measure satisfies the desired properties. It is also simple, requires no parameter tuning, and has much discriminating power.

5.1 Motivation

As we have shown in the overview of previous work (Section 3.3), most researchers evaluate hierarchical classification systems based on standard "flat" measures: accuracy/error and precision/recall. However, these measures are not suitable for hierarchical categorization since they do not differentiate between different kinds of misclassification errors (see Figure 5.1). Since most of the class hierarchies in text categorization reflect the semantic closeness of categories, misclassification to a sibling or parent node of a correct category intuitively seems preferable to misclassification to a distant node. Therefore, we need a new measure that would be more discriminating, allowing us to give credit for less severe misclassification errors.

The alternative measures for hierarchical categorization proposed so far are not flexible enough. The category similarity based measure proposed by Sun and Lim [Sun and Lim, 2001] considers two categories to be similar if they share the vocabulary. In real-life hierarchies, however, categories with similar vocabulary can be placed quite far away from each other and misclassification between them can be considered as a severe error.

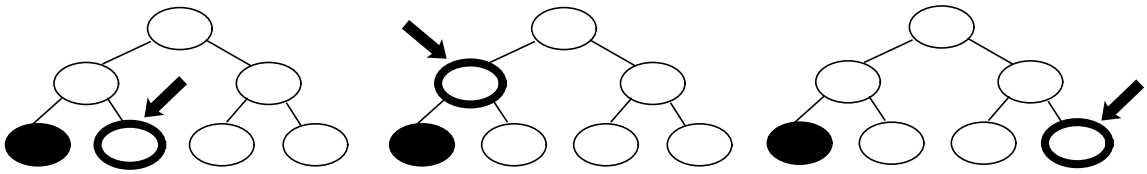


Figure 5.1: Weaknesses of the conventional non-hierarchical measures. The solid ellipse represents the real category of a test instance; the ellipse in bold (with an arrow pointing to it) represents the category assigned to the instance by a classifier. Non-hierarchical evaluation metrics give the errors equal weights, so the classification performance is the same in all three cases. (Strictly speaking, exactly the same values would be obtained only for microaveraged metrics; however, for each category the contribution of the errors would be the same for all non-hierarchical measures.)

For example, topics "genetic algorithms (a computer science discipline)" and "genetics (a bioscience discipline)" share many keywords such as chromosome, mutation, crossover, population (see Figure 5.2), yet misclassification between these two categories is undesirable. In addition, documents can have several topics. For example, chess is a common application for AI algorithms, such as heuristic search. Therefore, there exist many documents that can be classified as "chess-related" and as "AI-related" (for example, articles about the match between Gary Kasparov and the IBM computer Deep Blue). These two categories share some documents and some vocabulary. However, misclassifying a document about a chess match at the world championship as an AI topic would be completely wrong. The same is true for the class similarity measure proposed by Wang and colleagues [Wang et al., 2001]. Categories that share many documents are not necessarily close in the hierarchy. In different applications the closeness of categories is determined differently. Thus, if we want to have a general measure suitable for all applications, we have to base our measure only on the information available to us: a given hierarchy. We argue that the only reliable relations between categories are the relations given by the hierarchy.

Distance-based measures give a relatively good approximation of category relations in a hierarchy, but they are not discriminating enough. For example, misclassification into a sibling and into a grandparent of a correct category would have a distance error of 2, and we cannot separate these two cases (see Figure 5.3). Moreover, a distance-based measure does not change with depth. Misclassification into a sibling category of a top level node and misclassification into a sibling of the node 10-level deep are considered the same type of error (distance of 2). However, an error at the 10th level seems a lot

The **evolution** of the probabilities of **genetic** identity within and between the loci of a multigene family dispersed among multiple **chromosomes** is investigated. Unbiased **gene** conversion, equal **crossing over**, **random genetic** drift, and **mutation** to **new** alleles are incorporated. **Generations** are discrete and nonoverlapping; the diploid, monoecious **population** mates at **random**. The linkage map is arbitrary, but the same for every **chromosome**; the dependence of the probabilities of identity on the location on each **chromosome** is formulated exactly. The greatest of the rates of **gene** conversion, **random** drift, and **mutation** is {epsilon} << 1.

T. Nagylaki. Gene Conversion, Linkage, and the Evolution of Repeated Genes Dispersed Among Multiple Chromosomes. *Genetics*, 126: 261-276, 1990.

In general, **genetic** algorithms start with an initial set of **random** solutions, called **population**. Each individual in the **population** is called a **chromosome**, representing a solution to the problem at hand. A **chromosome** is a string of symbols (usually represented by a binary bit string). The **chromosomes evolve** through successive iterations, called **generations**. During each **generation**, the **chromosomes** are evaluated using some measures of fitness. To create the next **generation**, **new chromosomes** are formed by three essential operations: selection, **crossover**, and **mutation**.

G. Kim and S. Kim. Feature Selection Using Genetic Algorithms for Handwritten Character Recognition. In the *Proceedings of the 7th International Workshop on Frontiers in Handwriting Recognition*, 2000.

Figure 5.2: Weaknesses of the category similarity based measure. The extracts of scientific articles show vocabulary shared by two diverse topics: genetic algorithms (a computer science discipline) and genetics (a bioscience discipline). Shared keywords are shown in bold.

less harmful than an error at the top level.

5.1.1 Desired properties of a hierarchical evaluation measure

To express the desired properties of a hierarchical evaluation measure, we formulate the following requirements [Kiritchenko et al., 2005b]:

1. Partially correct classification.

The measure gives credit to partially correct classification (Figure 5.4(1)), *e.g.* misclassification into node A when the correct category is G should be penalized less than misclassification into node B since A is in the same subgraph as G and B is not. With this property we want the measure to be able to separate the cases of completely wrong classification, *i.e.* when the classification is wrong even at the most general level, and partially correct classification, *i.e.* when classification is correct at least at the top level.

2. Error discrimination by distance.

The measure penalizes distant errors more heavily:

a) the measure gives higher score for correctly classifying one level down compared with staying at the parent node (Figure 5.4(2a)), *e.g.* classification into node D is better than classification into its parent A since D is closer to the correct category G;

b) the measure gives lower score for incorrectly classifying one level down compared

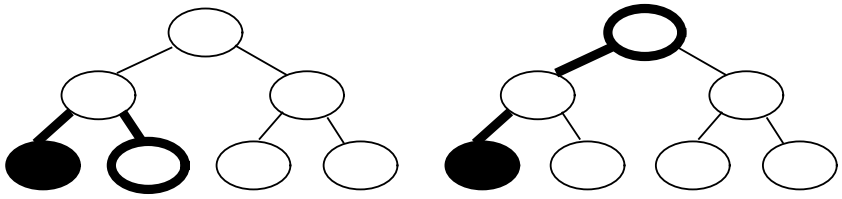


Figure 5.3: Weaknesses of distance-based hierarchical measures. The solid ellipse represents the real category of a test instance; the ellipse in bold represents the category assigned to the instance by a classifier; edges in bold represent the shortest path between the real and assigned categories or, in other words, the distance-based error. In both cases, the distance-based error equals to 2.

with staying at the parent node (Figure 5.4(2b)), *e.g.* classification into node C is worse than classification into its parent A since C is farther away from G.

Since most of the first hierarchical measures were distance-based, it is obvious that this property is desired for an ideal hierarchical measure.

3. Error discrimination by depth.

The measure penalizes errors at higher levels of a hierarchy more heavily (Figure 5.4(3)), *e.g.* misclassification into node H when the correct category is its sibling G is less severe than misclassification into node D when the correct category is its sibling C. This property supports our intuition that errors made at deeper levels are less severe.

Formally, let us denote $HM(c_1|c_2)$ the hierarchical evaluation score of classifying an instance $d \in D$ into class $c_1 \in C$ when the correct class is $c_2 \in C$ in a given tree hierarchy $\mathcal{H} = \langle C, \leq \rangle$. This score represents the level of correctness of classification results, *i.e.* the better classification, the higher evaluation score HM. Then, the following properties should be observed.

1. **Partially correct classification:** for any instance $(d, c_0) \in D \times C$, if $Ancestors(c_1) \cap Ancestors(c_0) \neq \oslash$ and $Ancestors(c_2) \cap Ancestors(c_0) = \oslash$, then $HM(c_1|c_0) > HM(c_2|c_0)$;[1]

2. **Error discrimination by distance:**

a) for any instance $(d, c_0) \in D \times C$, if $c_1 = Parent(c_2)$ and $distance(c_1, c_0) > distance(c_2, c_0)$, then $HM(c_1|c_0) < HM(c_2|c_0)$;

b) for any instance $(d, c_0) \in D \times C$, if $c_1 = Parent(c_2)$ and $distance(c_1, c_0) < distance(c_2, c_0)$, then $HM(c_1|c_0) > HM(c_2|c_0)$;

3. **Error discrimination by depth:** for any instances (d_1, c_1), $(d_2, c_2) \in D \times C$, if

[1]Let us remind that ancestor sets $Ancestors(c_i)$, $c_i \in C$ do not contain the root of the class hierarchy.

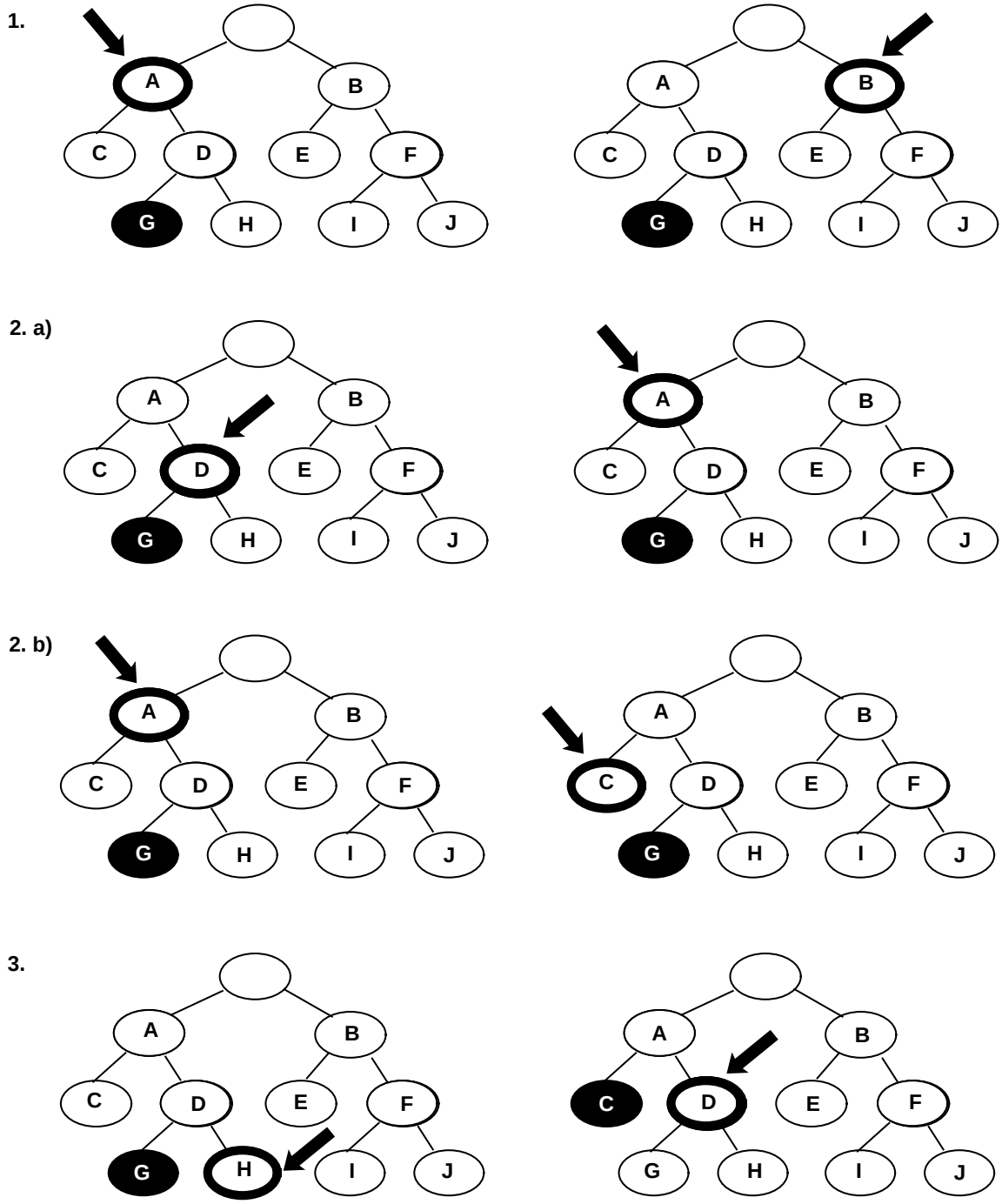


Figure 5.4: The desired properties of a hierarchical evaluation measure. The solid ellipse represents the real category of a test instance; the ellipse in bold (with an arrow pointing to it) represents the category assigned to the instance by a classifier. A good hierarchical measure should give more credit to the situations on the left comparing to the corresponding situations on the right.

Evaluation Measure	**Desired properties of a hierarchical evaluation measure**		
	Partially correct classification	**Error discrimination by distance**	**Error discrimination by depth**
conventional "flat" measures	-	-	-
distance-based measures	-	+	-
weighted distance [Blockeel et al., 2002]	-	+	+
weighted penalty [Blockeel et al., 2002] (semantic similarity measure [Lord et al., 2003])	+	-	+
class similarity measure [Wang et al., 2001]	-	-	-
category similarity measure [Sun and Lim, 2001]	-	+	+
measure proposed by Ipeirotis et al. (2001)	-	+	-

Table 5.1: Characteristics of the "flat" and existing hierarchical evaluation measures.

$distance(c_1, c_1') = distance(c_2, c_2')$, $level(c_1) = level(c_2) + \Delta$, $level(c_1') = level(c_2') + \Delta$, $\Delta > 0$, $c_1 \neq c_1'$, $c_2 \neq c_2'$, and $level(x)$ is the length of the unique path from the root to node x, then $HM(c_1'|c_1) > HM(c_2'|c_2)$.

The listed requirements are natural properties that any hierarchical evaluation measure should possess. These requirements cover straightforward situations where there is an intuitive behavior of a measure. We ensure that hierarchical evaluation measures behave consistently at least in these basic situations.

Clearly, all previous measures do not satisfy at least one of the properties (Table 5.1).

- Conventional "flat" measures, such as standard accuracy or precision/recall, consider all kinds of misclassification errors to be equally bad; thus, they do not satisfy

any of the three requirements.

- Distance-based hierarchical measures calculate the distance between correct and predicted categories in a hierarchical tree. They satisfy the second principle, but not the first and not the third. In addition, they are not easily extendable to DAG hierarchies (where multiple paths between two categories can exist) and multi-label tasks.

- The weighted distance measure, where all hierarchy edges are given weights decreasing exponentially with depth [Blockeel et al., 2002], solves the problem with the third property, but other drawbacks of distance-based measures remain. Also, this weighted distance measure requires a set of predefined weights (possibly application-dependent) which are not obvious to get. It is even more problematic with the cost-sensitive distance measure [Cai and Hofmann, 2004], where two sets of weights $cost_1(v)$ and $cost_2(v)$ are required for each node v.

- The weighted penalty measure [Blockeel et al., 2002] (semantic similarity measure [Lord et al., 2003]) is calculated as the weight of the deepest common ancestor of correct and predicted categories, where deeper nodes have smaller weights. This measure satisfies the first and third properties, but not the second one. Since many pairs of categories would share the same ancestor nodes and, therefore, have the same weighted penalty, this measure has little discriminating power.

- Class similarity measure [Wang et al., 2001] considers the similarity of the sets of documents belonging to the categories. It heavily depends on a given corpus and, in general, does not satisfy all three requirements. For example, misclassification into a sibling category which does not share any documents with a correct category would be given zero credit.

- Category similarity measure [Sun and Lim, 2001] is based on the content of documents comprising the categories. It also heavily depends on a given corpus. However, in general, it should satisfy the second and third properties, but may violate the first one.

- The measure proposed by Ipeirotis et al. [Ipeirotis et al., 2001] considers the overlap in subtrees induced by a correct and predicted category sets. It satisfies only the second property. This measure gives credit only to misclassification into an

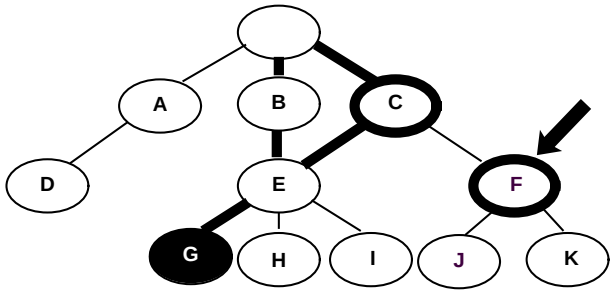


Figure 5.5: New hierarchical measure. The solid ellipse G represents the real category of an instance; the ellipse in bold F (with an arrow pointing to it) represents the category assigned to the instance by a classifier. All nodes on the path from the root to the assigned category (*i.e.* the ancestors of the assigned category) are shown in bold since they are also assigned to the instance by our measure. The path from the root to the real category - the correct path - is shown in bold. $hP = |\{C\}| / |\{C, F\}| = 1/2$, $hR = |\{C\}| / |\{B, C, E, G\}| = 1/4$.

ancestor or a descendant category of a correct category, but gives zero credit for misclassifying into a sibling of a correct category.

5.2 New hierarchical evaluation measure

We propose a new measure for evaluating hierarchical text categorization systems that is based solely on a given hierarchy, gives credits for partially correct classification and is very discriminating [Kiritchenko et al., 2005b]. Our new measure is the pair precision and recall with the following addition: each example belongs/classified not only to a class, but also to all ancestors of the class in a hierarchical graph, except the root. We exclude the root of the tree, since all examples belong to the root by default. We call the new measure hP (hierarchical precision) and hR (hierarchical recall).

Formally, in the multi-label settings, for any instance (d_i, C_i), $d \in D$, $C_i \subseteq C$ classified into subset $C'_i \subseteq C$ we extend sets C_i and C'_i with the corresponding ancestor labels: $\hat{C}_i = \{\bigcup_{c_k \in C_i} Ancestors(c_k)\}$, $\hat{C}'_i = \{\bigcup_{c_l \in C'_i} Ancestors(c_l)\}$. Then, we calculate (microaveraged) hP and hR as follows:

$$hP = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}'_i|} \qquad hR = \frac{\sum_i |\hat{C}_i \cap \hat{C}'_i|}{\sum_i |\hat{C}_i|}$$

For example, suppose an instance is classified into class F while it really belongs to class G in a sample DAG class hierarchy shown in Figure 5.5. To calculate our

hierarchical measure, we extend the set of real classes $C_i = \{G\}$ with all ancestors of class G: $\hat{C}_i = \{B, C, E, G\}$. We also extend the set of predicted classes $C'_i = \{F\}$ with all ancestors of class F: $\hat{C}'_i = \{C, F\}$. So, class C is the only correctly assigned label from the extended set: $|\hat{C}_i \cap \hat{C}'_i| = 1$. There are $|\hat{C}'_i| = 2$ assigned labels and $|\hat{C}_i| = 4$ real classes. Therefore, we get $hP = \frac{|\hat{C}_i \cap \hat{C}'_i|}{|\hat{C}'_i|} = \frac{1}{2}$ and $hR = \frac{|\hat{C}_i \cap \hat{C}'_i|}{|\hat{C}_i|} = \frac{1}{4}$. [2]

The new measure is close in spirit to the one proposed by Ipeirotis, Gravano, and Sahami [Ipeirotis et al., 2001]. In their work, for a given instance all categories in a subtree rooted in a correct category are also considered correct and the overlap in subtrees induced by the correct and predicted category sets is measured. The principle difference between the two measures is that instead of counting the descendants we count the ancestors. We believe that our method is more intuitive since in general, category relationships, "is-a" and "part-of", are transitive. In other words, it is legitimate to say that an entity belonging to a category also belongs to the category's ancestors. However, an entity belonging to a category normally belongs only to some of the category's descendants. For example, a document about cellular processes in general cannot be classified under the cell growth category because it also describes other processes in a cell. At the same time, a document about cell growth is about cellular processes.

To summarize the two parts of the measure, precision and recall, into one value, we compute the hierarchical F-value:

$$hF_\beta = \frac{(\beta^2 + 1) \cdot hP \cdot hR}{(\beta^2 \cdot hP + hR)}, \beta \in [0, +\infty)$$

Parameter β is chosen for a task at hand and represents the relevant importance of one part of the measure over the other in a given application. The hierarchical evaluation measure makes natural decisions in terms of precision and recall separately. Then, in the combined measure the preference can be given to either part depending on the application. The final decision is left to a user: if one is interested in recall, β should be set to

[2] It may seem reasonable to count an ancestor label several times if a few initial class labels share this ancestor. We, however, follow the policy to add each ancestor only once so that the set of (true or predicted) class labels remains a set after the addition of ancestor labels. For example, the set of nodes $\{G, F\}$ in Figure 5.5 would be extended to set $\{B, C, E, G, F\}$ even though ancestor label C is shared by both nodes G and F. Such an extension perfectly reflects the semantics of transitive class relations where the extended set represents all the categories an instance is semantically associated with. It also corresponds to the typical classifier behavior when a class label is assigned only once. For example, suppose nodes G and F are true categories for some instance, but a classifier predicts category C. A classifier is capable to return label C as its prediction only once and should not be penalized for not returning it the second time. Therefore, the label should be counted as a true category only once.

a value greater than 1; if one is interested in precision, β should be less than 1[3].

Our new hierarchical measure has already been adopted in the community. Joslyn and colleagues used it in their work on automatic ontological function annotation [Joslyn et al., 2005]. They have extended our measure to more precisely quantify the contribution of each category in multi-label classification. Instead of calculating hierarchical precision and recall for complete sets of predicted and true categories of an instance (as we do), they exploit the pairwise calculations. For each predicted category of an instance, the maximal pairwise hierarchical precision is calculated over all true categories, then summed over all predicted categories to obtain the total precision:

$$hP = \sum_{q \in C'_i} max_{p \in C_i} \frac{|Ancestors(p) \cap Ancestors(q)|}{|Ancestors(q)|}$$

Similarly, for each true category, the maximal pairwise hierarchical recall is calculated over all predicted categories, then summed over all true categories to obtain total recall:

$$hR = \sum_{p \in C_i} max_{q \in C'_i} \frac{|Ancestors(p) \cap Ancestors(q)|}{|Ancestors(p)|}$$

5.3 Probabilistic interpretation of precision and recall

In the previous section, we have given and explained the formulas for calculating the hierarchical measures of precision and recall. Now, we present the natural interpretation of these notions from the probabilistic point of view.

Precision can be defined as the probability that an instance classified into category c_i indeed belongs to this category, and recall is the probability that an instance that belongs to category c_i will be classified into this category. Hierarchical precision and recall follow the same definitions if we extend the sets of correct and predicted categories with their corresponding ancestor classes. Now, we can view precision and recall as parameters in some probabilistic model that generates our observed data. The formulas for precision and recall

[3]In all reported experiments we use microaveraged hF_1 hierarchical measure, giving precision and recall equal weights.

$$p = \frac{TP}{TP+FP} \qquad r = \frac{TP}{TP+FN}$$

are estimates of these unknown parameters.

Goutte and Gaussier [Goutte and Gaussier, 2005] present one such probabilistic model. For each category $c_i \in C$, the experimental outcome can be summarized in four numbers: TP (true positives), FP (false positives), TN (true negatives), and FN (false negatives) (Table 5.2).

	classified into c_i	**not classified into c_i**
belong to c_i	TP	FN
do not belong to c_i	FP	TN

Table 5.2: Contingency matrix defining TP (true positives), FP (false positives), TN (true negatives), and FN (false negatives).

We can assume that observed TP, FP, FN, and TN counts follow a multinomial distribution with parameters $n = TP + FP + FN + TN$ and $\pi = (\pi_{TP}, \pi_{FP}, \pi_{FN}, \pi_{TN})$:

$$P(D = (TP, FP, FN, TN)) = \frac{n!}{TP!FP!FN!TN!}\pi_{TP}^{TP}\pi_{FP}^{FP}\pi_{FN}^{FN}\pi_{TN}^{TN},$$

where $\pi_{TP} + \pi_{FP} + \pi_{FN} + \pi_{TN} = 1$.

From this, Goutte and Gaussier project to a lower dimensional space. Using the properties of multinomial distributions, they show that observed TP counts follow a binomial distribution with parameters $TP + FP$ and p (precision). Similarly, observed TP counts follow a binomial distribution with parameters $TP + FN$ and r (recall). So, we can write for precision

$$P(D|p) = \frac{(TP + FP)!}{TP!FP!}p^{TP}(1-p)^{FP}$$

and for recall

$$P(D|r) = \frac{(TP + FN)!}{TP!FN!}r^{TP}(1-r)^{FN}.$$

Now, we want to find the most probable estimates for the parameters p and r given the observed data D:

$$\hat{p} = argmax_p\ P(p|D) = argmax_p\ \frac{P(D|p)P(p)}{P(D)} = argmax_p\ P(D|p)P(p).$$

If we assume that all values for p are equally probable a priori, then we can derive

the maximum likelihood estimate for parameter p:

$$\hat{p}_{ML} = argmax_p\ P(D|p) = argmax_p\ \frac{(TP+FP)!}{TP!FP!}p^{TP}(1-p)^{FP}.$$

Taking logarithm of the last expression

$$\hat{p}_{ML} = argmax_p\ ln(P(D|p)) = ln\left(\frac{(TP+FP)!}{TP!FP!}\right) + TP * ln(p) + FP * ln(1-p)$$

and then the first derivative

$$\frac{\partial log(P(D|p))}{\partial p} = \frac{TP}{p} - \frac{FP}{1-p},$$

we deduce the formula for the maximum likelihood estimate of p:

$$\hat{p}_{ML} = \frac{TP}{TP+FP}.$$

Similarly, the maximum likelihood estimate for recall r is

$$\hat{r}_{ML} = argmax_r\ P(D|r) = \frac{TP}{TP+FN}.$$

We can see that the usual formulas for precision and recall (and with class set extensions for hierarchical precision and recall) are the maximum likelihood estimates of these notions.

5.4 Properties of the new hierarchical measure

5.4.1 Satisfying all requirements for a hierarchical evaluation measure

Theorem 1. *The new hierarchical measure hF satisfies all three requirements for hierarchical evaluation measures listed above.*

Proof. **Requirement 1 (Partially correct classification):** for any instance $(d, c_0) \in D \times C$, if $Ancestors(c_1) \cap Ancestors(c_0) \neq \oslash$ and $Ancestors(c_2) \cap Ancestors(c_0) = \oslash$, then $HM(c_1|c_0) > HM(c_2|c_0)$.

To calculate hF, we first extend labels c_0, c_1, and c_2 with their ancestor labels: $\hat{C}_0 = Ancestors(c_0)$, $\hat{C}_1 = Ancestors(c_1)$, $\hat{C}_2 = Ancestors(c_2)$. Since $Ancestors(c_1) \cap Ancestors(c_0) \neq \oslash$, $hP(c_1|c_0) = \frac{|\hat{C}_1 \cap \hat{C}_0|}{|\hat{C}_1|} > 0$. Similarly, $hR(c_1|c_0) > 0$ and $hF(c_1|c_0) > 0$. On the other hand, $Ancestors(c_2) \cap Ancestors(c_0) = \oslash$, which means that $hP(c_2|c_0) = 0$, $hR(c_2|c_0) = 0$, and $hF(c_2|c_0) = 0$. Therefore, $hF(c_1|c_0) > hF(c_2|c_0)$.

Requirement 2 (Error discrimination by distance):

Part 1: for any instance $(d, c_0) \in D \times C$, if $c_1 = Parent(c_2)$ and $distance(c_1, c_0) > distance(c_2, c_0)$, then $HM(c_1|c_0) < HM(c_2|c_0)$.

In a hierarchical tree,

$$distance(x, y) = |(Ancestors(x) \cup Ancestors(y)) - (Ancestors(x) \cap Ancestors(y))|.$$

Since it is given that $distance(c_1, c_0) > distance(c_2, c_0)$ and $c_1 = Parent(c_2)$, then it follows that $|Ancestors(c_1) \cap Ancestors(c_0)| < |Ancestors(c_2) \cap Ancestors(c_0)|$. As a result, $hP(c_1|c_0) < hP(c_2|c_0)$, $hR(c_1|c_0) < hR(c_2|c_0)$, and $hF(c_1|c_0) < hF(c_2|c_0)$.

Part 2: for any instance $(d, c_0) \in D \times C$, if $c_1 = Parent(c_2)$ and $distance(c_1, c_0) < distance(c_2, c_0)$, then $HM(c_1|c_0) > HM(c_2|c_0)$.

Given that $distance(c_1, c_0) < distance(c_2, c_0)$ and $c_1 = Parent(c_2)$, we get that $|Ancestors(c_1)| < |Ancestors(c_2)|$ and $|Ancestors(c_1) \cap Ancestors(c_0)| = |Ancestors(c_2) \cap Ancestors(c_0)|$. As a result, $hP(c_1|c_0) > hP(c_2|c_0)$, $hR(c_1|c_0) = hR(c_2|c_0)$, and $hF(c_1|c_0) > hF(c_2|c_0)$.

Requirement 3 (Error discrimination by depth): for any instances (d_1, c_1), $(d_2, c_2) \in D \times C$, if $distance(c_1, c_1') = distance(c_2, c_2')$, $level(c_1) = level(c_2) + \Delta$, $level(c_1') = level(c_2') + \Delta$, $\Delta > 0$, $c_1 \neq c_1'$, and $c_2 \neq c_2'$, then $HM(c_1'|c_1) > HM(c_2'|c_2)$.

In a hierarchical tree, $level(x)$ can be defined as the number of ancestor categories of x: $level(x) = |Ancestors(x)|$. Since $level(c_1) = level(c_2) + \Delta$, we get $|Ancestors(c_1)| = |Ancestors(c_2)| + \Delta$. Similarly, $|Ancestors(c_1')| = |Ancestors(c_2')| + \Delta$. However, it is given that $distance(c_1, c_1') = distance(c_2, c_2')$. This means that $|Ancestors(c_1) \cap Ancestors(c_1')| = |Ancestors(c_2) \cap Ancestors(c_2')| + \Delta$. Thus, $hP(c_1'|c_1) = \frac{|Ancestors(c_1) \cap Ancestors(c_1')|}{|Ancestors(c_1')|} = \frac{|Ancestors(c_2) \cap Ancestors(c_2')| + \Delta}{|Ancestors(c_2')| + \Delta} > hP(c_2'|c_2)$. Similarly, $hR(c_1'|c_1) > hR(c_2'|c_2)$. As a result, $hF(c_1'|c_1) > hF(c_2'|c_2)$. □

5.4.2 Simplicity

The new measure is very simple and easy to compute. Unlike some of the previous measures, *e.g.* the weighted distance or category similarity measures, it is based solely on a given hierarchy, so neither a set of weights nor any parameter tuning is required. However, if an application at hand calls for a different treatment for nodes in a class hierarchy and the set of weights for all nodes is given, our new measure can easily incorporate these weights by counting each node with its specified weight instead of the uniform weight of 1.

5.4.3 Generality

Many previous measures, *e.g.* distance-based measures, were designed only to handle tree hierarchies. The new hierarchical measure is already formulated for a general case of multi-label classification with a DAG class hierarchy.

Moreover, our new measure can be efficiently employed in other applications. Whenever a task has its target entities organized hierarchically and the hierarchical relations are transitive, this task can be evaluated using our hierarchical measure. For example, some dictionaries arrange word senses in "is-a" hierarchies [Resnik and Yarowsky, 1997]. Therefore, word sense disambiguation systems can be compared using hierarchical precision and recall.

5.4.4 Consistency and discriminancy

We have also investigated the consistency and discriminancy properties of our new hierarchical measure in comparison with standard measures, such as non-hierarchical F-value. For this, we follow the definitions introduced by Huang and Ling [Huang and Ling, 2005].

Definition (Consistency). *For two measures f, g on domain Ψ, f, g are (strictly) consistent if there exist no $a, b \in \Psi$, such that $f(a) > f(b)$ and $g(a) < g(b)$.*

The consistency property means that if we have two classifiers A and B and A is more accurate in terms of non-hierarchical measures, then the hierarchical measure agrees, and A is better than B in terms of hF as well.

Evidently, our new measure is not strictly consistent with regular non-hierarchical measures. Figure 5.6 shows an example where our measure gives inconsistent result with standard measures. However, it seems like for most classifiers the consistency property

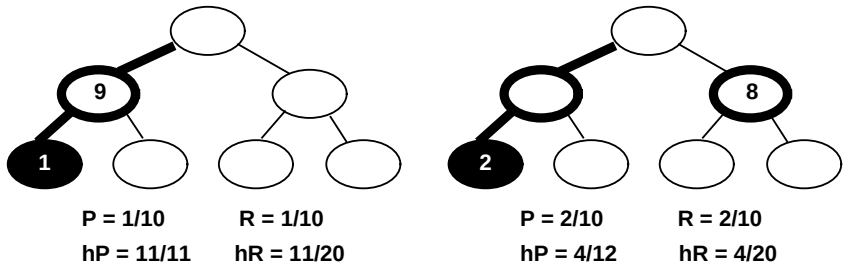


Figure 5.6: An example of inconsistency of the new hierarchical measure with the conventional non-hierarchical measure. The numbers in the nodes show the number of examples classified into a particular node. The correct category for all 10 examples is shown as a solid ellipse. The non-hierarchical measure favors the classifier on the right since it classifies more test instances correctly. The hierarchical measure favors the classifier on the left since it puts more instances on the correct path. (The values are microaveraged.)

holds; in other words, the new hierarchical measure is *statistically* consistent with non-hierarchical ones.

Definition (Statistical Consistency). *For two measures f, g on domain Ψ, let $R = \{(a,b)|a,b \in \Psi, f(a) > f(b), g(a) > g(b)\}$, $S = \{(a,b)|a,b \in \Psi, f(a) > f(b), g(a) < g(b)\}$. The degree of consistency of f and g is $C = \frac{|R|}{|R|+|S|}$. The measure f is statistically consistent with g if and only if $C > 0.5$.*

For the discriminancy property, we again follow the definition introduced by Huang and Ling [Huang and Ling, 2005].

Definition (Discriminancy). *For two measures f, g on domain Ψ, f is (strictly) more discriminating than g if there exist $a, b \in \Psi$ such that $f(a) > f(b)$ and $g(a) = g(b)$, and there exist no $a, b \in \Psi$ such that $g(a) > g(b)$ and $f(a) = f(b)$.*

The discriminancy property implies that if non-hierarchical measures cannot tell apart the performances of classifiers A and B, our measure is more discriminating and prefers one over the other.

Unfortunately, the new measure is not strictly more discriminating than regular non-hierarchical measures. Figure 5.7 shows an example where standard measures are more discriminating than the hierarchical measure. However, the hierarchical measure has a larger range of values; therefore, it can potentially discriminate more pairs of classifiers (see Figure 5.8). This suggests that the new measure is *statistically* more discriminating than standard measures.

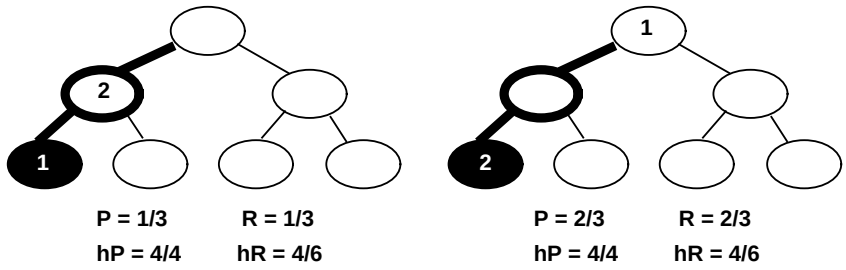


Figure 5.7: An example of the conventional non-hierarchical measure being more discriminating than the new hierarchical measure. The numbers in the nodes show the number of examples classified into a particular node. The correct category for all 3 examples is shown as a solid ellipse. The non-hierarchical measure favors the classifier on the right since it classifies more test instances correctly. The hierarchical measure gives the same values to both classifiers. The root category is not counted because it is a default category for all test instances. (The values are microaveraged.)

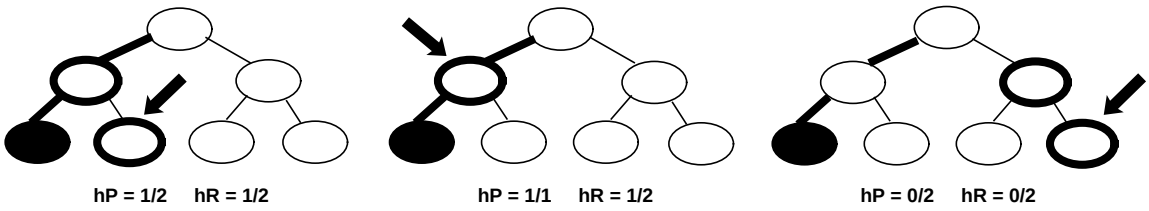


Figure 5.8: An example of the new hierarchical measure being more discriminating than the conventional non-hierarchical measure. The solid ellipse represents the real category of a test instance; the ellipse in bold with an arrow pointing to it represents the category assigned to the instance by a classifier. For one non-hierarchical value we show three different hierarchical values. (The values are microaveraged.)

Definition (Statistical Discriminancy). *For two measures f, g on domain Ψ, let $P = \{(a,b)|a,b \in \Psi, f(a) > f(b), g(a) = g(b)\}$, $Q = \{(a,b)|a,b \in \Psi, g(a) > g(b), f(a) = f(b)\}$. The degree of discriminancy for f over g is $D = \frac{|P|}{|Q|}$. The measure f is statistically more discriminating than g if and only if $D > 1$.*

We believe that the new hierarchical measure is superior to the "flat" measures in that it is statistically consistent with the "flat" measures, but statistically more discriminating. To check if our intuition is indeed correct, we have run a series of tests to compare the new and standard measures on a large number of settings. It is infeasible to do the exhaustive comparison since the number of possible hierarchical topologies and class distributions is infinite. Even for a given hierarchy and a class distribution, the exhaustive comparison of all pairs of classification results is computationally heavy.

hierarchy size		**degree of consistency**	**degree of**
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	74.22 ± 0.01596	183.69
2	3	67.81 ± 0.01705	546.79
2	4	63.69 ± 0.01755	1572.17
2	5	60.67 ± 0.01782	5462.65
3	2	72.82 ± 0.01623	210.21
3	3	65.13 ± 0.01739	1105.49
3	4	59.54 ± 0.01791	10198.22
3	5	56.30 ± 0.01810	92330.57
4	2	71.97 ± 0.01639	280.22
4	3	62.03 ± 0.01771	3137.72
4	4	57.11 ± 0.01806	41922.02
5	2	69.92 ± 0.01673	381.32
5	3	60.56 ± 0.01783	6727.27

Table 5.3: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for uniform class distribution, 100 examples per class, and random classifiers.

hierarchy size		**degree of consistency**	**degree of**
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	74.88 ± 0.01583	1596.79
2	3	68.29 ± 0.01698	4776.00
2	4	64.80 ± 0.01743	22009.67
2	5	62.06 ± 0.01771	38584.20
3	2	73.10 ± 0.01618	2321.54
3	3	66.23 ± 0.01726	8969.53
3	4	61.62 ± 0.01774	13167.65
3	5	57.66 ± 0.01803	152072.67
4	2	72.53 ± 0.01629	1655.33
4	3	64.47 ± 0.01746	6429.06
4	4	58.67 ± 0.01797	36289.75
5	2	70.81 ± 0.01659	1785.63
5	3	62.36 ± 0.01768	9372.69

Table 5.4: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for uniform class distribution, 1000 examples per class, and random classifiers.

The alternative to the exhaustive comparison is a Monte Carlo approach: for a given hierarchical topology and a class distribution pick randomly a pair of classification results and do the comparison. Many random experiments on average should yield a good approximation to the results of the exhaustive approach.

So, to compare the new hierarchical and standard "flat" F-measures for consistency and discriminancy, we adopt the Monte Carlo approach. For given parameters of a class hierarchy, the number of levels and the out-degree, we build a balanced tree hierarchical structure. For initial experiments we assume the uniform class distribution (for all internal and leaf categories). Also, we do not restrict the set of tested classification results to a particular type, so each example is assigned randomly with the uniform distribution to some category from the hierarchical tree. The number of examples is set to 100 per class unless specified otherwise. We generate 10,000 random classification results, calculate the "flat" and the hierarchical F-measure, and then compare these evaluations for all pairs of classification results.

Table 5.3 shows the degree of consistency and the degree of discriminancy for a number of class hierarchies of different sizes. The degree of consistency is above the required threshold of 50% for all parameter settings. On small hierarchies, this value gets over 70%. For larger hierarchies, the consistency drops yet still remaining above the threshold. Moreover, the reduction rate is getting smaller with depth, so it is reasonable to expect the consistency property to hold for hierarchies at least 3-4 levels deeper and a few degrees "wider" than tested. The degree of discriminancy is far above the required threshold of 1. It reaches several hundreds on small hierarchies to several thousands and even tens of thousand on larger class structures.

In the next set of experiments, we increase the amount of data, setting the number of examples to 1000 per class (Table 5.4). Extra examples considerably extend the variety of possible values for the hierarchical hF-measure resulting in a tremendous increase in the degree of discriminancy. At the same time, consistency of the two measures also amplifies by a few percent.

Since uniform class distributions rarely present in real-world applications, we also experiment with imbalanced data. In particular, we set the a priori probability of one of the top level categories to be 5 or 10 times higher than the probability of any other category in the hierarchy. Table 5.5 shows the results for 5:1 skewness, and Table 5.6 shows the corresponding numbers for 10:1 skewness. The introduction of class imbalances at the top level has an effect of slight decrease in consistency and a general increase in discriminancy. The larger the class imbalance is, the greater differences in both consistency

hierarchy size		degree of consistency	degree of
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	69.48 ± 0.01680	268.42
2	3	64.70 ± 0.01744	584.82
2	4	61.80 ± 0.01773	1537.89
2	5	59.03 ± 0.01794	5536.70
3	2	68.57 ± 0.01694	307.96
3	3	62.63 ± 0.01765	1169.47
3	4	59.01 ± 0.01795	9432.81
3	5	55.41 ± 0.01814	78002.34
4	2	67.30 ± 0.01712	347.87
4	3	60.74 ± 0.01782	3042.48
4	4	57.01 ± 0.01806	42253.45
5	2	66.62 ± 0.01721	452.04
5	3	60.12 ± 0.01787	6881.80

Table 5.5: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for imbalanced class distribution (5:1), 100 examples per class, and random classifiers.

hierarchy size		degree of consistency	degree of
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	69.06 ± 0.01687	504.74
2	3	63.19 ± 0.01760	856.59
2	4	59.85 ± 0.01789	1727.50
2	5	57.93 ± 0.01801	4652.63
3	2	66.70 ± 0.01720	467.71
3	3	60.86 ± 0.01781	1279.15
3	4	58.09 ± 0.01800	8635.36
3	5	55.35 ± 0.01814	70237.61
4	2	65.18 ± 0.01738	477.48
4	3	59.43 ± 0.01792	2983.78
4	4	56.46 ± 0.01809	50809.21
5	2	63.99 ± 0.01752	542.14
5	3	58.33 ± 0.01799	7167.04

Table 5.6: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for imbalanced class distribution (10:1), 100 examples per class, and random classifiers.

hierarchy size		**degree of consistency**	**degree of**
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	78.02 ± 0.01511	644.76
2	3	71.74 ± 0.01643	1554.11
2	4	66.98 ± 0.01716	3260.39
2	5	62.16 ± 0.01770	9829.94
3	2	76.30 ± 0.01552	679.02
3	3	67.14 ± 0.01714	2207.58
3	4	60.87 ± 0.01781	11400.49
3	5	56.12 ± 0.01811	111023.14
4	2	74.51 ± 0.01590	679.71
4	3	63.67 ± 0.01755	4726.01
4	4	56.58 ± 0.01809	51714.16
5	2	72.05 ± 0.01637	798.98
5	3	61.20 ± 0.01778	9030.14

Table 5.7: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for imbalanced leaf class distribution (10:1), 100 examples per class, and random classifiers.

and discriminancy we observe. However, all numbers are still well above the required thresholds. If the class imbalance is introduced to the leaf level of a class hierarchy, then both the degree of consistency and the degree of discriminancy significantly improve comparing to the uniform class distribution (Table 5.7).

Finally, we compare the two measures for more "realistic" classification results, where predictions are better than random guesses. For this, we assign examples randomly with the probability of assigning to their true categories twice as large as the probability of assigning them to any incorrect category. The results are presented in Table 5.8. In these more practical settings, the degree of consistency is considerably higher than it was for random classifiers. At the same time, the degree of discriminancy is just slightly lower. For more accurate classification results, where a correct decision is made 5 times more often than an incorrect one, the degree of consistency is extremely high reaching over 70% even for large hierarchies (Table 5.9). The degree of discriminancy is getting smaller though stays still far above 1.

Overall, the experiments support our intuition that the hierarchical hF-measure is statistically consistent while more statistically discriminating than standard "flat" F-measure. By testing the two measures on a variety of parameter settings, such as the

hierarchy size		degree of consistency	degree of
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	81.53 ± 0.01416	152.00
2	3	77.43 ± 0.01525	422.09
2	4	73.82 ± 0.01604	989.06
2	5	69.21 ± 0.01684	3118.95
3	2	81.73 ± 0.01410	175.36
3	3	75.17 ± 0.01576	755.73
3	4	67.61 ± 0.01708	6296.94
3	5	62.49 ± 0.01767	78569.58
4	2	80.97 ± 0.01432	206.06
4	3	72.02 ± 0.01638	1913.21
4	4	60.97 ± 0.01780	27575.29
5	2	80.25 ± 0.01453	281.10
5	3	67.66 ± 0.01707	4412.88

Table 5.8: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for uniform class distribution, 100 examples per class, and "realistic" classification results (correct prediction is twice as probable as incorrect one).

hierarchy size		degree of consistency	degree of
out-degree	**depth**	**(with 99% confidence interval)**	**discriminancy**
2	2	87.39 ± 0.01211	140.36
2	3	85.74 ± 0.01276	372.59
2	4	83.94 ± 0.01340	883.24
2	5	81.50 ± 0.01417	2702.15
3	2	88.43 ± 0.01167	164.69
3	3	86.12 ± 0.01262	647.74
3	4	80.91 ± 0.01434	4535.26
3	5	70.70 ± 0.01661	29640.21
4	2	89.13 ± 0.01136	203.09
4	3	84.01 ± 0.01337	1379.32
4	4	70.75 ± 0.01660	15398.48
5	2	88.74 ± 0.01153	258.67
5	3	80.62 ± 0.01442	3138.80

Table 5.9: The degree of consistency and discriminancy for hF-measure over "flat" F-measure for uniform class distribution, 100 examples per class, and "realistic" classification results (correct prediction is 5 times as probable as incorrect one).

size of a class hierarchy, the number of instances, the a priori class distribution, and the probability of correct classification, we always obtain the degree of consistency higher than 50% and the degree of discriminancy hundreds or thousands times larger than 1. This confirms the superiority of the hierarchical measure over the "flat" measure at least on (tested) hierarchies of small and moderate sizes.

5.4.5 Allowing a trade-off between classification precision and classification depth

Similar to the pair of standard precision and recall, hierarchical precision and recall allow a trade-off: depending on the nature of the hierarchical classification task, we may prefer high precision at the cost of recall, which means that we classify mostly into high level categories, or we may prefer higher recall at the cost of precision, which means that we classify into the most specific categories. Combining the two measures into one hF-measure, we can set $\beta < 1$ if we are interested in highly precise classification, or we can set $\beta > 1$ if we want as much detail classification as possible.

5.5 Summary

In this chapter we discuss hierarchical performance evaluation measures. We formally introduce a set of intuitively desired characteristics for a hierarchical measure and compare the existing evaluation techniques based on these properties. We show that none of the measures proposed to date possesses all the desired properties and, therefore, introduce a novel hierarchical evaluation technique based on the notions of precision and recall adapted to the hierarchical settings. We formally prove that the new measure exhibits all the desired characteristics. Furthermore, we demonstrate that it is statistically consistent, yet more discriminating than the conventional "flat" evaluation techniques.

Chapter 6

Experimental results

In this chapter we experimentally compare the two hierarchical learning algorithms proposed in Chapter 4 on several real and synthetic datasets. Furthermore, we compare the two algorithms with the conventional "flat" method that ignores any hierarchical information. We limit the experiments to the two hierarchical techniques as they are the only ones known to us that produce classification consistent with a given class hierarchy. Comparison to an inconsistent classifier would be unfair since consistent and inconsistent label sets differ radically especially for large, real-life hierarchies. The only exception made is for the "flat" algorithm. Here we follow an established practice in the hierarchical text categorization research where hierarchical methods are often compared to the corresponding non-hierarchical, "flat" techniques. This comparison demonstrates the value of the hierarchical research.

For performance evaluation we employ the novel hierarchical measure introduced in Chapter 5. We would like to note that the evaluation procedure using the hierarchical evaluation measure is suitable not only for comparing two hierarchical learning algorithms, but also for comparing a hierarchical method and the "flat" algorithm. As discussed in Section 5, the hierarchical measure gives us an opportunity to reward partially correct classification and discriminate different kinds of misclassification errors, which is essential for hierarchical classifiers. At the same time, the hierarchical measure gives an advantage to the "flat" method by automatically extending the set of categories predicted by the "flat" algorithm with all their ancestor categories. The hierarchical learning approaches, on the other hand, have to explicitly predict all the correct categories including all their ancestors.

Both hierarchical as well as the "flat" approach are executed with AdaBoost.MH as

dataset	class hierarchy			number of documents			number of attributes
	number of categories	depth	out-degree	total in dataset	training	testing	
20 newsgroups	15	2	3	14,997	10,523	4,474	1,487
reuters-21578	120	2	20	11,367	7,952	3,414	2,035
RCV1_V2	103	4	4.68	634,831	30,208	33,275	1,696

Table 6.1: Characteristics of the text corpora used in the experiments. The number of training and test documents and the number of attributes are averaged over 10 trials.

an underlying learning algorithm. The same number of boosting iterations is performed for the "flat", the global approach, and each subtask of the local classifier.

The experiments reveal the dominance of the hierarchical approaches over the "flat" algorithm. Moreover, the advantage of the hierarchical techniques gets more evident on larger class hierarchies. Between the two hierarchical approaches, the global algorithm shows the best performance on all synthetic and some of the real datasets. In particular, its classification is more precise while a little inferior in recall. Thus, we recommend it for hierarchical classification tasks where precision is crucial.

6.1 Datasets

6.1.1 20 newsgroups

20 newsgroups is a widely used dataset of Usenet postings collected by Ken Lang [Lang, 1995]. It consists of 20 categories each having approximately 1000 documents. Each document is considered to belong to exactly one category. The original dataset has no hierarchical structure. Nevertheless, McCallum et al. suggested a two-level tree hierarchy by grouping thematically 15 (out of 20) categories in 5 parent nodes [McCallum et al., 1998]. The resulting hierarchical tree is presented in Appendix B.

In our experiments, the data are split into training and test sets reserving two thirds for training and the rest for testing. We keep the initial uniform class distribution. All experiments are repeated on 10 random train/test splits.

6.1.2 Reuters-21578

Reuters-21578 is another widely used dataset of news articles collected by David Lewis[1]. It consists of 21578 documents and 135 thematic categories (topics). Each document is labeled with zero, one, or several categories. We discard the documents that have no labels ending up with 11,367 documents and 120 categories. The thematic categories form a two-level tree hierarchy with 6 parent nodes. The hierarchy is presented in Appendix B.

As for 20 newsgroups, the data are split into training and test sets (two thirds for training, one third for testing) keeping the initial class distribution. All experiments are repeated on 10 random train/test splits.

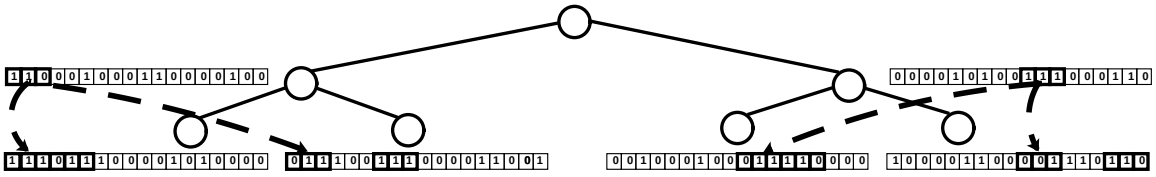
6.1.3 RCV1_V2

Reuters Corpus Volume 1 (RCV1) is a new benchmark collection of news articles recently made available by Reuters Ltd. for research purposes. The cleaned version of the corpus, called RCV1_V2, appeared later due to the effort by David Lewis and colleagues [Lewis et al., 2004]. The dataset consists of over 800,000 documents comprising all English language news stories written by Reuters journalists in the period of one year, from August 20, 1996 to August 19, 1997. All articles were manually or semi-automatically labeled with categories from three different sets: Topics, Industries, and Regions. We exploit the Topics categories that form a 4-level hierarchy with 126 nodes. Only 103 categories were actually used for document labeling. Originally, all Topics categories are assigned to documents in a hierarchically consistent manner, *i.e.* all ancestor labels are included with any given fine-grain topic. Thus, we had to remove ancestor labels for experiments with "flat" algorithm to simulate the non-hierarchical settings.

Due to the large size of the corpus, we are able to split the data into training and testing subsets in a time-sensitive manner. Data from 10 full months of the mentioned period (September, 1996 - June, 1997) are brought into play to form 10 splits: the first half of a month (from the 1st to the 14th) is used for training, while the second half is used for testing. In this way, we simulate the operation of a learning system in the real-life settings where classifiers are trained on older, archive data and tested on new, recently acquired instances.

[1] http://www.daviddlewis.com/resources/testcollections/reuters21578/

a) inherited attribute distributions



b) no inheritance of attribute distribution

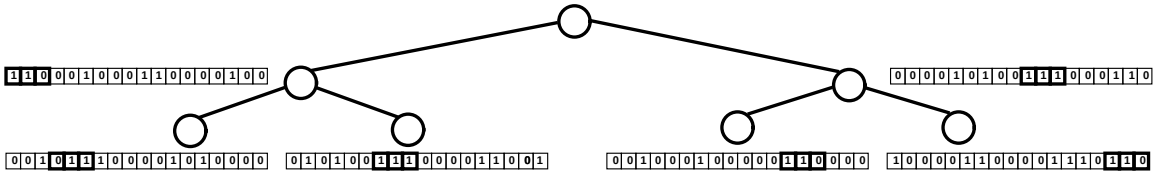


Figure 6.1: Generation of synthetic data with inheritance of attribute distribution (a) and with no inheritance (b). Each category in a hierarchy is represented by 3 bits. Consequently, a binary 2-level balanced tree hierarchy, which has 6 nodes (excluding the root), is represented by an 18-bit vector. The instances are generated randomly according to a specific distribution. In the first case (a), an instance that belongs to category $c_i \in C$ has a high probability (70%) of 1s in the bits corresponding to any category from $Ancestors(c_i)$; all other bits will be 1s with a low probability (20%). In the second case (b), only bits that correspond to the category c_i have the high probability of being 1. In both figures, bits with the high probability of 1 are shown in bold.

The datasets are summarized in Table 6.1. For all learning algorithms compared in the experiments, the data are pre-processed in the same way. First, stop words are removed, and the remaining words are normalized with the Porter stemmer [Porter, 1980]. Then, a simple but effective feature selection method is employed: all stems occurring in fewer than n documents are discarded. The number n is chosen for each dataset separately to keep the data files at manageable sizes. Finally, the remaining stems are converted into binary attributes (a stem is present or not).

6.1.4 Synthetic data

We make use of synthetic data to be able to control the size of a class hierarchy and the presence or absence of attribute inheritance between an ancestor class and its descendant classes. The data are designed as follows. For a specified number of levels and for a specified out-degree, *i.e.* the number of children nodes for each intermediate node, we build a balanced tree hierarchy. For each class, including the internal ones, we allocate 3

characteristics	**hierarchical**		**"flat" (non-hierarchical)**
	local	**global**	
description	generalized top-down level-by-level pachinko machine (Section 4.1)	AdaBoost.MH applied to consistently labeled training data (Section 4.2)	AdaBoost.MH applied to the *set* of all categories in a class hierarchy
learning procedure	local	global	global
taking advantage of hierarchical information	yes	yes	no
extending training data	yes	yes	no

Table 6.2: Comparative characteristics of the three learning algorithms: hierarchical local, hierarchical global, and "flat".

binary attributes and generate 10 training and 5 test instances per class. Each instance is assigned to exactly one class (single-label categorization). The instances are generated randomly according to the following distribution: attributes associated with the class of an instance are set to 1 with 70% probability, all other attributes are set to 1 with 20% probability[2]. We test synthetic data for two extreme situations (Figure 6.1). The first one is when each class inherits the distribution of attributes from its parent class on top of its own distribution. In other words, the attributes for a class and all its ancestor classes have the high probability (70%) of 1 (shown in bold in Figure 6.1(a)); all other attributes have the small probability (20%). The second situation is when there is no inheritance of attribute distribution: only the attributes associated with the class of an instance have 70% probability of 1 (shown in bold in Figure 6.1(b)), all others have 20% probability. We ran experiments for hierarchies with the number of levels and out-degree each ranging from 2 to 5. Experiments are repeated 100 times for every configuration.

6.2 Learning algorithms

In the following experiments we compare the performances of the three learning algorithms (Table 6.2):

- **Hierarchical local approach**. This is a generalized hierarchical top-down level-by-level pachinko machine described in Section 4.1. The algorithm splits the initial

[2]The numbers 70% and 20% are chosen arbitrarily to represent a fairy large and a fairy small probabilities to separate attributes associated with different classes.

classification task into subtasks according to a class hierarchy. It proceeds in a top-down fashion first picking the most relevant categories of the top level and then recursively making the choice among the low-level categories, children of the relevant top-level categories. In each classification subtask, the training sets of internal nodes are extended with the examples of their descendant classes.

- **Hierarchical global approach**. This is a novel hierarchically consistent global algorithm introduced in Section 4.2. It builds a single classifier to discriminate all the categories in a given class hierarchy simultaneously. The algorithm proceeds in three steps: re-labeling of the training data in the consistent manner, application of a regular multi-label learning algorithm (AdaBoost.MH) on the modified data, and (possible) post-processing to ensure the consistent classification of test instances. As in the local approach, the training sets of all internal nodes are extended with the examples of their descendant classes.

- **"Flat" approach**. This is standard, non-hierarchical AdaBoost.MH applied to the *"flat" set* of all categories (internal and leaves) from a given class hierarchy. It builds a single classifier for all categories similar the hierarchical global approach. Yet, the class relations are ignored in this algorithm, and the training data are *not* extended in the hierarchical way.

6.3 Results

6.3.1 Hierarchical vs. "flat" learning algorithms

The first set of experiments compares the performance of hierarchical approaches, local and global, with the performance of the "flat" approach. The results are presented in Tables 6.3 and 6.4. Figure 6.2 summarizes the results plotting one point for each of the 20 synthetic (star points) and 3 real (square points) datasets used in the experiments. Clearly, for the hF-measure most of the points lie above the diagonal line $y = x$, which indicates that both hierarchical approaches significantly outperform the "flat" algorithm on all real and most of the synthetic data. The differences in performance are quite impressive reaching up to 55% on synthetic data with inherited attribute distributions[3]. This is not surprising since these data were designed specifically to represent

[3]Figures 6.2 and 6.3 do not show the distinction between the two types of synthetic data (with and without attribute distribution inheritance) as they aggregate the results.

ideal testbeds for the hierarchical approaches. The only two exceptions of the superior performance of the hierarchical algorithms are the synthetic data with the smallest binary class hierarchies (2-level and 3-level) with no attribute distribution inheritance. Synthetic data with no attribute distribution inheritance were intended to represent the most challenging situation for the hierarchical methods. Indeed, we can see that these datasets are harder to learn for all techniques, hierarchical as well as "flat". This can be explained by insufficient amount of training data. Each category is defined only by 3 attributes and the values of the attributes are set probabilistically. Thus, 10 training examples per class do not provide enough information to learn an accurate model for a class. As a result, the performance of all tested algorithms considerably deteriorates on these data. When the number of categories is small, *e.g.* with 2-level and 3-level binary hierarchies, the "flat" method is able to produce quite accurate classifiers and surpass the local hierarchical algorithm. At the same time, the global approach is superior to "flat" on all tested data.

Another important observation is that the local approach is generally less accurate (in terms of hierarchical precision) than "flat" while its recall is always higher (up to 63%) (Figure 6.2, the top row). The global approach, on the other hand, outperforms the "flat" method in both precision and recall on all data, except Reuters-21578 where both algorithms show similar precision (Figure 6.2, the bottom row). Both the "flat" and the hierarchical global algorithms work with the global information, *i.e.* with all the categories and all the data simultaneously. Therefore, they assign instances only to those categories that have enough support from the training data. The local algorithm, on the contrary, works with the local information failing to see the big picture. Since at each classification node it deals with only a few categories, it tends to assign labels at each level pushing instances deep down the hierarchy. As a result, it can lose precision on hard to classify instances and categories with insufficient training data. When the number of categories gets bigger, the "flat" algorithm fails to keep up and produces very poor results. The inadequate amount of training data prevents it from making informed decisions and results in many instances left unresolved. For example, on a ternary 4-level hierarchy (120 categories) it is capable to assign at least one category to only 2.56% of test instances while the global hierarchical approach classifies 93.78% of instances. Such a conservative policy results in very low values of recall while maintaining reasonable numbers for precision. The hierarchical approaches work with extended training sets having much more data, especially at high level categories and, therefore, are able to make reliable decisions at least at the top of a hierarchy. This results in considerably

dataset	**out-degree**	**depth**	**boost. iter.**	**"flat"**			**hierarchical local**		
				hP	**hR**	hF_1	**hP**	**hR**	hF_1
20 newsgroups	3	2	500	75.81	75.23	75.51	80.85	79.19	**80.01**
reuters-21578	20	2	500	91.31	83.18	87.06	90.75	87.54	**89.11**
RCV1_V2	4.68	4	500	74.14	72.10	73.10	72.19	75.99	**74.03**
synthetic	2	2	200	70.76	66.60	68.30	67.26	80.82	**73.42**
(with attr.	2	3	500	67.25	51.82	58.35	62.79	77.56	**69.40**
inheritance)	2	4	1000	68.37	33.52	44.90	61.93	75.83	**68.18**
	2	5	2000	71.24	12.27	20.88	62.70	75.34	**68.44**
	3	2	400	58.46	49.60	53.47	58.71	65.88	**61.99**
	3	3	1000	57.03	19.98	29.51	56.85	60.97	**58.81**
	3	4	3500	57.61	1.37	2.67	57.98	56.84	**57.40**
	4	2	600	50.81	35.03	41.35	53.24	55.47	**54.26**
	4	3	2500	50.11	3.76	6.98	52.67	48.84	**50.66**
	5	2	900	45.13	22.55	29.99	48.03	46.63	**47.26**
synthetic	2	2	200	63.79	60.36	61.69	53.62	67.88	59.83
(no attr.	2	3	500	46.78	39.22	42.47	37.81	52.92	44.00
inheritance)	2	4	1000	34.03	19.20	24.49	27.96	41.78	**33.44**
	2	5	2000	25.53	5.08	8.45	21.59	32.87	**26.03**
	3	2	400	43.58	39.94	41.53	39.08	50.42	**43.87**
	3	3	1000	24.66	10.31	14.50	21.85	33.31	**26.33**
	3	4	3500	16.23	0.41	0.79	14.88	22.73	**17.97**
	4	2	600	30.91	23.68	26.72	28.69	37.75	**32.51**
	4	3	2500	15.23	1.34	2.46	15.03	22.35	**17.96**
	5	2	900	23.38	13.59	17.14	23.09	30.01	**26.04**

Table 6.3: Performance of the hierarchical local and "flat" AdaBoost.MH on real text corpora and synthetic data. Numbers in bold are statistically significantly better with 99% confidence.

dataset	**out-degree**	**depth**	**boost. iter.**	**"flat"**			**hierarchical global**		
				hP	**hR**	hF_1	**hP**	**hR**	hF_1
20 newsgroups	3	2	500	75.81	75.23	75.51	81.32	77.31	**79.26**
reuters-21578	20	2	500	91.31	83.18	87.06	91.30	85.51	**88.31**
RCV1_V2	4.68	4	500	74.14	72.10	73.10	76.89	74.88	**75.86**
synthetic	2	2	200	70.76	66.60	68.30	76.93	75.76	**76.22**
(with attr.	2	3	500	67.25	51.82	58.35	76.38	72.28	**74.21**
inheritance)	2	4	1000	68.37	33.52	44.90	77.02	69.81	**73.22**
	2	5	2000	71.24	12.27	20.88	78.96	67.38	**72.70**
	3	2	400	58.46	49.60	53.47	66.19	61.13	**63.45**
	3	3	1000	57.03	19.98	29.51	69.02	54.20	**60.69**
	3	4	3500	57.61	1.37	2.67	71.68	49.03	**58.22**
	4	2	600	50.81	35.03	41.35	60.42	51.02	**55.25**
	4	3	2500	50.11	3.76	6.98	63.11	42.39	**50.70**
	5	2	900	45.13	22.55	29.99	55.57	42.12	**47.87**
synthetic	2	2	200	63.79	60.36	61.69	64.75	67.54	**65.95**
(no attr.	2	3	500	46.78	39.22	42.47	48.92	54.69	**51.53**
inheritance)	2	4	1000	34.03	19.20	24.49	37.22	43.80	**40.18**
	2	5	2000	25.53	5.08	8.45	29.95	35.81	**32.61**
	3	2	400	43.58	39.94	41.53	46.17	50.32	**48.02**
	3	3	1000	24.66	10.31	14.50	29.87	30.14	**29.97**
	3	4	3500	16.23	0.41	0.79	21.96	21.87	**21.91**
	4	2	600	30.91	23.68	26.72	34.82	35.43	**35.01**
	4	3	2500	15.23	1.34	2.46	20.55	19.00	**19.70**
	5	2	900	23.38	13.59	17.14	28.48	26.01	**27.12**

Table 6.4: Performance of the hierarchical global and "flat" AdaBoost.MH on real text corpora and synthetic data. Numbers in bold are statistically significantly better with 99% confidence.

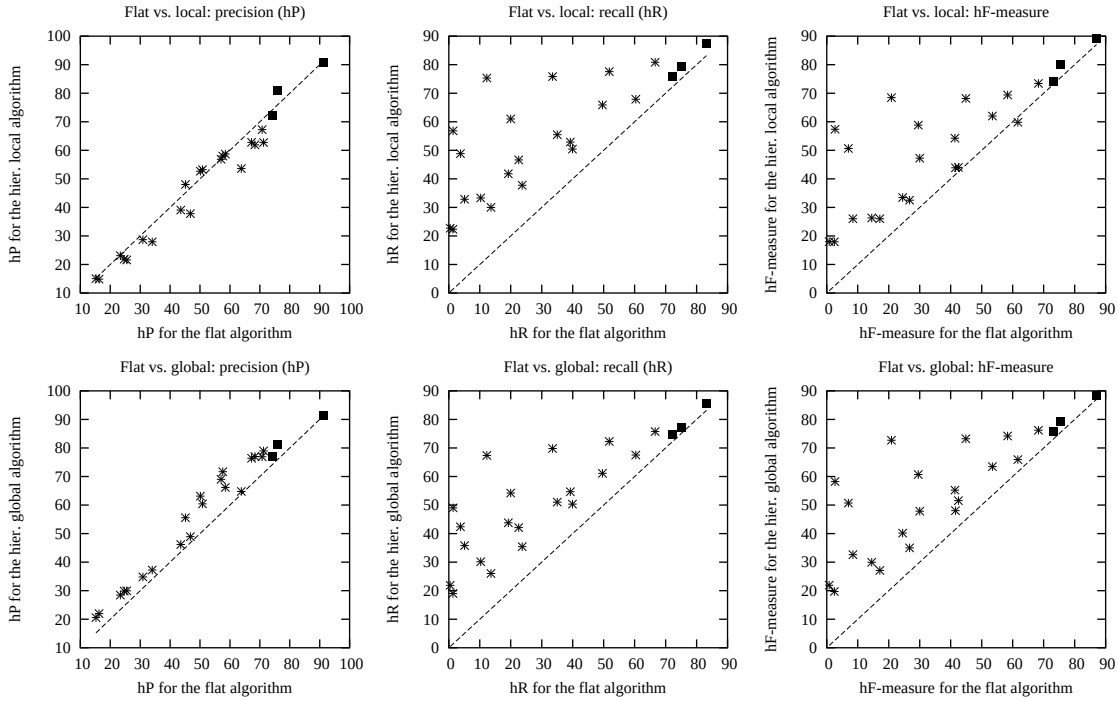


Figure 6.2: Performance comparison of the conventional "flat" algorithm with the hierarchical approaches, local (top) and global (bottom), in hierarchical precision (left), hierarchical recall (center), and hierarchical F-measure (right). All algorithms are executed with AdaBoost.MH as a base learner. Each star point represents one of the 20 synthetic datasets (with and without attribute distribution inheritance); each square point corresponds to one of the 3 real datasets. Points lying above the diagonal line show the superior performance of the hierarchical approaches over the "flat" one.

higher values of recall comparing to the "flat" method. Moreover, additional training data gives the opportunity to the global hierarchical algorithm to learn more accurate models comparing to the "flat" method.

Synthetic data (see Section 6.1.4) allow us to study the behavior of the algorithms subject to the size of a class hierarchy. The first conclusion we can draw is that increase in out-degree has a negative effect on all algorithms both in terms of precision and recall. Large out-degree corresponds to an enormous number of classes that the "flat" and global algorithms have to deal with at once. It also increases, yet to a substantially lesser degree, the number of classes the local method has to discriminate between in each classification subtask. At the same time, increase in depth of a hierarchy has no such straight-forward effect. When categories do not inherit attribute distributions (Figure 6.1(b)), deeper hierarchies only increase the complexity of a classification task

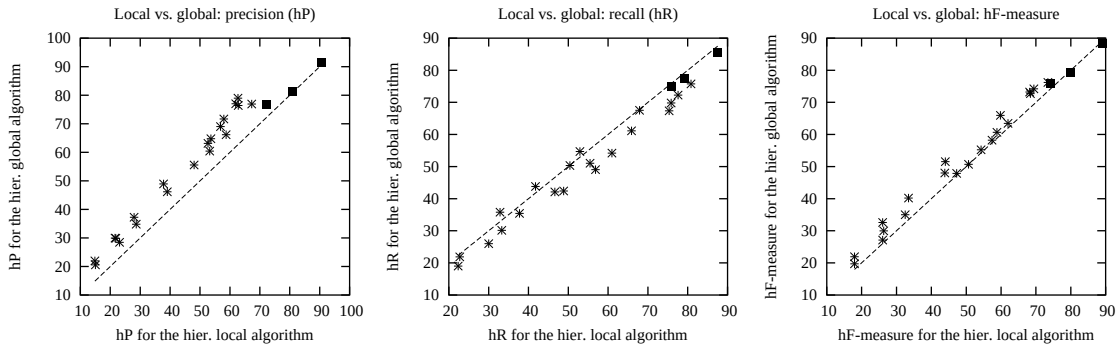


Figure 6.3: Performance comparison of the local and global hierarchical approaches in hierarchical precision (left), hierarchical recall (center), and hierarchical F-measure (right). Both algorithms are executed with AdaBoost.MH as a base learner. Each star point represents one of the 20 synthetic datasets (with and without attribute distribution inheritance); each square point corresponds to one of the 3 real datasets. Points lying above the diagonal line show the superior performance of the hierarchical global over the local approach.

and, therefore, correspond to lower values of precision and recall for all algorithms. However, when the categories do inherit attribute distributions (Figure 6.1(a)), extended training data in hierarchical approaches becomes extremely useful. As a result, the global technique is able to improve its precision with depth. The local method also benefits from extended training data, yet making more mistakes at lower levels. The "flat" approach cannot take advantage of the extended data, but gets an increased chance of classifying an instance in a correct neighborhood, since the related categories share quite a few attributes. Hence, precision of the "flat" algorithm does not deteriorate, as opposed to recall that decreases substantially. Overall, larger hierarchies (both in depth and out-degree) imply bigger gain in performance of the hierarchical techniques over the "flat" one.

6.3.2 Hierarchical global vs. local approaches

We now compare the performance of the hierarchical global vs. hierarchical local approaches. Table 6.5 presents the extracts of the relevant information from Table 6.3 and Table 6.4, and Figure 6.3 summarizes the results plotting one point for each of the 20 synthetic and 3 real datasets used in the experiments. For most synthetic and one real (RCV1_V2) tasks the global hierarchical approach outperforms the local method. In particular, the global algorithm is always superior to the local one in terms of precision (by

dataset	**out-degree**	**depth**	**boost. iter.**	**hierarchical local**			**hierarchical global**		
				hP	**hR**	hF_1	**hP**	**hR**	hF_1
20 newsgroups	3	2	500	80.85	79.19	**80.01**	81.32	77.31	79.26
reuters-21578	20	2	500	90.75	87.54	**89.11**	91.30	85.51	88.31
RCV1_V2	4.68	4	500	72.19	75.99	74.03	76.89	74.88	**75.86**
synthetic	2	2	200	67.26	80.82	73.42	76.93	75.76	**76.22**
(with attr.	2	3	500	62.79	77.56	69.40	76.38	72.28	**74.21**
inheritance)	2	4	1000	61.93	75.83	68.18	77.02	69.81	**73.22**
	2	5	2000	62.70	75.34	68.44	78.96	67.38	**72.70**
	3	2	400	58.71	65.88	61.99	66.19	61.13	**63.45**
	3	3	1000	56.85	60.97	58.81	69.02	54.20	**60.69**
	3	4	3500	57.98	56.84	57.40	71.68	49.03	**58.22**
	4	2	600	53.24	55.47	54.26	60.42	51.02	**55.25**
	4	3	2500	52.67	48.84	50.66	63.11	42.39	50.70
	5	2	900	48.03	46.63	47.26	55.57	42.12	**47.87**
synthetic	2	2	200	53.62	67.88	59.83	64.75	67.54	**65.95**
(no attr.	2	3	500	37.81	52.92	44.00	48.92	54.69	**51.53**
inheritance)	2	4	1000	27.96	41.78	33.44	37.22	43.80	**40.18**
	2	5	2000	21.59	32.87	26.03	29.95	35.81	**32.61**
	3	2	400	39.08	50.42	43.87	46.17	50.32	**48.02**
	3	3	1000	21.85	33.31	26.33	29.87	30.14	**29.97**
	3	4	3500	14.88	22.73	17.97	21.96	21.87	**21.91**
	4	2	600	28.69	37.75	32.51	34.82	35.43	**35.01**
	4	3	2500	15.03	22.35	17.96	20.55	19.00	**19.70**
	5	2	900	23.09	30.01	26.04	28.48	26.01	**27.12**

Table 6.5: Performance of the hierarchical local and global AdaBoost.MH on real text corpora and synthetic data. Numbers in bold are statistically significantly better with 99% confidence.

up to 16%) while slightly yielding in recall (by up to 8%) (Figure 6.3). Both algorithms take advantage of extended training data. However, the global approach explores all the categories simultaneously (in a global fashion) assigning only labels with high confidence scores. The local method, on the other hand, uses only local information and, therefore, is forced to make classification decisions at each internal node of a hierarchy, in general, pushing most instances deep down. For example, on a ternary 4-level tree hierarchy the local algorithm classifies 54.79% of test instances to the deepest level while the global method assigns only 2.27% of instances to the leaf classes. This reflects the conservative nature of the global approach comparing to the local one. Therefore, it should be the method of choice for tasks where precision is the key measure of success. For example, in the task of indexing biomedical articles with Medical Subject Headings (MeSH) (see Section 7.1), high precision might be preferable to high recall so that searching the Medline library with MeSH terms retrieves only documents relevant to a user's query. In this case, the hierarchical global approach will be more appropriate to use for this task.

Increase in depth (d) of a class hierarchy raises exponentially the number of classes ($\sim k^d$) and, as a result, the difficulty of the classification task for the global approach. It also adds an extra level of complexity for the local algorithm. On the other hand, deeper hierarchies bring more training data allowing both algorithms to learn better classification models for high level categories. This results in improved precision for the global approach. The local algorithm, however, diminishes this improvement with errors made on the additional level as opposed to the global method that mostly ignores low levels since they do not provide enough data to learn reliable models. Consequently, the global algorithm ends up with higher precision, but lower recall values comparing to the local technique. Increase in out-degree (k) only slightly (linearly) complicates the task for the local method while adding a significant number of categories ($\sim k^{d-1}$) to the global method. This is reflected in higher decrease in precision and overall the smaller advantage of the global algorithm on synthetic hierarchies with large out-degrees.

6.4 Summary

This chapter presents an experimental comparison of two hierarchical learning approaches, the generalized local pachinko machine and the novel global hierarchically consistent algorithm, with the conventional "flat" approach. The experiments demonstrate that both hierarchical techniques outperform the "flat" method on almost all tested datasets, often by a large margin. Moreover, the extent of the achieved improvement increases with

the size of a class hierarchy. Between the two hierarchical algorithms, the global one is superior to the local on most synthetic and some real problems. In particular, the global approach exhibits considerably higher precision, while slightly yielding in recall. Therefore, we recommend the global learning algorithm for classification tasks where precision is favored over recall and the local method for tasks where high recall is essential.

Chapter 7

Hierarchical text categorization in bioinformatics

As an application area we have chosen bioinformatics in view of the fact that it is an important, quickly developing area that has many text-related problems. More specifically, we address the task of indexing biomedical articles with MeSH terms and two genomics[1] tasks, namely functional annotation of genes from biomedical literature and gene expression analysis with background knowledge.

7.1 Indexing of biomedical literature with Medical Subject Headings

As our first application we have chosen the task of indexing biomedical articles with Medical Subject Headings (MeSH). MeSH is a manually built controlled vocabulary for the biomedical domain where terms are arranged into several hierarchical structures called MeSH Trees. We address this task as a hierarchical text categorization task with class hierarchies derived from the MeSH vocabulary. In other words, we classify biomedical articles from the Medline library into one or several keywords from a specified MeSH Tree. This task represents an interesting and challenging real-world application for hierarchical text categorization techniques developed in this research. MeSH indexing is an important part of the Medline library. Articles from thousands of journals get indexed on

[1]Genomics is a new scientific discipline that studies genes and their functions. It is characterized by high-throughput genome-wide experimental approaches combined with statistical and computational techniques of bioinformatics for the analysis of the results.

a daily basis. Having articles annotated with carefully chosen Medical Subject Headings helps users to quickly find relevant information in the ocean of millions of abstracts offered by the Medline database. At the same time, MeSH embodies very large, deep class hierarchies comprised of thousands of categories. Therefore, it is an excellent testbed for hierarchical learning algorithms.

7.1.1 Motivation

The Medical Subject Headings thesaurus is used by the National Library of Medicine for indexing articles from 4,800 of the world's leading biomedical journals. This indexing is essential for the search facilities of the Medline database. Using a Pubmed search mechanism, a user can type a text query to retrieve all Medline articles containing the words in the query. However, different terminology used by the authors and constantly evolving biomedical vocabulary pose a real challenge for the users to precisely express their information need. Formulating a good query that returns all and only relevant references is a difficult task, particularly for unexperienced users or people who are not specialists in the area of interest. MeSH indexing is intended to eliminate this problem. The controlled vocabulary of MeSH is carefully designed by trained indexers to represent biomedical concepts rather than English language words. One such concept corresponding to a MeSH heading implies a number of linguistic variants that can be found in manuscripts discussing this concept. As a result, a Pubmed search with MeSH terms would return all articles relevant to a query concept even if the words themselves do not occur in the texts. Furthermore, the Pubmed search engine can automatically replace some of the common words entered by a user with the corresponding MeSH headings, thus improving the search outcome.

Hundreds of thousands of articles need to be annotated every year. This requires tremendous effort on the part of the trained indexers and, therefore, is a costly and time-consuming process. Fully automatic or semi-automatic annotation techniques could significantly reduce the costs and speed up the process. As a result, several research studies have been conducted on this task [Cooper and Miller, 1998, Kim et al., 2001]. Unlike previous work, we address this problem from the hierarchical point of view.

We would like to observe that MeSH indexing calls for hierarchically consistent classification. Consistent classification will allow an article indexed with a term t to be retrieved in a search with a term $\hat{t}$, an ancestor of term t. For example, if a user is looking for articles on primates, the search engine should return not only the articles

Animals [B01]
Chordata [B01.150]
Vertebrates [B01.150.900]
Mammals [B01.150.900.649]
Primates [B01.150.900.649.801]
Haplorhini [B01.150.900.649.801.400]
Hominidae [B01.150.900.649.801.400.350]
Gorilla gorilla [B01.150.900.649.801.400.350.375]
>> Humans [B01.150.900.649.801.400.350.400]
Pan paniscus [B01.150.900.649.801.400.350.600]
Pan troglodytes [B01.150.900.649.801.400.350.620]
Pongo pygmaeus [B01.150.900.649.801.400.350.650]

Figure 7.1: Part of the MeSH "Organisms" (B) hierarchical tree. The MeSH Tree numbers in square brackets determine the position of a subject heading in the hierarchy.

indexed with term "primates", but also the articles indexed with the offspring concepts, such as "gorilla", "humans", etc. Also, all MeSH leaf and non-leaf concepts are proper terms to use in indexing. Therefore, our hierarchical learning algorithms presented in Chapter 4, which produce consistent classification and allow internal class assignments, are ideal candidates for this task.

7.1.2 Medical Subject Headings (MeSH)

Medical Subject Headings (MeSH) is a controlled vocabulary of the National Library of Medicine (NLM)[2]. It consists of specialized terminology used for indexing, cataloging, and searching for biomedical and health-related information and documents. The terms mostly represent biomedical concepts and are arranged hierarchically from most general to most specific in hierarchical structures called "MeSH Trees" of up to eleven levels deep. There are 15 hierarchical structures representing different aspects: A for anatomic terms, B for organisms, C for diseases, D for drugs and chemicals, etc. (the full list of categories is shown in Table 7.1). Figure 7.1 shows an excerpt from the "Organisms" hierarchy. At the top level, the hierarchies contain very general terms, such as "Animals" and "Bacteria". At lower levels, more specific terms are located, such as "Humans" and "Streptococcus pneumoniae". Although called trees, strictly speaking, the MeSH hierarchical structures are not trees as any term may appear in several places in a hierarchy.

There is a total of 22,997 subject headings in MeSH (2005 edition). Along with the main headings, also called descriptors, that characterize the subject matter, there are

[2]http://www.nlm.nih.gov/mesh/meshhome.html

A	Anatomy
B	Organisms
C	Diseases
D	Chemicals and Drugs
E	Analytical, Diagnostic and Therapeutic Techniques and Equipment
F	Psychiatry and Psychology
G	Biological Sciences
H	Physical Sciences
I	Anthropology, Education, Sociology and Social Phenomena
J	Technology and Food and Beverages
K	Humanities
L	Information Science
M	Persons
N	Health Care
Z	Geographic Locations

Table 7.1: MeSH hierarchical trees.

qualifiers that are used together with descriptors to characterize a specific aspect of a subject. For example, "drug effects" is a qualifier for subject "Streptococcus pneumoniae" to index articles that discuss the effects of drugs and chemicals associated with Streptococcus pneumoniae. In addition, there are so called entry terms or see references. These are synonyms or related terms linked to the corresponding subject headings, for example, Vitamin C see Ascorbic Acid. Entry terms can be beneficial for novices or occasional users not very familiar with the MeSH vocabulary to quickly locate a concept of interest. As the biomedical field is changing, the MeSH thesaurus is constantly evolving adopting new emerging concepts, discarding out-of-date terminology, and renaming the existing headings. Every year an updated version of MeSH is published by the National Library of Medicine.

The MeSH thesaurus is freely available from the NLM website. It is distributed electronically in two formats: XML and plain ASCII. Both formats include the information on all descriptors and qualifiers stating the name of a term, its position in a hierarchy, all possible qualifiers (for subject headings), cross-references, etc. The hierarchical structure of MeSH main headings is also available in a separate file `mtrees2005.bin`. This file contains all subject headings and their positions in MeSH Trees. The following is an example of the MeSH Trees file content that matches the structure shown in Figure 7.1:

```
Animals;B01
Chordata;B01.150
Vertebrates;B01.150.900
Mammals;B01.150.900.649
Primates;B01.150.900.649.801
Haplorhini;B01.150.900.649.801.400
Hominidae;B01.150.900.649.801.400.350
Gorilla gorilla;B01.150.900.649.801.400.350.375
Humans;B01.150.900.649.801.400.350.400
Pan paniscus;B01.150.900.649.801.400.350.600
Pan troglodytes;B01.150.900.649.801.400.350.620
Pongo pygmaeus;B01.150.900.649.801.400.350.650
```

Each entry includes a subject heading and a MeSH Tree number separated by a semicolon. Since one subject heading can have multiple occurrences in the hierarchical structures, several entries can correspond to one heading. A Tree number determines the exact position of an entry in a hierarchy. The first letter of a Tree number specifies a hierarchical tree (A-Z). The following numbers denote the position at each level of a hierarchy from top to bottom. In this way, a concept has the Tree number of its parent concept with three additional digits at the end to distinguish it from its siblings.

7.1.3 OHSUMED dataset

As a source of data we used a large test collection called OHSUMED [Hersh et al., 1994]. It was obtained by William Hersh and colleagues for medical information retrieval research. Later on, the corpus was utilized in the Text Retrieval Conference (TREC-9) competition (Filtering Track). The dataset contains 348,566 references to biomedical articles from the Medline library from 270 medical journals over a five-year period (1987-1991). All references have the following information: title, abstract (possibly empty), MeSH indexing terms, author, source, and publication type. Originally, the corpus was designed for evaluation of information retrieval systems; therefore, 101 queries generated by actual physicians in the course of patient care are provided along with the document relevance judgement. However, it has also been widely used in the text categorization research as the MeSH annotations provide a high-quality set of category labels. Since Medical Subject Headings are arranged hierarchically and any article can be indexed

dataset	class hierarchy			number of documents			number of attributes
	number of categories	depth	out-degree	total in dataset	training	testing	
OHSUMED B	949	10	2.23	71,181	13,944	15,011	2,245
OHSUMED F	673	7	3.38	40,789	7,926	8,692	3,177

Table 7.2: Characteristics of the OHSUMED data used in the experiments. All numbers (except the total number of documents in dataset) are averaged over 4 trials.

with any number of headings, this collection represents an ideal testbed for hierarchical multi-class multi-label text categorization techniques.

The references in the corpus are organized in 5 files by the year of publication. We take advantage of this natural separation and obtain the training/test splits in a time-sensitive manner, similar to what we do with the RCV1_V2 dataset. For OHSUMED, we collect 4 splits with one year of data used for training and the following year of data used for testing. Accordingly, the first split is comprised of 1987 articles (training set) and 1988 articles (test set), the second split contains 1988 articles (training set) and 1989 articles (test set), and so on. We experimented with two MeSH hierarchical structures, "Organisms"(B) and "Psychiatry & Psychology" (F), as representatives of fairly large hierarchies with strong biomedical content. The category sets include only the main headings ignoring all possible qualifiers. The datasets are summarized in Table 7.2.

7.1.4 Results

We apply both hierarchical local and global learning techniques described in Section 4 with AdaBoost.MH as a basic learner on the OHSUMED data. We compare the performance of these algorithms with the "flat" version of AdaBoost.MH. All algorithms are run on the identical settings: 4 training/test splits, the same attribute sets, 500 boosting iterations[3]. The results are presented in Table 7.3. Evidently, the results of these experiments are in full agreement with the conclusions derived on the similar set of experiments on textual and synthetic data described in Section 6. Both hierarchical algorithms significantly outperform the "flat" AdaBoost.MH on both MeSH Trees. The second hierarchy, "Psychiatry & Psychology" (F), presents more challenge for all

[3]For the hierarchical local approach, AdaBoost.MH is executed for 500 iterations at each node of a class hierarchy.

dataset	**"flat"**			**hierarchical local**			**hierarchical global**		
	hP	**hR**	hF_1	**hP**	**hR**	hF_1	**hP**	**hR**	hF_1
OHSUMED B	78.87	62.82	69.92	70.40	74.08	72.17	79.15	71.60	**75.17**
OHSUMED F	52.98	30.13	38.37	44.11	49.85	46.79	55.00	45.62	**49.87**

Table 7.3: Performance of the "flat", hierarchical local, and hierarchical global AdaBoost.MH on the OHSUMED data. Numbers in bold are statistically significantly better with 99% confidence.

algorithms having larger out-degree and fewer training data per class. As a result, the performance of all algorithms is considerably lower, but the differences in performance between the "flat" and hierarchical methods are more pronounced reaching up to 11.5%. In addition, the hierarchical global approach does better than "flat" in both precision (by ∼0.3-2%) and recall (by ∼9-15%) while the local approach is defeated by "flat" in precision (by ∼8-9%) being superior in recall (by ∼11-20%). Comparing the two hierarchical algorithms, we can see that the global approach wins in precision by ∼8.5-11%, but yields in recall by ∼2.5-4%. Overall, the global approach shows the best performance surpassing the local algorithm by about 3% on each dataset.

These experiments demonstrate that the proposed hierarchical approaches can be successfully applied to the task of automatic MeSH annotation of biomedical literature. The application of these algorithms results in good quality annotations at least on some MeSH hierarchical structures. More importantly, we experimentally show that the hierarchical approaches are more suitable for this task being superior to the "flat" method on all tested category hierarchies. Two obvious extensions can be made in future work to improve the value of automatic annotations. First, more training data need to be accumulated, especially for low level categories. Second, the feature sets can be improved by taking into account the background knowledge, such as biologically relevant n-grams ($n \geq 2$), named entities (*e.g.* chemical/drug names or medical procedures), etc.

7.2 Functional annotation of genes from biomedical literature

Our second application concerns the task of functional annotation of genes [Kiritchenko et al., 2004, Kiritchenko et al., 2005a]. We propose a system to classify genes/gene

products into Gene Ontology (GO) terms based on the classification of documents from the Medline library[4] that describe the genes. The purpose of this task is to retrieve the known functionality of a group of genes from the literature and translate it into a controlled vocabulary of the Gene Ontology (see Figure 7.2).

When new information about genes is discovered, it typically becomes known to the scientific community through a journal publication. Thus, an electronic library such as Medline represents a vital resource of most up-to-date results achieved in biomedical sciences. Our goal is to use this resource to find the functional information for a given group of genes.

We address this task in two steps: learning and classification. In the first step (learning), we use annotations of well-known genes from genetic databases (*e.g.* SGD) to collect training data that consist of Gene Ontology terms manually assigned to the genes and corresponding Medline references that support these assignments. Then, we learn a classifier on these training data. In the second step (classification), we use the classifier to annotate less studied genes whose function assignments are missing in the database. For this, we search the Medline library for the documents relevant to the genes and classify these documents into one or several Gene Ontology terms.

Gene Ontology [Ashburner et al., 2000] constitutes a hierarchy of carefully chosen terms that describe all possible gene/gene product functionalities. By classifying biomedical articles associated with a gene into GO terms, we attempt to retrieve the biological functions that the gene performs in a given organism. Gene function is an essential characteristic of genes required for understanding many processes in a living organism. Therefore, information obtained by the proposed classification system is essential for life scientists in their everyday activities. In addition, by using GO terms as the outcomes of classification, we achieve yet another goal of converting diverse vocabularies introduced by the authors of publications into one standardized terminology.

7.2.1 Motivation

The problem of functional annotation of genes is of great importance for biomedical and bioinformatics communities. As has been shown in many studies, gene mutation is the primary cause of many diseases including cancer and hemophilia [Burke, 2003, Wooster and Weber, 2003]. For some common diseases, such as asthma, diabetes or Alzheimer's

[4]Not all entries in the Medline database have an access to the full texts. Therefore, we use only titles and abstracts of the articles.

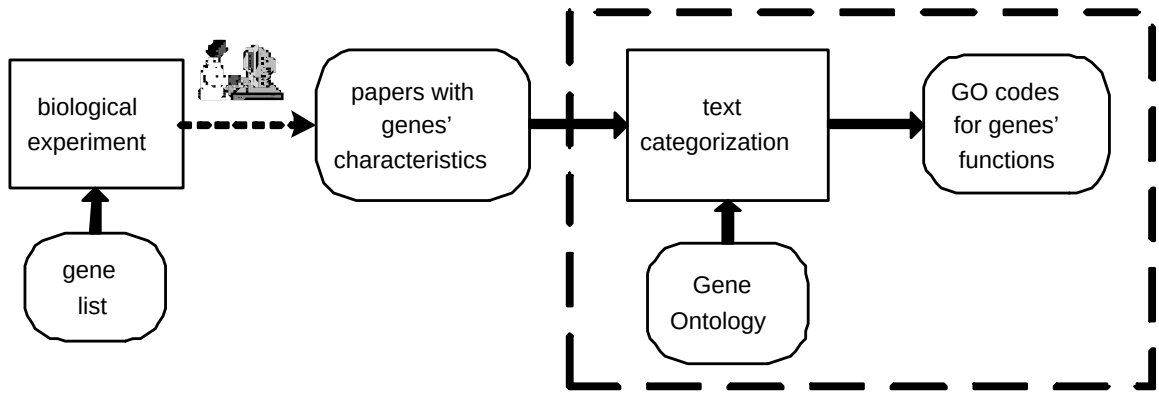


Figure 7.2: Functional annotation process. Genes' functions are determined in biological experiments and stated in scientific publications. The goal of our task (in the dashed box) is to retrieve these functions and translate them into corresponding GO codes.

disease, both genetic and environmental factors play an important role [Guttmacher and Collins, 2002, Burke, 2003]. Genomics research is aimed, among other things, at studying the genes responsible for diseases, their important variations, their interactions, and their behavior under different conditions, leading us to understanding the underlying mechanisms of these diseases and discovering efficient treatment. Our work could help biologists in genomics research by providing them with relevant information automatically extracted from scientific literature and structured in a standardized way.

In many genomics studies one of the major steps is the gene expression analysis using high-throughput DNA microarrays. Measuring the expression profiles of genes from normal and disease tissues or from the same tissue exposed to different conditions can help discover genes responsible for the disease. It can also shed light on the functionality of genes whose role was previously unknown or ESTs (Expressed Sequence Tags). Traditionally, most computational research on analyzing gene expression data has focused on working with microarray data alone, using statistical [Eisen et al., 1998] or data mining [Furey et al., 2000, Hvidsten et al., 2003] tools. However, raw gene expression data are very hard to analyze even for an experienced scientist. On the other hand, there exists a wealth of information pertaining to the function and behavior of genes, described in papers and reports. Most of these are available on-line and could potentially be useful in the analysis of gene expression, if we had a way of harvesting this information and combining it synergistically with the knowledge acquired from the microrray data experiments. Specifically, our research is aimed at providing molecular biologists with known functional information on genes used in the experiments in order to make microarray

results and their analysis more biologically meaningful.

Another important aspect of any genomics study is the validation step. To become widely accepted, new discoveries have to be validated by further biological experiments or confirmed by related research. One of the common practices for validation is to check scientific literature for similar results. For example, suppose that in an Alzheimer's study several genes were identified as highly related to the disease. Then, the literature search of related research showed that some of these genes have already been known as associated with other neurological disorders. This fact would be a supporting evidence for the results of the Alzheimer's study. However, such validation requires extensive literature search, which is most often done manually. Automatic text analysis techniques can effectively replace manual effort in this area.

Even though many genes for well-studied organisms, such as *Escherichia coli* or *Saccharomyces cerevisiae*, have been already annotated in specialized databases (EcoCyc, SGD), information on many other genes currently can be found only in scientific publications. Public databases are created and curated manually; thus, they cannot keep up with an overwhelming number of new discoveries published on a daily basis. Furthermore, these databases often use different vocabularies to describe gene functionality, which raises an additional challenge for integrating the results. Consequently, genomics databases are not always adequate to find the requisite information. Therefore, we need to apply text mining and categorization techniques to retrieve up-to-date information from biomedical literature and translate it into a standardized vocabulary to help life scientists in their everyday activities. At the same time, the same process can be used as a tool to assist in updating and curating databases.

The present research continues the work on automatic functional annotation of genes from biomedical literature (*e.g.* [Raychaudhuri et al., 2002, Catona et al., 2004]), described in Section 3.5.5, by introducing the hierarchical text categorization techniques to the problem. The hierarchical techniques explore the additional information on class relations, which may lead to an improved performance of a classification system. At the same time, hierarchical categorization allows a trade-off between classification precision and the required level of details on gene functionality.

7.2.2 Gene Ontology

We employ hierarchical text categorization techniques described in Section 4 to classify Medline articles associated with given genes into one or several functional categories.

These functional categories come from the Gene Ontology (GO) [Ashburner et al., 2000]. In biology controlled vocabularies for different subdomains are traditionally designed in the form of ontologies [Rison et al., 2000, Stevens et al., 2000]. Gene Ontology is quickly becoming a standard for gene/protein function annotation, and therefore, it is our choice for the hierarchy of categories.

Gene Ontology describes gene products in terms of their associated molecular functions, biological processes, and cellular components in a species-independent manner [Ashburner et al., 2000]. Molecular function describes activities that are performed by individual gene products or complexes of gene products. Examples of high-level molecular functions are translation activity, catalytic activity and transporter activity; example of low-level function is vitamin B12 transporter activity. A biological process consists of several distinct steps and is accomplished by sequences of molecular functions. Examples of high-level biological processes are development, behavior and physiological process; example of low-level process is tissue regeneration. A cellular component is a component of a cell, such as nucleus, membrane, or chromosome, associated with a gene product.

The Gene Ontology consists of three hierarchies, one for each of the three aspects. Each hierarchy is a directed acyclic graph (DAG). Each GO term is given a unique identifier called a GO code (for example, the term "metabolism" has code "GO:0008152"). Figure 7.3 shows a part of the biological process hierarchy.

The hierarchies are contained in 3 files: `function.ontology`, `gene.ontology`, and `component.ontology`, respectively. Each line in these files corresponds to one GO code. The format of the line is as follows:

```
< | % term [; db cross ref]* [; synonym:text]*  [ < | % term]*
```

where "`|`" represents "or" and "`[]*`" indicates an optional item that can be repeated several times. The items have the following meaning:

- % represents the "is-a" relationships;
- < represents the "part-of" relationships;
- "db cross ref" is a general database cross reference, which refers to an identical object in another database;
- "synonym:text" is a list of synonyms in textual format.

The items on a line are separated by a semi-colon.

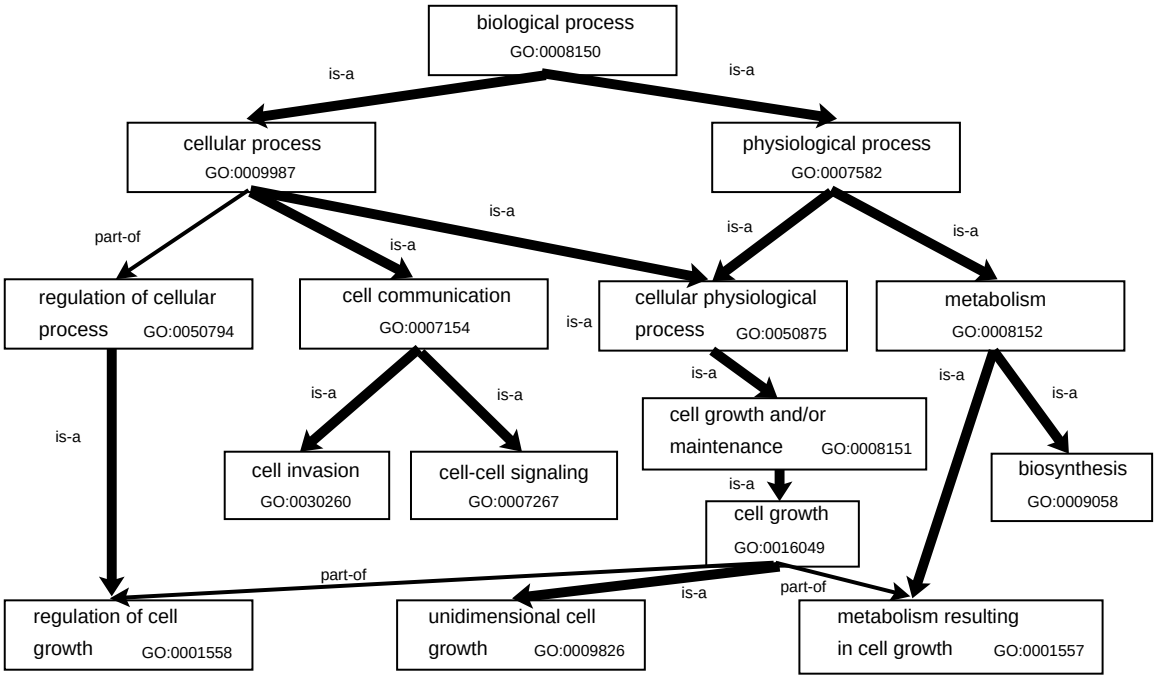


Figure 7.3: Part of the biological process hierarchy of the Gene Ontology. The hierarchy is represented as a directed acyclic graph. Two types of relationships between categories exist: "is-a" relation is shown as bold arrows, "part-of" relation is shown as regular arrows. Each term has a unique identifier (GO code).

The hierarchical relationships are represented by indentation and special symbols % (for "is-a" relationships) and < (for "part-of" relationships). For example,

```
%term0
 %term1 % term2
```

means that term1 is a subclass of term0 and also a subclass of term2;

```
%term0
  %term1 < term2 < term3
```

means that term1 is a subclass of term0 and also a part-of of term2 and term3.

Here is an excerpt from file process.ontology:

```
%rhythmic behavior ; GO:0007622
 %circadian rhythm ; GO:0007623
  %circadian sleep/wake cycle ; GO:0042745 % sleep ; GO:0030431
```

```
 <circadian sleep/wake cycle\, non-REM sleep ; GO:0042748
 <circadian sleep/wake cycle\, REM sleep ; GO:0042747
 <circadian sleep/wake cycle\, wakefulness ; GO:0042746
 <regulation of circadian sleep/wake cycle ; GO:0042749
                                  % regulation of sleep ; GO:0045187
%eclosion rhythm ; GO:0008062 < eclosion ; GO:0007562
<entrainment of circadian clock ; GO:0009649
                               % response to photoperiod ; GO:0009648
%locomotor rhythm ; GO:0045475 ; synonym:circadian locomotor
                 activity rhythm % locomotory behavior ; GO:0007626
```

The definitions of all terms are available in a separate file.

Each gene product has one or more molecular functions, is used in one or more biological processes, and might be associated with one or more cellular components. The GO Consortium assigns GO terms to gene products of several organisms, such as yeast (*Saccharomyces cerevisiae*), fly (*Drosophila melanogaster*), worm (*Caenorhabditis elegans*), human, and many others. More organisms are added regularly.

7.2.3 Genomic databases as the source of training data

Several databases, such as SGD, MGD, etc., have information on genes of a particular species. For example, the Saccharomyces Genome Database (SGD) [Dolinski et al., 2003] contains the sequences of yeast genes and proteins; descriptions and classifications of their biological roles, molecular functions, and subcellular localizations (GO codes); and links to literature information. We can use this information on well-studied genes to train a classification system in order to automatically fill in the gaps in annotations for more recently studied genes.

We start with the yeast genome. Yeast (*Saccharomyces cerevisiae*) is a relatively simple, well-studied organism. Its genome has been fully sequenced and contains about 6000 genes. The yeast gene names have been standardized due to the effort by SGD curators. As of Jan. 6, 2005, 6459 yeast gene products have been assigned GO codes with 4865 references included as evidence.

All GO annotations for yeast gene products (for protein or RNA) are contained in file `gene_association.sgd` (see Table 7.4). The `gene_association.sgd` file uses the standard file format for gene association files of the Gene Ontology (GO) Consortium. Each line in this file corresponds to one gene/GO code pair and has the following tab

DB	ID	Symbol	¬	GO ID (*)	Reference (**)	Evidence	With/ From	Aspect	Name	Synonym	Type	Taxon	Date	Assigned
SGD	S0007287	15S_RRNA		GO:0003735	SGD_REF:37735|PMID:6261980	ISS		F		15S_rRNA|15S_RRNA_2	gene	taxon:4932	20030723	SGD
SGD	S0007287	15S_RRNA		GO:0006412	SGD_REF:37736|PMID:6280192	IGI		P		15S_rRNA|15S_RRNA_2	gene	taxon:4932	20030723	SGD
SGD	S0007287	15S_RRNA		GO:0005761	SGD_REF:37735|PMID:6261980	ISS		C		15S_rRNA|15S_RRNA_2	gene	taxon:4932	20030723	SGD
SGD	S0007287	15S_RRNA		GO:0042255	SGD_REF:12699|PMID:2167435	IGI		P		15S_rRNA|15S_RRNA_2	gene	taxon:4932	20030723	SGD
SGD	S0007288	21S_RRNA_3		GO:0042255	SGD_REF:37752|PMID:6759872	IMP		P		21S_rRNA_3	gene	taxon:4932	20030721	SGD
SGD	S0007288	21S_RRNA_3		GO:0003735	SGD_REF:37752|PMID:6759872	IMP		F		21S_rRNA_3	gene	taxon:4932	20030721	SGD
SGD	S0007288	21S_RRNA_3		GO:0003735	SGD_REF:37752|PMID:6759872	ISS		F		21S_rRNA_3	gene	taxon:4932	20030721	SGD
SGD	S0007288	21S_RRNA_3		GO:0005761	SGD_REF:37752|PMID:6759872	IDA		C		21S_rRNA_3	gene	taxon:4932	20030721	SGD
SGD	S0007288	21S_RRNA_3		GO:0006412	SGD_REF:37752|PMID:6759872	IMP		P		21S_rRNA_3	gene	taxon:4932	20030721	SGD
SGD	S0007288	21S_RRNA_3		GO:0006412	SGD_REF:37752|PMID:6759872	ISS		P		21S_rRNA_3	gene	taxon:4932	20030721	SGD
SGD	S0007289	21S_RRNA_4		GO:0042255	SGD_REF:37752|PMID:6759872	IMP		P		21S_rRNA_4	gene	taxon:4932	20030721	SGD
SGD	S0007289	21S_RRNA_4		GO:0003735	SGD_REF:37752|PMID:6759872	IMP		F		21S_rRNA_4	gene	taxon:4932	20030721	SGD
SGD	S0007289	21S_RRNA_4		GO:0003735	SGD_REF:37752|PMID:6759872	ISS		F		21S_rRNA_4	gene	taxon:4932	20030721	SGD
SGD	S0007289	21S_RRNA_4		GO:0005761	SGD_REF:37752|PMID:6759872	IDA		C		21S_rRNA_4	gene	taxon:4932	20030721	SGD
SGD	S0007289	21S_RRNA_4		GO:0006412	SGD_REF:37752|PMID:6759872	IMP		P		21S_rRNA_4	gene	taxon:4932	20030721	SGD
SGD	S0007289	21S_RRNA_4		GO:0006412	SGD_REF:37752|PMID:6759872	ISS		P		21S_rRNA_4	gene	taxon:4932	20030721	SGD
SGD	S0004660	AAC1		GO:0005743	SGD_REF:12031|PMID:2167309	TAS		C	ADP/ATP translocator	YMR056C	gene	taxon:4932	20010118	SGD
SGD	S0004660	AAC1		GO:0006854	SGD_REF:12031|PMID:2167309	IDA		P	ADP/ATP translocator	YMR056C	gene	taxon:4932	20010118	SGD
SGD	S0004660	AAC1		GO:0005471	SGD_REF:12031|PMID:2167309	IDA		F	ADP/ATP translocator	YMR056C	gene	taxon:4932	20010213	SGD
SGD	S0000289	AAC3		GO:0005743	SGD_REF:13606|PMID:1915842	TAS		C	ADP/ATP translocator	YBR085W|ANC3	gene	taxon:4932	20010118	SGD
SGD	S0000289	AAC3		GO:0006854	SGD_REF:13606|PMID:1915842	IMP		P	ADP/ATP translocator	YBR085W|ANC3	gene	taxon:4932	20010118	SGD
SGD	S0000289	AAC3		GO:0005471	SGD_REF:13606|PMID:1915842	IMP		F	ADP/ATP translocator	YBR085W|ANC3	gene	taxon:4932	20010213	SGD
SGD	S0003916	AAD10		GO:0008372	SGD_REF:32926	ND		C	aryl-alcohol dehydrogenase (putative)	YJR155W	gene	taxon:4932	20010119	SGD
SGD	S0003916	AAD10		GO:0018456	SGD_REF:3088|PMID:10572264	ISS		F	aryl-alcohol dehydrogenase (putative)	YJR155W	gene	taxon:4932	20020902	SGD
SGD	S0003916	AAD10		GO:0006081	SGD_REF:3088|PMID:10572264	ISS		P	aryl-alcohol dehydrogenase (putative)	YJR155W	gene	taxon:4932	20020902	SGD
SGD	S0005275	AAD14		GO:0008372	SGD_REF:32926	ND		C	aryl-alcohol dehydrogenase (putative)	YNL331C	gene	taxon:4932	20010119	SGD
SGD	S0005275	AAD14		GO:0018456	SGD_REF:3088|PMID:10572264	ISS		F	aryl-alcohol dehydrogenase (putative)	YNL331C	gene	taxon:4932	20020902	SGD
SGD	S0005275	AAD14		GO:0006081	SGD_REF:3088|PMID:10572264	ISS		P	aryl-alcohol dehydrogenase (putative)	YNL331C	gene	taxon:4932	20020902	SGD
SGD	S0005525	AAD15		GO:0008372	SGD_REF:32926	ND		C	aryl-alcohol dehydrogenase (putative)	YOL165C	gene	taxon:4932	20010119	SGD

Table 7.4: GO annotations for yeast genes (for protein or RNA) contained in file gene_association.sgd (an excerpt). Columns "GO ID (*)" and "Reference (**)" provide the required information to form a training set.

delimited fields:

1. `DB` - database contributing the file (always "SGD" for this file)
2. `DB_Object_ID` - SGDID (the gene ID in the SGD)
3. `DB_Object_Symbol` - a standard gene name (*e.g.* CDC28, COX2) if it has been conferred; a systematic name (*e.g.* YAL001C, YGR116W, YAL034W-A) otherwise
4. `NOT` (optional)- 'NOT' qualifier for a GO annotation, when needed
5. `GO ID` - unique numeric identifier for the GO term, *i.e.* GO code
6. `DB:Reference(|DB:Reference)` - the reference associated with the GO annotation; the first item is the reference ID in the SGD, the second item (in parentheses) is the reference ID in the Medline database
7. `Evidence` - the evidence code for the GO annotation
8. `With (or) From` (optional) - any With or From qualifier for the GO annotation
9. `Aspect` - which ontology the GO term belongs in (F - molecular function, P - biological process, C - cellular component)
10. `DB_Object_Name(|Name)` (optional) - a name for the gene product in words, *e.g.* "acid phosphatase"
11. `DB_Object_Synonym(|Synonym)` (optional) - a systematic name along with any other names, including aliases (but not including the standard name, which will be in Column 3 if one exists) used for the gene
12. `DB_Object_Type` - type of object annotated, *e.g.* gene, protein, *etc.*
13. `taxon(|taxon)` - taxonomic identifier of species encoding gene product
14. `Date` - date GO annotation was made
15. `Assigned_by` - source of the annotation (always "SGD" for this file)

Column 7 provides the information on the kind of evidence that is found in the cited source to support the association between the gene product and the GO term. The possible kinds are

- IMP - inferred from mutant phenotype
- IGI - inferred from genetic interaction
- IPI - inferred from physical interaction
- ISS - inferred from sequence similarity
- IDA - inferred from direct assay
- IEP - inferred from expression pattern
- IEA - inferred from electronic annotation
- TAS - traceable author statement
- NAS - non-traceable author statement
- ND - no biological data available
- IC - inferred by curator

Each line in the `gene_association.sgd` file corresponds to an instance in the categorization task. From this file we collect the IDs of Medline articles associated with the genes and Gene Ontology codes manually assigned to these genes. Using this information we form a training set as described in the next section.

7.2.4 Learning process

The whole process of collecting training data, learning a classifier, and using it to categorize unseen data is summarized below (Figure 7.4).

Learning. We learn a classifier from the information on fully annotated genes in a genomic database (*e.g.* SGD) in the following steps:

1. retrieve the GO codes and the IDs of Medline articles associated with the genes from the database;
2. retrieve the corresponding articles from the Medline library;
3. form a training set consisting of the words from the retrieved articles as features and the corresponding GO codes as categories;

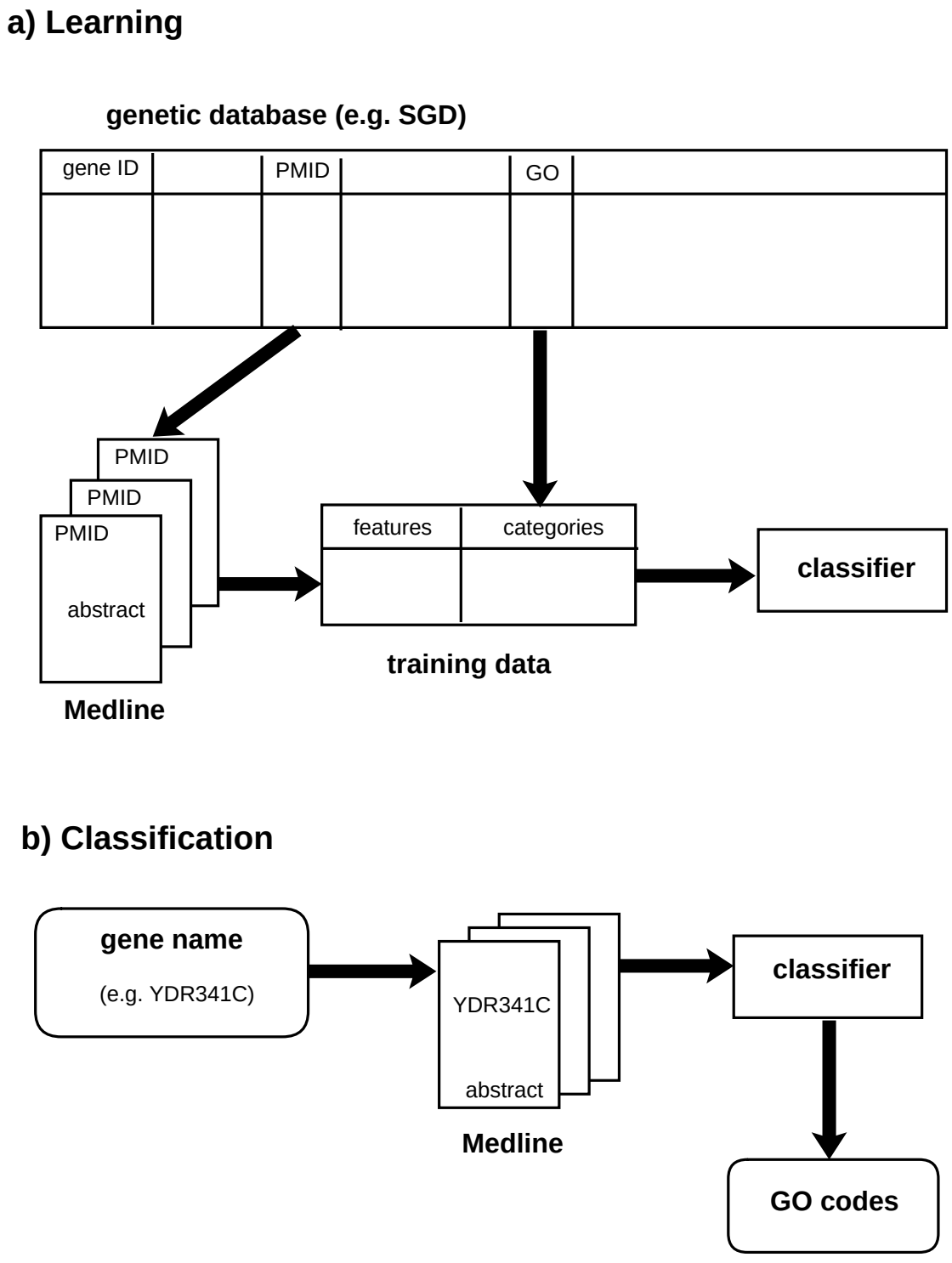


Figure 7.4: Learning (a) and classification (b) processes in automatic functional annotation of genes from biomedical literature.

Gene	Gene Ontology code/term (*)		Medline reference (**)	
YJR096W	GO:0019566	arabinose metabolism	PMID:12271459	Traff, K.L., Jonsson, L.J., Hahn-Hagerdal, B. Putative xylose and arabinose reductases in Saccharomyces cerevisiae. Yeast. 2002 Oct;19(14):1233-41.
YJR096W	GO:0042732	D-xylose metabolism	PMID:12271459	Traff, K.L., Jonsson, L.J., Hahn-Hagerdal, B. Putative xylose and arabinose reductases in Saccharomyces cerevisiae. Yeast. 2002 Oct;19(14):1233-41.
YMR056C	GO:0006854	ATP/ADP exchange	PMID:2167309	Gawaz, M., Douglas, M.G., Klingenberg, M. Structure-function studies of adenine nucleotide transport in mitochondria. II. Biochemical analysis of distinct AAC1 and AAC2 proteins in yeast. J Biol Chem. 1990 Aug 25;265(24):14202-8.
YGR054W	GO:0006413	translational initiation	PMID:12133843	Zoll, W.L., Horton, L.E., Komar, A.A., Hensold, J.O., Merrick, W.C. Characterization of mammalian eIF2A and identification of the yeast homolog. J Biol Chem. 2002 Oct 4; 277(40):37079-87. Epub 2002 Jul 19.
YKR079C	GO:0042779	removal of tRNA 3'-trailer sequence	PMID:12711671	Takaku, H., Minagawa, A., Takagi, M., Nashimoto, M. A candidate prostate cancer susceptibility gene encodes tRNA 3'-processing endoribonuclease. Nucleic Acids Res. 2003 May 1;31(9):2272-8.
...	...	...	...	...
YLL057C	GO:0006790	sulfur metabolism	PMID:10482536	Hogan, D.A., Auchtung, T.A., Hausinger, R.P. Cloning and characterization of a sulfonate/ alpha-ketoglutarate dioxygenase from Saccharomyces cerevisiae. J Bacteriol. 1999 Sep;181(18):5876-9.

Table 7.5: Information on yeast genes from the SGD database. Medline articles with corresponding IDs (**) along with their GO labels (*) form a training set.

4. learn a classifier using hierarchical text categorization techniques.

Classification. After the classifier has been learned, we can apply it to genes that do not have annotations in the database:

1. retrieve all Medline articles that mention the genes (possibly using all their aliases to improve recall);

2. classify the retrieved articles into GO codes with the classifier;

3. assign the most promising codes to the genes (the most promising codes can be identified as the majority codes from article classification or by using more complex techniques involving background knowledge).

Features: bag-of-words from Medline articles (**)	**Categories: Gene Ontology codes (*)**
Putative xylose and arabinose reductases in Saccharomyces cerevisiae. Traff, K.L., Jonsson, L.J., Hahn-Hagerdal B. Saccharomyces cerevisiae mutants, in which open reading frames (ORFs) displaying similarity to the aldo-keto reductase GRE3 gene have been deleted, were investigated regarding their ability to utilize xylose and arabinose. Reduced xylitol formation from D-xylose in gre3 mutants of S. cerevisiae suggests that Gre3p is the major D-xylose-reducing enzyme in S. cerevisiae. ...	GO:0019566, GO:0042732
Structure-function studies of adenine nucleotide transport in mitochondria. II. Biochemical analysis of distinct AAC1 and AAC2 proteins in yeast. Gawaz M., Douglas M.G., Klingenberg, M. AAC1 and AAC2 genes in yeast each encode functional ADP/ATP carrier (AAC) proteins of the mitochondrial inner membrane. In the present study, mitochondria harboring distinct AAC proteins and the pet9 Arg96 to HIS mutant (Lawson, J., Gawaz, M., Klingenberg, M., and Douglas, M. G. (1990) J. Biol. Chem. 265, 14195-14201) protein have been characterized. In addition, properties of the different AAC proteins have been defined following reconstitution into proteoliposomes. ...	GO:0006854
...	...

Table 7.6: Training set formed from the information on yeast genes from the SGD database: Medline articles (**) provide "bag-of-words" features, GO codes (*) provide category labels.

To assemble the data, we exploit the SGD database (Table 7.4) to collect the IDs of Medline articles associated with the yeast genes and their corresponding Gene Ontology codes (Table 7.5). Then, we retrieve the corresponding articles from the Medline library. The abstracts of the articles and manually assigned GO labels form a dataset (Table 7.6). We treat every aspect of the Gene Ontology - biological process (P), molecular function (F), and cellular component (C) - separately, putting together 3 datasets: MEDLINE P, MEDLINE F, and MEDLINE C. Before applying the proposed system in the real-life settings to discover the functionality of poorly characterized genes, we set up a series of experiments to assess the quality of the classification system we can learn with the hierarchical text categorization techniques. For this, the hierarchical approaches are evaluated on annotated genes by splitting the available data into training and test sets with two thirds of data reserved for training and one third for testing. Ten such splits are generated randomly, and the results of the experiments are averaged over 10 trials. The data are preprocessed in the same way as all text collections used in the previous experiments (Chapter 6). The parameters of the three datasets are given in Table 7.7. The collected data are very high-dimensional and sparse, which is a common situation for textual data. As in the previous experiments (Chapter 6), we discard the word stems

dataset	class hierarchy			number of documents			number of attributes
	number of categories	depth	out-degree	total in dataset	training	testing	
MEDLINE P	1,025	12	5.41	3,305	2,309	996	2,793
MEDLINE F	1,078	10	10.29	2,468	1,727	741	2,448
MEDLINE C	331	8	6.45	2,284	1,613	671	2,957

Table 7.7: Characteristics of the MEDLINE data used in the experiments. The number of training and test documents are averaged over 10 trials.

dataset	"flat"			hierarchical local			hierarchical global		
	hP	**hR**	hF_1	**hP**	**hR**	hF_1	**hP**	**hR**	hF_1
MEDLINE P	62.41	8.58	15.06	70.86	50.95	**59.27**	64.61	54.83	**59.31**
MEDLINE F	43.18	4.89	8.78	51.14	37.68	**43.36**	59.01	28.22	38.17
MEDLINE C	79.56	30.65	44.18	78.69	66.50	72.07	77.25	69.84	**73.35**

Table 7.8: Performance of the "flat", hierarchical local, and hierarchical global AdaBoost.MH on the MEDLINE data. Numbers in bold are statistically significantly better with 99% confidence.

that appear in fewer than n abstracts[5] and then rely on the AdaBoost.MH learning algorithm to pick up the most relevant terms to use in classification. 500 iterations of AdaBoost.MH are performed for the "flat" learner, the hierarchical global learner, and each subtask of the hierarchical local learner.

7.2.5 Results

Experiments with both hierarchical learning approaches described in Chapter 4 (with AdaBoost.MH as a base learner) are carried out on the MEDLINE data. In addition, the "flat" version of AdaBoost.MH is run on the same data to investigate the advantages of the hierarchical approaches over the "flat" method on the functional annotation task. The results are presented in Table 7.8.

The first observation we make is that both hierarchical algorithms outperform the

[5]The number n is chosen to keep the data at a manageable size and equals to 6 for the MEDLINE P and the MEDLINE F collections and to 4 for the MEDLINE C dataset.

"flat" method by a large margin on all three ontologies. The absolute differences in performance range from 28% on MEDLINE C dataset to 44% on MEDLINE P dataset. Both hierarchical algorithms are superior to "flat" in recall by ∼23-46%. This is in particular noticeable on the first two datasets, MEDLINE P and MEDLINE F. These datasets are very hard for the "flat" method to deal with having more than a thousand categories and very few training data. As a result, the "flat" method encounters much difficulty in learning reliable models and ends up with extremely low recall values. The hierarchical algorithms, on the other hand, benefit a great deal from having additional training data. The ontologies of such scale (10 and 12 levels deep) allow to populate high-level categories with hundreds of additional labeled examples and, thus, significantly improve recall. Moreover, these additional data result in improved precision as well, which indicates the presence of inherited attributes (see Section 6.1.4). Around two examples per class on average is inadequate amount of training data for such a complex task as functional annotation of genes. Bringing into play examples from lower levels of a hierarchy allows the hierarchical algorithms to explore and take advantage of much relevant vocabulary and, hence, improve precision. The third dataset, MEDLINE C, has a much smaller class hierarchy and more training data per class. It also appears to be easier to learn as the variety of the vocabulary related to cellular localization is much smaller than that of a biological process or a molecular function. As a result, all algorithms, including "flat" show notably better performance on this dataset. Yet again, both hierarchical approaches surpass the "flat" method in recall; though, the "flat" algorithm is superior in precision, which suggests the smaller degree of attribute inheritance.

The two hierarchical approaches exhibit comparable performance on the MEDLINE data. The global technique is inferior to local only on one dataset, MEDLINE F. This dataset is characterized by very large out-degree (10.29) that considerably increases the complexity of the classification task for the global algorithm comparing to the local one. As we saw in the experiments on synthetic data in Chapter 6, greater values of out-degree result in a significant number of additional categories ($\sim k^{d-1}$) for the global method and only a slight (linear) increase in complexity for the local algorithm. Dealing with a thousand categories and vast out-degree, the global algorithm acts cautiously, assigning less labels to test instances and trying to maintain high precision at the cost of lower recall. This becomes apparent on the highly "bushy" MEDLINE F data, where the global algorithm yields to local by ∼5%.

Overall, the Cellular Component ontology looks the easiest among the three ontologies

in the MEDLINE datasets. This, to a great extent, may be due to the largest amount of data available to learning algorithms for training. On average, the MEDLINE C dataset has about 5 training instances per class while the other two collections have only 1-2 examples. For that reason, the first direction for improvement of the proposed system we see in expanding the training data. Yet, this can be tricky. In general, only a few articles per gene get annotated with functional information in most genomic databases. One possible solution may be to combine the functional annotations available in databases from several organisms, such as yeast, fly, mouse, human, etc. Though, it might still be not enough. The functional annotation from literature appears to be a demanding task, so that learning an effective automatic classification system could require a substantial amount of labeled data. In the future, when thousands of new genes get annotated, this problem can be resolved. As for now, this kind of annotation systems can be used in semi-automatic settings by suggesting the functional labels to human annotators and leaving the final decision to trained professionals.

As the next step in this research we plan to apply the proposed approach to novel or poorly characterized genes whose functions have not been included in databases yet. For each such gene we can search the Medline database to retrieve the documents related to the gene and classify those documents into the Gene Ontology codes. The assigned annotations then can be evaluated by an expert who will go through the Medline articles associated with the genes to verify if they contain any evidence for the given annotations. Besides yeast, the same process may be applied to other organisms, fly (*Drosophila melanogaster*), worm (*Caenorhabditis elegans*), mouse (*Mus musculus*), human, whose genomes have also been partially annotated with Gene Ontology functions.

Finally, the feature sets used in the learning process could be enriched with available background knowledge. For example, Medical Subject Headings (MeSH) and Enzyme Commission numbers, which are manually assigned to a large part of the Medline abstracts, can be incorporated as additional features in the conventional "bag-of-words" feature sets. Besides, the same extensions discussed for the MeSH indexing task can be applied here as well.

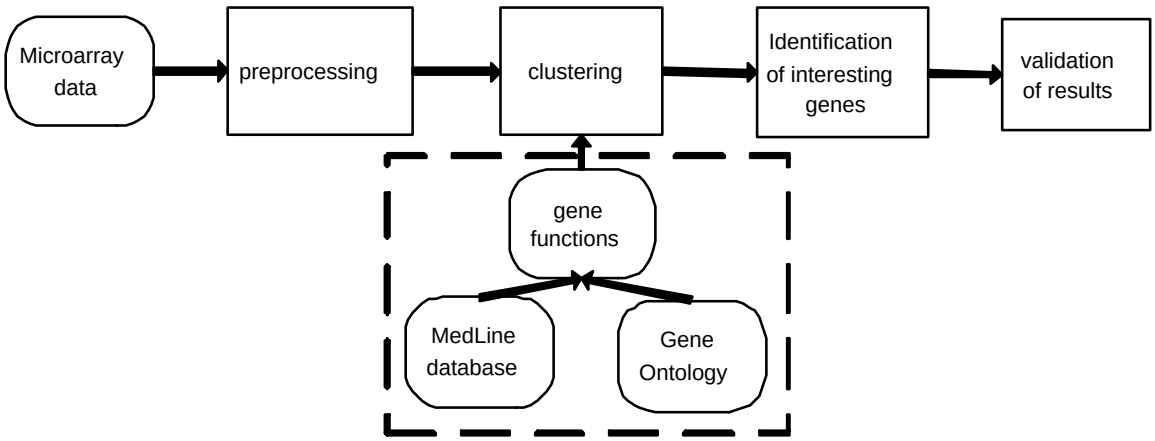


Figure 7.5: Gene expression analysis. Genes are put into a microarray to measure their expression levels. After that, the microarray data are preprocessed and clustered. Several the most interesting groups of genes are identified for follow-up studies. The goal of our task (in the dashed box) is to enhance the clustering process with gene function information to produce more biologically meaningful clusters.

7.3 Gene expression analysis in the presence of background knowledge

One of the major genomics problems that require the functional information on genes/ gene products is gene expression analysis. As mentioned in Section 3.5.6, the microarray technology has revolutionized the field of genomics. Hundreds of microarray studies are conducted on a daily basis. Consequently, a large amount of data need to be analyzed. Clustering techniques are typically the first step in this analysis, and often they are complemented with a manual analysis to identify the functionally coherent groups of genes with interesting expression profiles. Hence, the functional information on genes is essential at this stage. In this work, we would like to go one step further and replace some of the manual effort with automatic techniques that discover functionally and expression-wise related genes directly during the clustering process (see Figure 7.5). This would allow us to obtain clusters that are more meaningful and more practical for biologists and bioinformaticists than the ones produced by conventional clustering techniques.

The idea of combining gene expression data with background functional information has been investigated by several researchers [Hanisch et al., 2002, Raychaudhuri et al., 2003, Glenisson et al., 2003, Liu et al., 2004, Speer et al., 2004b]. Their studies show that the incorporation of background knowledge into the clustering process indeed re-

sults in functionally coherent clusters that are more appealing to biologists. However, in these studies cluster evaluation is mostly done by visual inspection and, hence, is very subjective. In this work, we introduce a practical, fully automatic, unbiased technique to evaluate the quality of functionally enriched clusters. We experiment with our novel hierarchical measure as a candidate for the distance measure between Gene Ontology annotations. Unlike previous work, we also introduce a technique to calculate cluster means, a required step in several clustering algorithms, such as K-means. Our experiments confirm that clustering enhanced with background knowledge results in more biologically meaningful and more valuable groupings of genes.

A natural way of incorporating the functional information into the clustering process is to modify the gene distance function. Normally, the distance function (*e.g.* Euclidean distance, Pearson correlation, etc.) only reflects the similarity of genes based on their expression profiles. To account for functional similarity, we can combine the conventional distance between the expression vectors with the measure of relation between the functional annotations of the genes:

$$D(x, y) = (1 - \alpha) \cdot d_1(x, y) + \alpha \cdot d_2(C_x, C_y),$$

where $d_1(x, y)$ is any conventional distance measure defined on the space of expression vectors, $d_2(C_x, C_y)$ is a distance measure between the functional categories of genes x and y in a Gene Ontology graph, and α is a coefficient representing the relative weight of the functional distance in the combined distance measure. In the present work, we adapt this approach and modify the well-known K-means clustering algorithm to take in the combined distance measure.

7.3.1 K-means clustering algorithm

As the core clustering algorithm we have chosen K-means [Hartigan, 1975]. It is a simple centroid algorithm that has been in use for decades in many applications including gene expression analysis. Despite its simplicity and some drawbacks, the algorithm typically demonstrates excellent performance and, therefore, has been a popular choice in microarray research. In fact, it has been shown to be among the top performing algorithms in gene expression analysis [Gibbons and Roth, 2002]. For that reason, we decided to employ this algorithm in our experiments.

The algorithm aims to partition a set of data points into k disjoint clusters so as to minimize the sum of intra-cluster distances, *i.e.* the distances between each point and its

cluster mean:

$$total\_d = \sum_{j=1}^{k} \sum_{x \in S_j} D(x, \mu_j),$$

where S_j, $j = 1, \ldots, k$ are k non-overlapping clusters, $\mu_j = \frac{\sum_{x \in S_j} x}{|S_j|}$ are the cluster means (aka centroids), and $D(x, \mu_j)$ is the distance between a point and a cluster mean, *e.g.* Euclidean distance.

The algorithm requires a user-defined parameter k that specifies the number of clusters to look for. It starts with (randomly) chosen k points to be the initial cluster centroids. Then, each point is assigned to the cluster with the nearest centroid, and the centroids are re-calculated. After that, the partitions are optimized iteratively by checking for every point whether it should be moved to another cluster in order to minimize the overall intra-cluster distance *total_d*. If a point is moved, the centroids of both clusters, new and old, are re-calculated. This process is repeated until no points are moved from their clusters. The pseudo-code for the algorithm is presented in Figure 7.6.

The K-means algorithm has two major drawbacks. The first one is the need to *a priori* choose the optimal number of clusters k. Sometimes, researchers have an idea of how many natural clusters present in their data, but most of the time, they can only guess. Hence, the optimal number is determined experimentally by trying several possible values for k. The second drawback is the non-deterministic nature of the algorithm. The optimization procedure of the K-means algorithm (provably) converges, but only to a local minimum. Thus, the final partition heavily depends on initially selected centroids. Consequently, if the initial centroids are picked randomly, different runs of the algorithm on the same data would lead to different results. Two alternative solutions can be thought of here: one can either try to find good starting points or run the algorithm with randomly chosen initial centroids several times picking the run that resulted in the smallest intra-cluster distance *total_d*. For our experiments, we follow the latter strategy and execute the clustering algorithm 10 times for each configuration reporting the run with the smallest *total_d*.

7.3.2 K-means enriched with functional information

In this section, we present the modified version of the K-means clustering algorithm that groups genes not only by their expression profiles, but also by their functionality in a cell. For this, we can directly employ our hierarchical evaluation measure as a distance measure between the functional annotations of a pair of genes and add this distance to

Given: $x \in S$ - a set of data points
k - the number of clusters

randomly choose k points $x_{i_j} \in S$ as centroids $\mu_j,\ j = 1, \ldots, k$

for each $x \in S$ do:
$x \in S_i : i = argmin_j\ D(x, \mu_j)$

for $j = 1$ to k do:
$\mu_j = \frac{\sum_{x \in S_j} x}{|S_j|}$

$convergence = false$

repeat until $convergence$:
$convergence = true$
for each $x \in S_i$, $i = 1, \ldots, k$ do:
$i' = argmin_j\ D(x, \mu_j)$
if $(i \neq i')$ then
$convergence = false$
$x \in S_{i'}$
$\mu_i = \frac{\sum_{y \in S_i} y}{|S_i|}$
$\mu_{i'} = \frac{\sum_{y \in S_{i'}} y}{|S_{i'}|}$

return S_j, $j = 1, \ldots, k$

Figure 7.6: K-means clustering algorithm.

a conventional distance (*e.g.* Euclidean) between the gene expression vectors:

$$D(x, y) = (1 - \alpha) \cdot D^E(x, y) + \alpha \cdot D^{GO}(C_x, C_y).$$

Then, we can calculate the distance between a gene and a cluster as the average distance between the given gene and all the genes in the cluster. However, this would require significant computational resources as at each iteration we have to consider distances between every pair of genes[6]. Therefore, we decided to stick with the K-means routine and propose a method of computing cluster centroids and distances between a gene and a centroid in the functional space of the Gene Ontology.

Cluster means are computed separately for expression data and functional annotations. For numeric data (expression values) we use the conventional formula for a

[6]Although the distances between every pair of genes can be precomputed, the calculation of the distance between a gene and a cluster would still be time-consuming as it would involve the averaging of several hundreds of numbers.

geometric centroid:

$$\mu_j^E = \frac{\sum_{x \in S_j} x}{|S_j|}.$$

For nominal data (GO annotations) we apply the following strategy. First, to account for hierarchical relations among functional annotations, we extend a set of functional categories C_x for each gene $x \in S$ with all their ancestor categories in the Gene Ontology $\hat{C}_x = \{\bigcup_{c_k \in C_x} Ancestors(c_k)\}$. Then, the functional part of a cluster centroid μ_j^{GO} is obtained as a set of all functional categories $c_i \in \hat{C}_x$ of all genes x in the cluster S_j: $\mu_j^{GO} = \{\bigcup_{x \in S_j} \hat{C}_x\}$. For each category $c_i \in \mu_j^{GO}$ we also maintain a numerical weight $w_j^{GO}(c_i)$ that corresponds to the number of genes in the cluster S_j having this function: $w_j^{GO}(c_i) = |S_j(c_i)|$, where $S_j(c_i) = \{x : x \in S_j \wedge c_i \in \hat{C}_x\}$. When a new gene x is added to a cluster, the cluster mean is updated by including the functional categories of gene x (and all their ancestor categories) that are not yet present in the cluster and increasing the weights of the gene's categories that have already been in the cluster. Similarly, a cluster mean is updated if a gene is removed from the cluster by decreasing the weights of the gene's functional categories.

The distance between a gene and a cluster mean also consists of two parts: a conventional distance between expression values and a functional distance between annotations in the Gene Ontology space:

$$D(x, \mu_j) = (1 - \alpha) \cdot D^E(x, \mu_j^E) + \alpha \cdot D^{GO}(C_x, \mu_j^{GO}).$$

As a conventional distance between two expression vectors, we use Euclidean distance:

$$D^E(x, \mu_j^E) = \sqrt{\sum_i (x_i - \mu_{ji}^E)^2}.$$

The distance is normalized to be in the same range as the GO distance [0..1]. As a functional distance $D^{GO}(C_x, \mu_j^{GO})$ we adapt our hierarchical measure. We calculate hierarchical precision and hierarchical recall of how well the cluster mean μ_j^{GO} represents gene x's functionality. In other words, we interpret gene x's functional categories as true labels while the categories from the cluster mean μ_j^{GO} form the predicted label set (with weights). Then, precision is calculated as the total weight of gene's functions (and all their ancestor categories) present in the cluster centroid divided by the total weight of

all categories in the cluster:

$$hP_j(x) = \frac{\sum_{c_i \in \hat{C}_x} w_j^{GO}(c_i)}{\sum_{c_k \in \mu_j^{GO}} w_j^{GO}(c_k)}.$$

Positive weights of $w_j^{GO}(c_i)$ represent the fact that the corresponding functions $c_i \in \hat{C}_x$ are present in the cluster. The gene's functions not found in the cluster would have zero weights: $w_j^{GO}(c_i) = 0$, if $c_i \in \hat{C}_x$ and $c_i \notin \mu_j^{GO}$.

Recall is calculated as the percentage of gene's functions (and all their ancestor categories) present in the cluster mean:

$$hR_j(x) = \frac{|\hat{C}_x \cap \mu_j^{GO}|}{|\hat{C}_x|}.$$

Combining the two values, precision and recall, in hF_1 value and subtracting it from 1 we get the functional distance $D^{GO}(C_x, \mu_j^{GO})$:

$$D^{GO}(C_x, \mu_j^{GO}) = 1 - \frac{2 \cdot hP_j(x) \cdot hR_j(x)}{hP_j(x) + hR_j(x)}.$$

7.3.3 Evaluation

Evaluation of clustering algorithms is not a trivial task as there is no notion of clustering accuracy. Generally, the ideal partitioning of data (or gold standard) is not known in advance. So, we can only compare the results obtained by different clustering techniques. A visual inspection is often performed for this task. Yet, even if it is done by an expert, such evaluation is very subjective. An objective evaluation has been the focus of several studies, and some criteria have been proposed, for example cluster compactness [Davies and Bouldin, 1979, Rousseeuw, 1987] and cluster stability [Famili et al., 2004]. Although these are good criteria for clustering just expression data, they are less useful when functionally oriented clusters are the target. A cluster quality measure that takes into account both the cluster compactness and functional coherence is needed for such tasks. However, it is not enough. An ideal measure is the one that can estimate how *useful* the clusters are for a biological study.

In this work, we propose one such measure that is based on the ability of the clusters to predict the functional categories of genes with missing annotations. Since one of the main purposes of clustering is to predict the functions of novel genes, such evaluation

would demonstrate the usefulness of the obtained clusters.

We evaluate cluster prediction performance in a 10-fold cross-validation fashion. A given set of genes, participating in a microarray study, is split into 10 subsets of approximately equal sizes. Then, each subset in turn is used as a test set in the prediction task. The other 9 subsets form a training set that is clustered with the modified K-means algorithm. The obtained clusters are used to predict the functional categories of the genes from the test subset. For this, the genes from the test subset are assigned to clusters based on their expression profiles. Then, the degree of correspondence between the genes' annotations and their clusters' annotations is established with our hierarchical evaluation measure introduced in Chapter 5.

In the described evaluation procedure, two points remain to be defined: how to determine the cluster membership for a non-annotated gene and which categories should be assigned to a gene from a cluster. Clearly, a gene should be assigned to the cluster which it resembles the most or, in other words, the nearest cluster. However, the modified distance measure is ineffective for this task because of the lack of the functional information on a tested gene. Euclidean distance is also inadequate since functionally enriched clusters can overlap in the Euclidean space. The solution we adopt is based on the n-nearest neighbor principle: n nearest points in the Euclidean space are identified and the cluster of the majority of these points is assigned to the tested gene. In this way, we select a cluster with a strong representation in the area of interest, *i.e.* where the tested point lies.

After a cluster is chosen, the tested gene will be annotated with the cluster's functional representation. Large clusters, however, can hold genes with a number of different functions so that assigning all the cluster's functional categories would result in very low precision. For that reason, we apply a pruning procedure to the category assignment process: we assign only those categories that are represented by at least half of the genes in the cluster. The 50% threshold is chosen as a reasonable trade-off between precision and recall.

Finally, we want to emphasize that gene function prediction is not the primary goal of these experiments and is used only as an evaluation tool.

7.3.4 Datasets

We experiment with a well-known dataset of gene expression in budding yeast (*Saccharomyces cerevisiae*) [Eisen et al., 1998]. This dataset contains the expressions of 2467

annotated genes from several experimental conditions, namely the diauxic shift, the mitotic cell division cycle, sporulation, and temperature and reducing shocks. There is a total of 79 individual array experiments. In the original paper [Eisen et al., 1998], the data were hierarchically structured with a pairwise average-linkage clustering technique. The genes' relationships were determined purely based on the similarity of their expression profiles. Then, 10 best clusters were identified manually that represent functionally coherent groups of genes. We collected the genes from these 10 clusters to form a smaller dataset to experiment with. We call this dataset 10-cluster subset of the original data. It consists of 262 genes and all 79 conditions. This dataset is an excellent starting point for our experiments because of its small size, the known number of clusters to look for, and low noise. We use this dataset to study the behavior of the proposed clustering approach with different parameter settings, such as the value of alpha, the number of neighbors in cluster identification for a test point, and the aspect of the Gene Ontology in functional annotations. Then, we run the experiments on the full data to confirm the results.

7.3.5 Results

We run the extended K-means clustering algorithm on the 10-cluster subset of the original yeast expression data separately for each aspect of the Gene Ontology, biological process, molecular function, and cellular component. As the dataset is small, only one nearest neighbor is used to determine the cluster membership for genes in a test set[7]. For each aspect, 10 runs of the 10-fold cross-validation experiments are performed. The results are presented in Figure 7.7. The figure shows the prediction performance of the obtained clusters for different values of the alpha coefficient. The alpha coefficient represents the weight of the GO distance in the combined distance measure. Alpha equal to 0 corresponds to the original K-means with Euclidean distance and is shown separately as a baseline. Another baseline, at the top, shows the prediction performance of the 10 manually chosen clusters in the original study by Eisen et al. [Eisen et al., 1998]. The prediction performance of the manually chosen clusters is assessed in the same way as it is done for automatically derived clusters (Section 7.3.3).

The experiments demonstrate an improved quality of functionally enriched clusters in terms of their prediction capabilities over the conventional Euclidean clusters on all three hierarchies. The most notable improvement is on the molecular function ontology where the differences are statistically significant with 99% confidence for almost all val-

[7]Larger numbers of neighbors have been tried, but did not result in better prediction performance.

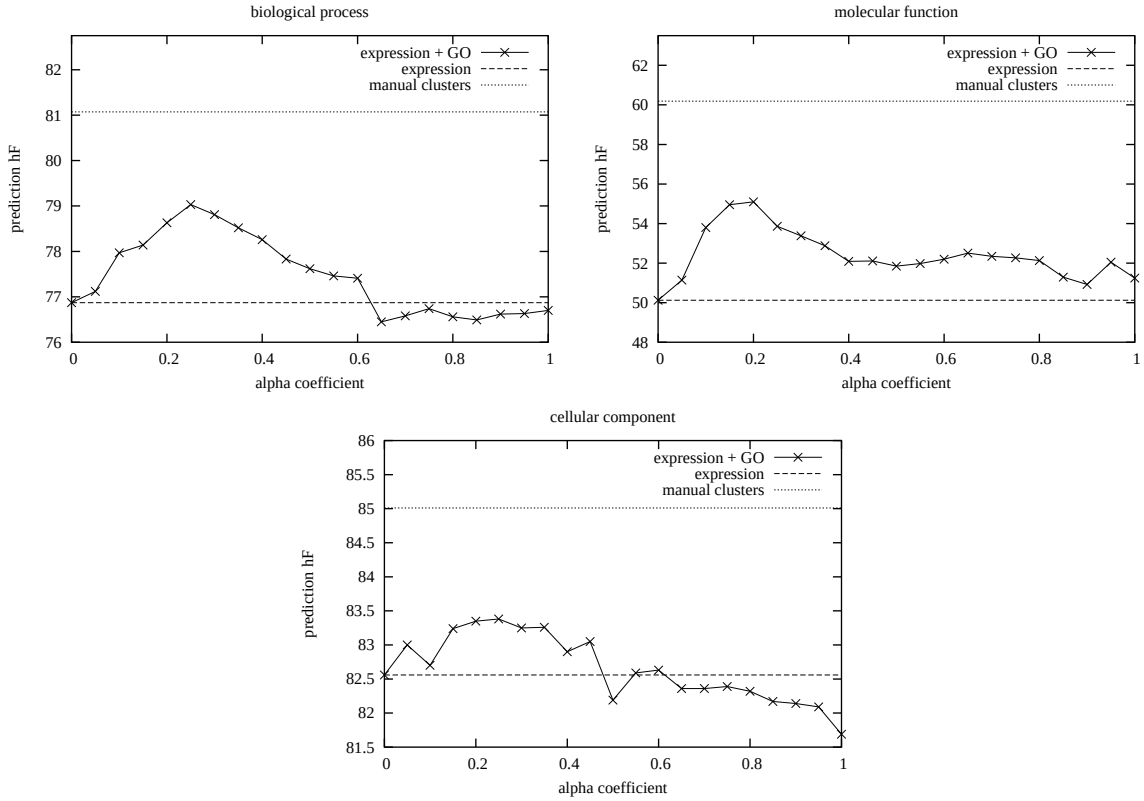


Figure 7.7: Regular and functionally enhanced K-means clustering on the 10-cluster subset of the yeast expression data [Eisen et al., 1998]. Clustering results are evaluated on the function prediction task in a 10-fold cross-validation fashion.

ues of alpha. The maximal gain of 5% is obtained for $\alpha = 0.2$. Then, the extent of the improvement slowly decreases, yet staying positive on all range of alphas. On the biological process hierarchy the differences for most alpha values are positive (with statistical significance for alphas 0.2 - 0.4). The highest gain is obtained for $\alpha = 0.25$. The smaller gain is reached on the cellular component ontology where better prediction is demonstrated only for alpha less than 0.5 (with statistical significance for alphas 0.15 - 0.25). An interesting observation is that on all three ontologies the best results are achieved for approximately the same value of alpha, 0.2 - 0.25. For larger values, the performance starts to slightly deteriorate. Looking at precision-recall breakdown, we can see that the extended clustering algorithm shows considerably improved recall with some loss in precision, which is explained by an improved functional coherence of clusters. In the original K-means algorithm, the clusters are functionally quite diverse so that only the top, highly used functional categories are shared among the large number of genes in each

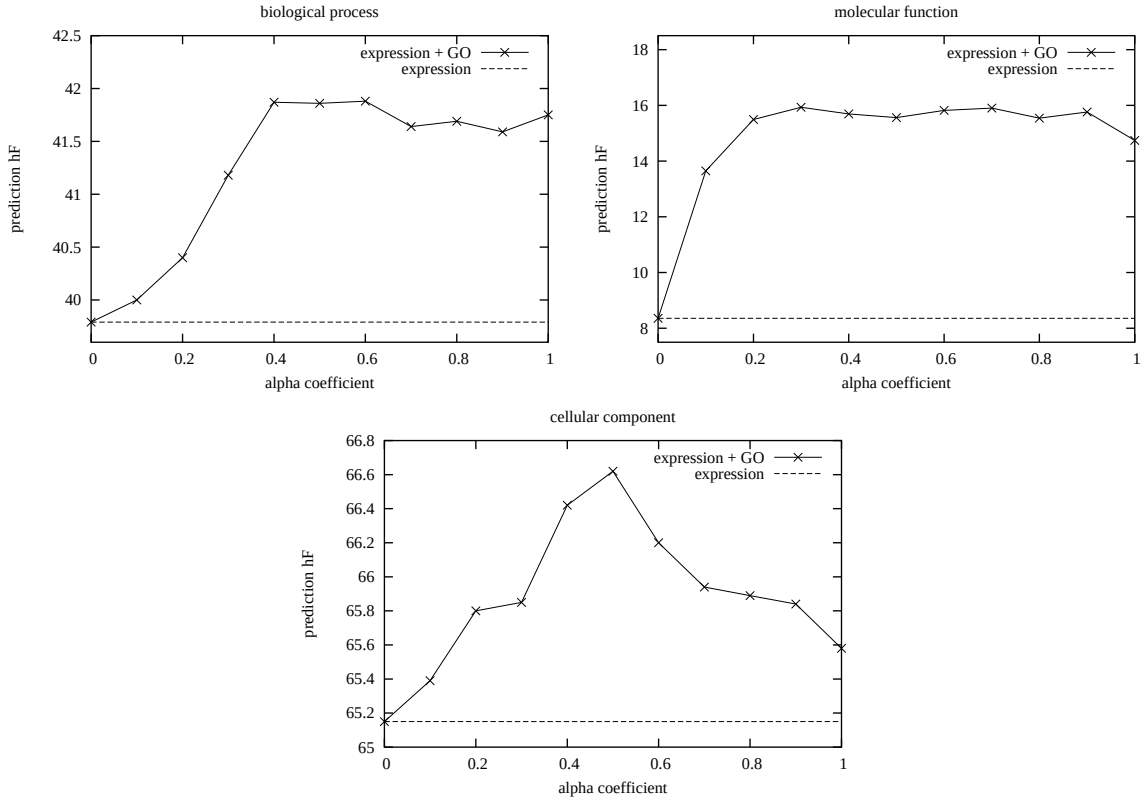


Figure 7.8: Regular and functionally enhanced K-means clustering on the full yeast expression data [Eisen et al., 1998]. Clustering results are evaluated on the function prediction task in a 10-fold cross-validation fashion.

cluster. Therefore, the use of the standard K-means for this task results in highly precise, but hardly useful automatic annotation. The enhanced version of K-means groups genes with shared functionality resulting in more functionally compact clusters where many functional categories are shared among the large number of genes. Thus, more categories are assigned to test genes, which corresponds to significantly higher recall values.

We repeat the experiments on the full version of the dataset. Since the optimal number of clusters for these data is not known a priori, we have tried several values from 5 to 25. Figure 7.8 shows the performance of the extended K-means clustering algorithm with $k = 10$. On these large datasets, the number of nearest neighbors for determining the cluster membership for genes in a test set is increased to 15. The results appear to be better than the ones obtained on the 10-cluster subset with the extended algorithm significantly outperforming the conventional K-means for all values of alpha on all three ontologies. Again, the most noticeable improvement ($\sim 8\%$) is achieved on the molecular

function ontology. On the biological process hierarchy a gain in $\sim 2\%$ is obtained, while on the cellular component ontology the gain is the smallest ($\sim 1.5\%$). Unlike the small dataset, on the full data the peak of the performance is reached for larger values of alpha: $\alpha \approx 0.5$.

Overall, the experiments demonstrate that functional information does indeed enrich the clustering process producing more functionally coherent and, therefore, more biologically meaningful clusters. The contribution of the functional information into the distance function should to be set at about 50% to give enough weight to the functional information while still preserving the cluster compactness in the Euclidean (expression) space at least to a some degree. The cluster prediction ability improves considerably for the molecular function aspect of gene functionality while the cellular component aspect seems to benefit the least. This is expected since a protein's location does not directly correspond to the protein's function, so proteins located in the same part of a cell are not necessarily there for the same reason and, therefore, can have different expression profiles.

An interesting next step will be to combine the functional information from all three aspects of the Gene Ontology: biological process, molecular function, and cellular component. For this, we would calculate the functional distances between a gene and a cluster in each Gene Ontology graph separately and then aggregate those distances into one value. For example, we can take the maximum of the three values. In this way, we would cluster genes that have similar expression profiles and share a biological function, which can be explained by different factors: the genes can be involved in the same pathway, may be located in the same cellular component, etc.

In addition, several aspects of the presented approach, including the cluster mean calculation, the GO distance, and the prediction mechanism, can be improved in the future. Also more experiments will be conducted to see if the conclusions hold on other data and for other clustering techniques and distance measures. In particular, the proposed techniques will be applied to datasets containing poorly characterized genes to evaluate the effectiveness of the approach in the real-world settings.

7.4 Summary

This chapter presents three applications of the hierarchical text categorization techniques to the area of bioinformatics. The three practical problems that we address include article indexing with Medical Subject Headings (MeSH), functional annotation of genes

from biomedical literature, and gene expression analysis in the presence of background knowledge. Our experiments demonstrate that the proposed hierarchical learning and evaluation techniques can be successfully applied to these tasks showing superior results over the conventional "flat" techniques. In our third application, gene expression analysis in the presence of background knowledge, we present a novel technique of co-clustering gene expression (experimental) data with gene functional information (background knowledge), which results in biologically meaningful, practical clusters of genes. Furthermore, we introduce an innovative cluster quality evaluation procedure that assesses not only how good the clusters are, but also how useful they are for a particular task of predicting functionality of poorly characterized genes.

Chapter 8

Conclusions and future work

This work addresses the task of hierarchical text categorization. In this task we are given a set of predefined categories that are organized in a hierarchical structure. The goal of hierarchical text categorization is to efficiently and effectively incorporate the additional information on category structure into the learning process.

The research presented in this thesis focuses on two aspects of hierarchical categorization: learning and performance evaluation. We argue that hierarchical classification should be consistent with a given class hierarchy to fully reproduce the semantics of hierarchical relations. Consequently, consistent classification results in more meaningful and easily interpretable output for end-users. Then, we present two learning algorithms that carry out consistent classification. The first one is a local top-down approach that has been extended to the general case of DAG hierarchies with internal class assignments. The second algorithm is a novel hierarchical global approach. In addition, we perform an extensive set of experiments on real and synthetic data to demonstrate that the two hierarchical techniques significantly outperform the corresponding "flat" approach, i.e. the approach that does not take into account any hierarchical information.

The second main contribution of this research is the new hierarchical performance evaluation measure. After discussing performance measures for hierarchical classification and introducing natural, desired properties that these measures ought to satisfy, we define a novel hierarchical evaluation measure, and show that, unlike the conventional "flat" as well as the existing hierarchical measures, the new measure satisfies all the desired properties. It is also simple, requires no parameter tuning, and has much discriminating power. Moreover, it is superior to standard "flat" measures in terms of statistical consistency and discriminancy, the two concepts introduced by Huang and Ling [Huang and

Ling, 2005] to systematically compare classifier performance measures.

Also, our work illustrates the benefits of the proposed hierarchical text categorization techniques over conventional "flat" classification on real-world applications from bioinformatics. Bioinformatics is a vital, quickly developing scientific discipline that has many text-related problems. A wealth of biomedical literature accumulated through decades presents a valuable source of essential knowledge required by biomedical scientists and practitioners in their everyday activities. While manual search through such a vast collection of free texts is tedious and time consuming, automatic text categorization methods offer users the means of fast and reliable search and retrieval of the requisite information.

In this work, we address three bioinformatics problems. The objective of the first task, indexing biomedical articles with Medical Subject Headings (MeSH), is to associate documents with biomedical concepts they discuss from the specialized vocabulary of MeSH. Having articles indexed with MeSH terms considerably improves the performance of search engines, such as Pubmed. In our second application, we tackle a challenging problem of gene functional annotation from biomedical literature. We view this task as a text categorization problem of classifying the articles describing a gene of interest into functional categories of the Gene Ontology. Our experiments demonstrate the advantage of hierarchical text categorization techniques over the "flat" method on this task. In the third application, our goal is to enrich the analysis of plain experimental data with biological knowledge. In particular, we incorporate the functional information on genes, available from the specialized genomic databases or from literature, directly into the clustering process of microarray data. This results in improved biological relevance and value of clustering results.

In future work, we plan to extend the proposed global hierarchical learning method to other base learning algorithms. Unlike AdaBoost.MH, some multi-label classification methods may be found behaving consistently in the hierarchical framework even without the post-processing step. Also, we would like to investigate the relationship between the performance of a hierarchical classifier and the number of training examples required to attain that level of performance. In machine learning research, this is a well-known issue that can be addressed in a theoretical framework (*e.g.* PAC-learning) or in an experimental setting. Since the hierarchical problem and the learning process considerably differ from the standard "flat" case, this question needs to be examined in the context of the hierarchical learning task.

For the bioinformatics applications, MeSH indexing and gene functional annotation, our primary goal is obtaining more training data. As well-known both theoretically and

practically, extra training data often result in a better classification performance. The addressed applications deal with hundreds of classes and tremendously diverse biomedical vocabulary; therefore, they require a substantial amount of labeled data to learn reliable classification models and to achieve results that can be practical in real-life settings. In addition, biology oriented text problems can benefit from more sophisticated feature selection and construction approaches. Besides conventional "bag of words" features, we can include additional information on gene aliases, MeSH terms, and/or Enzyme Commission numbers associated with the documents. Moreover, special representation of biologically relevant named entities, *e.g.* gene names, chemical names, etc., can have a positive effect on classification performance.

In our third application, gene expression analysis with functional information, we would like to continue experiments and explore other possible distance measures (*e.g.* Pearson correlation) and clustering algorithms (*e.g.* SOM, hierarchical algorithms). In addition, several aspects of the presented approach, *e.g.* centroid calculation and the prediction mechanism, can be further refined.

Finally, we will investigate other bioinformatics problems where background textual information can enrich the traditional practices. In particular, biomedical literature can be brought into play at various stages of a biological study: from planning and refining biological experiments to the analysis of the obtained results and validation of the derived conclusions. Text mining techniques, including the presented hierarchical text categorization methods, would be the central means in pursuing these goals.

Appendix A

In this section we present details on other hierarchical approaches explored in this research (see Section 4.3), namely hierarchical decision trees, ECOC, and cost-sensitive learning.

Hierarchical decision trees

To include hierarchical information into the decision tree induction method C4.5[1], we modified the entropy/gain ratio splitting criteria in a number of ways. The general idea was to force the induction algorithm to focus first on high level categories and only later on low level categories in some way simulating top-down level by level hierarchical classification (the local hierarchical approach). We did this by giving more weight to closely located categories in a hierarchy (sibling categories) and less weight to distant categories. Below are listed the ways we approached these objectives and results we got.

1. **Hierarchical precision (Chapter 5) as a splitting criterion**[2][3]. Since we measure the performance of a classification system with our new hierarchical measure, it is a natural choice to try to optimize this measure while building a decision tree. However, as in the regular decision tree induction process optimization of the classification error does not work pretty well, in the hierarchical decision tree

[1] In these experiments we used C4.5 software as well as Weka's implementation of the decision tree learning algorithm.

[2] For single-label classification with examples belonging only to leaf classes, the values of precision (P), recall (R), and F_1-measure are equal. Therefore, we report only standard precision (P) and hierarchical precision (hP).

[3] For experiments in this section we used a small version of the "20 newsgroups" dataset. It consists of 15 leaf and 5 intermediate categories; examples are assigned only to the leaf categories with 50 examples per category. In addition, we used a single-label version of the "Reuters-21578" dataset. We selected examples that belong only to one class. Overall, this dataset had 66 leaf categories and 6 intermediate categories. Each experiment on these datasets was repeated 10 times in a cross-validation fashion.

induction optimization of the hierarchical error did not lead to a significant improvement. The resulting decision trees were in general much larger causing a very small training error rate but about the same test error rate.

	C4.5		modified C4.5	
Dataset	**P**	**hP**	**P**	**hP**
20 newsgroups (small)	50.00	62.60	**50.40**	62.20
20 newsgroups	65.14	74.40	**65.60**	**75.20**

2. **Hierarchical precision as a splitting criterion + feature selection.** Preliminary experiments with synthetic data indicated that the presence of useless features can significantly hurt the performance of the algorithm. Therefore, we combined the algorithm with one of the standard feature selection methods that chooses the top n features according to their info gain, gain ratio, or Chi-square statistics (see Table 3.1). Selecting a small number of the most informative features boosted the performance of the hierarchical algorithm.

			C4.5		modified C4.5	
Dataset	**FS method**	**# of features**	**P**	**hP**	**P**	**hP**
20 newsgroups (small)	info gain	50	45.20	58.67	**45.73**	**59.73**
		100	48.80	61.27	**51.33**	**64.20**
		500	50.53	62.53	**51.87**	**64.13**
		1000	50.00	61.67	**51.47**	**63.00**
	gain ratio	100	40.40	52.40	**41.20**	**53.33**
		500	55.87	66.00	54.13	64.73
	Chi-square	100	52.53	63.80	**53.60**	**65.67**
		500	52.93	64.93	**54.67**	**66.67**

3. **A linear combination of standard gain ratio and hierarchical precision as a splitting criterion.**

$$\alpha \cdot GainRatio + (1 - \alpha) \cdot hP$$

We decided to combine the hierarchical precision splitting criterion with the conventional gain ratio splitting criterion to be able to control the size of a decision

tree. Gain ratio is known to be a good choice to select a small decision tree that satisfies training data. Combining it with our hierarchical criterion led to a slight improvement in the performance.

		modified C4.5	
Dataset	α	**P**	**hP**
20 newsgroups	0.0 (C4.5 with hP)	65.60	75.20
	0.1	65.80	75.21
	0.2	65.22	74.88
	0.3	65.36	75.27
	0.4	65.61	**75.54**
	0.5	**65.91**	75.48
	0.6	65.65	75.27
	0.7	65.01	74.63
	0.8	64.62	74.25
	0.9	64.70	74.25
	1.0 (original C4.5)	65.14	74.40
reuters (single-label)	0.2	**83.80**	**87.00**
	0.3	82.24	85.81
	0.4	82.64	86.23
	0.5	82.65	86.35
	0.6	82.15	85.87
	0.7	82.48	86.14
	0.8	82.23	86.03
	0.9	82.83	86.55
	1.0 (original C4.5)	83.05	86.50

4. **Hierarchical precision as a class probability in the gain ratio splitting criterion.** The class probability is the proportion of the examples that belong to the class out of all examples that reach the current decision node. It also corresponds to the accuracy of the node if we classify all examples into that class. We replace the standard accuracy measure with a hierarchical one changing accordingly the entropy formula:

$$Entropy = -\sum_i hP(c_i) log\ hP(c_i),$$

where $hP(c_i)$ computes the hierarchical precision of the decision node where all examples are classified in c_i.

Dataset	C4.5		modified C4.5	
	P	**hP**	**P**	**hP**
20 newsgroups	67.31	76.67	63.77	75.03
reuters (single-label)	84.32	87.75	81.10	85.41

5. **Modifying the entropy measure to give more weight to siblings**

$$Entropy = -\alpha \cdot \sum_{siblings\ of\ majority\ class} P(c_i) log P(c_i) - (1-\alpha) \cdot \sum_{other\ categories} P(c_j) log P(c_j)$$

Dataset	C4.5		modified C4.5		
	P	**hP**	α	**P**	**hP**
20 newsgroups (small)	50.00	62.6	0.9	48.27	62.47
			0.75	44.13	58.73

6. **Shrinkage (partly counting examples from sibling categories)**

$$P(c_i) = \frac{|c_i| + \alpha \cdot \sum_{siblings\ of\ c_i} |c_j|}{n + n \cdot \alpha \cdot (\#classes - 1)}; \qquad Entropy = -\sum_i P(c_i) log P(c_i)$$

Dataset	C4.5		modified C4.5		
	P	**hP**	α	**P**	**hP**
20 newsgroups (small)	50.00	62.6	0.1	44.00	58.33
			0.5	37.47	52.47

7. **Modifying the entropy measure to incorporate the parents' entropy**

$$Entropy = -\alpha \cdot \sum_{children\ categories} P(c_i) log P(c_i) - (1-\alpha) \cdot \sum_{parent\ categories} P(c_j) log P(c_j)$$

Dataset	α	modified C4.5	
		P	**hP**
20 newsgroups (small)	0.1	39.87	54.27
	0.5	46.67	59.27
	0.6	47.47	60.13
	0.75	48.93	60.60
	0.8	47.47	59.87
	0.9	48.27	60.27
	1.0 (original C4.5)	50.00	62.60

8. **Gain ratio as a splitting criterion + pruning to optimize the hierarchical measure.** Generally, the decision tree induction algorithm consists of two parts. First, a decision tree is grown until leaf nodes contain just a few examples or no more splits can be produced. Then, the decision tree is pruned to reduce its size but keep the training accuracy. Smaller trees are less prone to overfitting leading to better performance on test instances. We keep the growing part of the algorithm the same, but change the pruning part modifying the pruning criterion. Instead of optimizing the standard accuracy we optimize the hierarchical accuracy while pruning the tree.

Dataset	C4.5		modified C4.5	
	P	**hP**	**P**	**hP**
20 newsgroups	67.31	76.67	62.19	73.40
reuters (single-label)	84.32	87.75	**84.88**	**88.19**

9. **Gini index with misclassification costs derived from the class hierarchy as a splitting criterion.** Gini index is an alternative splitting criterion used in another successful decision tree building system CART [Breiman et al., 1984]. It is based on the notion of node impurity:

$$i(t) = \sum_{i \neq j} cost(c_i|c_j) P(c_i|t) P(c_j|t).$$

At each step a node t is splitted into two nodes, left t_L and right t_R, and the best split is the one that maximizes the reduction in impurity:

$$\Delta(s,t) = i(t) - p_L \cdot i(t_L) - p_R \cdot i(t_R),$$

where p_L (p_R) is the proportion of examples that go left (right). Usually, misclassification costs $cost(c_i|c_j)$ are not taken into account. However, a class hierarchy clearly introduces some non-uniform costs. We tried this cost-sensitive formula with the following costs: cost of the correct classification $cost(c_i|c_i)$ is 0, cost of misclassification into a sibling category is 1, cost of misclassification into a distant category varies from 1 to 4.

		modified C4.5	
Dataset	**cost**	**P**	**hP**
20 newsgroups (small)	1	47.47	59.20
	1.5	45.33	58.07
	2	45.47	58.07
	4	41.73	55.87

ECOC

Error-Correcting Codes (ECOC) [Dietterich and Bakiri, 1995] is a learning technique for multi-class classification problems. It works in the following way. Each of the n categories is mapped into a unique k-bit string ($k < n$) called a codeword. Then, k classifiers are trained, one for each bit. When a new instance is classified, a category with the codeword closest to the bit string produced by the classifiers is selected.

To adapt ECOC to the hierarchical settings, we add bits representing the hierarchical information and/or allow more/less separation between siblings than between non-siblings categories. The first conclusion that we got was that less separation between some of the nodes does not reduce the length of the code, so we can't save the computational cost here. The second conclusion was that the classification accuracy, both standard and hierarchical, depends only on row and column bit separation. By adding bits to represent the hierarchical relations between the categories, we increase the row separation but decrease the column separation. The following are the results for the "20 newsgroups" (small) dataset and decision trees as an underlying learning method:

BCH code	**separation between siblings**	**separation between non-siblings**	**P**	**hP**
15 bits BCH code	7 bits	7 bits	46.67	56.33
15-bit BCH code + 15-bit BCH code for parent categories	7 bits	14 bits	46.67	56.67
15-bit BCH code + 15-bit BCH code for children categories	14 bits	7 bits	42.67	53.00
31-bit BCH code	15 bits	15 bits	56.67	65.00

Nevertheless, hierarchical precision can be increased (at the cost of recall) if classification to a child node is replaced with classification to its parent in situations where the code for a test instance is very distant from the code of the child node, *i.e.* the Hamming distance to the child category is greater than a given threshold:

threshold	**hP**
∞	56.33
5	58.68
4	61.31
3	66.39
2	**70.39**
1	69.57
0	68.52

Cost-sensitive learning

We experimented with two cost-sensitive learning approaches: C5.0 and MetaCost. The cost matrix was derived as follows. The cost of the correct classification $cost(c_i, c_i) = 0$, the cost of misclassification into a sibling category equals to 1, and the cost of misclassification into a distant (non-sibling) category varied from 1 to 10. All experiments were run on the "20 newsgroups" (small) dataset in a 10-fold cross-validation fashion.

C5.0 produced the following results:

cost	**P**	**hP**
1	49.20	61.93
2	49.50	61.20
3	**49.70**	**62.13**
5	48.30	60.07
10	48.40	60.80

C5.0's cost-sensitive component is designed to minimize expected misclassification costs:

$$cost(c_i|d) = \sum_j P(c_j|d) \cdot cost(c_i, c_j).$$

In general, a decision tree induction algorithm does not predict class probabilities, but only the classes themselves. However, we can easily produce class probabilities for a given test instance as the proportion of training instances that belong to the class out of all training instances classified into the same node as the test instance. Usually, the leaf nodes of a decision tree contain examples of only a few categories, *i.e.* only a few categories would have probabilities greater than zero for a given instance. Suppose, there are two such categories, which is the most frequent case. Then,

$$cost(c_1|d) = P(c_1|d) \cdot cost(c_1, c_1) + P(c_2|d) \cdot cost(c_1, c_2) = P(c_2|d) \cdot cost(c_1, c_2),$$

$$cost(c_2|d) = P(c_1|d) \cdot cost(c_2, c_1) + P(c_2|d) \cdot cost(c_2, c_2) = P(c_1|d) \cdot cost(c_2, c_1).$$

For most classification tasks with non-uniform costs, the costs are non-symmetrical, *i.e.* $cost(c_1, c_2) \neq cost(c_2, c_1)$, and the algorithm would choose the class that minimizes the overall cost. In the case of hierarchies, the costs are symmetrical, *i.e.* $cost(c_1, c_2) = cost(c_2, c_1)$, and the classification decision in this case will be identical to the decision taken in the case of uniform costs. As a result, the performance of C5.0 with costs does not differ much from its performance with uniform costs.

MetaCost algorithm also aims at minimizing expected misclassification cost. It uses bagging to estimate the class probabilities on training examples, relabels the training examples with the estimated optimal class that minimizes misclassification costs and applies a base learner on the relabeled training set. We applied MetaCost with Naive Bayes as a base learner on the "20 newsgroups" (small) dataset:

algorithm	**P**	**hP**
Naive Bayes	46.13	57.87
MetaCost with Naive Bayes (cost = 2)	27.07	40.00

Overall, our conclusion is that the existing cost-sensitive approaches are not very advantageous for hierarchical categorization unless we apply some smoothing techniques to calibrate predicted class probabilities.

Appendix B

20 newsgroups dataset

computers
- comp.graphics
- comp.os.ms-windows.misc
- comp.sys.ibm.pc.hardware
- comp.sys.mac.hardware
- comp.windows.x

politics
- talk.politics.guns
- talk.politics.mideast
- talk.politics.misc

religion
- alt.atheism
- soc.religion.christian
- talk.religion.misc

sports
- rec.sport.baseball
- rec.sport.hockey

vehicles
- rec.autos
- rec.motorcycles

Reuters dataset

Commodity Codes (69)

- ALUM
- BARLEY
- CARCASS
- CASTOR-OIL
- CASTORSEED
- CITRUSPULP
- COCOA
- COCONUT-OIL
- COCONUT
- COFFEE
- COPPER
- COPRA-CAKE
- CORN-OIL
- CORN
- CORNGLUTENFEED
- COTTON
- COTTON-OIL
- COTTONSEED
- F-CATTLE
- FISHMEAL
- GOLD
- GRAIN
- GROUNDNUT
- GROUNDNUT-OIL
- IRON-STEEL
- LEAD
- LIN-MEAL
- LIN-OIL
- LINSEED
- LIVESTOCK
- L-CATTLE
- HOG

LUMBER
MEAL-FEED
NICKEL
OAT
OILSEED
ORANGE
PALLADIUM
PALM-OIL
PALMKERNEL
PLATINUM
PLYWOOD
PORK-BELLY
POTATO
RAPE-MEAL
RAPE-OIL
RAPESEED
RED-BEAN
RICE
RUBBER
RYE
SILVER
SORGHUM
SOY-MEAL
SOY-OIL
SOYBEAN
STRATEGIC-METAL
SUGAR
SUN-MEAL
SUN-OIL
SUNSEED
TAPIOCA
TEA
TIN
VEG-OIL
WHEAT

- WOOL
- ZINC

Corporate Codes (2)

- Mergers/Acquisitions (ACQ)
- Earnings and Earnings Forecasts (EARN)

Currency Codes (21)

- U.S. Dollar (DLR)
- Australian Dollar (AUSTDLR)
- Hong Kong Dollar (HK)
- New Zealand Dollar (NZDLR)
- Canadian Dollar (CAN)
- Sterling (STG)
- D-Mark (DMK)
- Japanese Yen (YEN)
- Swiss Franc (SFR)
- Belgian Franc (BFR)
- Netherlands Guilder/Florin (DFL)
- Italian Lira (LIT)
- Danish Krone/Crown (DKR)
- Norwegian Krone/Crown (NKR)
- Swedish Krona/Crown (SKR)
- Brazilian Cruzado (CRUZADO)
- Saudi Arabian Riyal (SAUDRIYAL)
- South African Rand (RAND)
- Indonesian Rupiah (RUPIAH)
- Malaysian Ringitt (RINGGIT)
- Spanish Peseta (PESETA)

Economic Indicator Codes (16)

- Balance of Payments (BOP)
- Trade (TRADE)
- Consumer Price Index (CPI)
- Wholesale Price Index (WPI)
- Unemployment (JOBS)
- Industrial Production Index (IPI)

- Capacity Utilization (CPU)
- Gross National/Domestic Product (GNP)
- Money Supply (MONEY-SUPPLY)
- Reserves (RESERVES)
- Leading Economic Indicators (LEI)
- Housing Starts (HOUSING)
- Personal Income (INCOME)
- Inventories (INVENTORIES)
- Installment Debt/Consumer Credit (INSTAL-DEBT)
- Retail Sales (RETAIL)

Energy Codes (9)

- Crude Oil (CRUDE)
- Heating Oil/Gas Oil (HEAT)
- Fuel Oil (FUEL)
- Gasoline (GAS)
- Natural Gas (NAT-GAS)
- Petro-Chemicals (PET-CHEM)
- Propane (PROPANE)
- Jet and Kerosene (JET)
- Naphtha (NAPHTHA)

Others (3)

- Money/Foreign Exchange (MONEY-FX)
- Shipping (SHIP)
- Interest Rates (INTEREST)

Glossary

This appendix provides a glossary of terminology introduced and/or used in this dissertation.

Ancestor set For a category $p \in C$ and a class hierarchy $\mathcal{H} = \langle C, \leq \rangle$, $Ancestors(p) = \{q \in C : q \geq p\}$. (sect. 1.2.1)

Average out-degree of a directed graph The average of the out-degree of all vertices in the graph. (sect. 2.2)

Binary categorization task A categorization task with two classes: $|C| = 2$. (sect. 1.1.1)

Bioinformatics An interdisciplinary area at the intersection of biological, computer, and information sciences necessary to manage, process, and understand large amounts of biological data.

Child category Category $q \in C$ is a child category of class $p \in C$ in a class hierarchy $\mathcal{H} = \langle C, \leq \rangle$, if $q < p$ and $\nexists r \in C : q < r < p$. (sect. 1.2.1)

Consistency of evaluation measures There exists no pair of classification results $a, b \in \Psi$, which the two measure f, g on domain Ψ evaluate differently: $f(a) > f(b)$ and $g(a) < g(b)$. (sect. 5.4)

Controlled vocabulary An established list of standardized terminology for use in indexing and retrieval of information. A controlled vocabulary ensures that a subject will be described using the same preferred term each time it is indexed and this will make it easier to find all information about a specific topic during the search process *[National Library of Canada]*. (sect. 2.1)

Degree of consistency of evaluation measures The ratio $\frac{|R|}{|R|+|S|}$, where $R = \{(a,b)|$ $a,b \in \Psi, f(a) > f(b), g(a) > g(b)\}$, $S = \{(a,b)|a,b \in \Psi, f(a) > f(b), g(a) < g(b)\}$ for two measures f, g on domain Ψ. (sect. 5.4)

Degree of discriminancy of evaluation measures The degree of discriminancy of measure f over measure g on domain Ψ is the ratio $\frac{|P|}{|Q|}$, where $P = \{(a,b)|a,b \in \Psi, f(a) > f(b), g(a) = g(b)\}$ and $Q = \{(a,b)|a,b \in \Psi, g(a) > g(b), f(a) = f(b)\}$. (sect. 5.4)

Depth of a directed acyclic graph The maximal depth over all vertices in the graph. (sect. 2.2)

Depth (or level) of a vertex in a DAG The length of the shortest path from the root to the vertex. (sect. 2.2)

Directed acyclic graph (DAG) A graph where each edge has a direction and there are no cycles. (sect. 2.2)

Discriminancy of evaluation measures Measure f is more discriminating than measure g on domain Ψ if there exist $a,b \in \Psi$ such that $f(a) > f(b)$ and $g(a) = g(b)$, and there exist no $a,b \in \Psi$ such that $g(a) > g(b)$ and $f(a) = f(b)$. (sect. 5.4)

Distance between categories The length of the shortest (undirected) path from one category to another in a hierarchical graph. (sect. 3.3)

F-measure $F_\beta = \frac{(\beta^2+1)\cdot P \cdot R}{(\beta^2 \cdot P + R)}, \beta \in [0, +\infty)$, where P denotes precision and R denotes recall. (sect. 3.3)

Finite partially ordered set (poset) A structure $\mathcal{H} = \langle C, \leq \rangle$, where C is a finite set and $\leq \subseteq C \times C$ is a reflexive, anti-symmetric, transitive binary relation on C *[Joslyn, 2004]*. (sect. 1.2.1)

"Flat" learning algorithm A standard, non-hierarchical learning algorithm applied to the "flat" set of all categories (internal and leaves) from a given class hierarchy.

Gene Ontology An ontology of gene/gene product functionalities developed by the Gene Ontology Consortium (http://www.geneontology.org). (sect. 7.2.2)

Genomics A new scientific discipline that studies genes and their functions. It is characterized by high-throughput genome-wide experimental approaches combined with

statistical and computational techniques of bioinformatics for the analysis of the results.

Graph A pair (V, E), where V is a set of vertices (aka nodes), and E is a set of edges between the vertices $E = \{(u, v)|u, v \in V\}$. (sect. 2.2)

Hierarchical consistency Any label set $C_i \in C$ assigned to an instance $d_i \in D$ includes complete ancestor sets for every label $c_k \in C_i$, *i.e.* if $c_k \in C_i$ and $c_j \in Ancestors(c_k)$, then $c_j \in C_i$. (sect. 2.3)

Hierarchical F-measure The F-measure calculated on the true and predicted category sets extended with the corresponding ancestor classes. (sect. 5.2)

Hierarchical global feature selection Selection of relevant features for all categories in a given hierarchy simultaneously. (sect. 3.1)

Hierarchical global learning algorithm A learning algorithm that builds only one classifier to discriminate all categories in a hierarchy. (sect. 3.2)

Hierarchical local feature selection Selection of relevant features for each subproblem, corresponding to an internal node of a class hierarchy, separately. (sect. 3.1)

Hierarchical local learning algorithm A learning algorithm that builds separate classifiers for internal nodes of a class hierarchy. (sect. 3.2)

Hierarchical precision Precision calculated on the true and predicted category sets extended with the corresponding ancestor classes. (sect. 5.2)

Hierarchical recall Recall calculated on the true and predicted category sets extended with the corresponding ancestor classes. (sect. 5.2)

Hierarchical text categorization task A text categorization task with a given poset structure $\mathcal{H} = \langle C, \leq \rangle$ on category set C. (sect. 1.2.1)

Hierarchy A set of predefined categories C with a given partially ordered structure on it $\mathcal{H} = \langle C, \leq \rangle$. (sect. 1.2.1)

Internal (intermediate) category A category that has both parent and children classes. (sect. 1.2.1)

Leaf category A category that has no children classes. (sect. 1.2.1)

Length of the path in a directed graph The number of edges traversed in the path. (sect. 2.2)

Macroaveraging Averaging individual categories' performances (*e.g.* precision/recall) over all categories. (sect. 3.3)

Medical Subject Headings (MeSH) A controlled vocabulary of the National Library of Medicine (NLM) (http://www.nlm.nih.gov/mesh/meshhome.html). It consists of specialized terminology used for indexing, cataloging, and searching for biomedical and health-related information and documents. (sect. 7.1.2)

Medline The online public database of biomedical literature at the National Library of Medicine (NLM) (http://www.ncbi.nih.gov/entrez/query.fcgi), developed by the National Center for Biotechnology Information (NCBI) at the National Institutes of Health (NIH).

Microaveraging Summing individual cells in contingency matrices (*i.e.* TP, FP, FN, TN) and then calculating the performance measure (*e.g.* precision/recall) for a global matrix. (sect. 3.3)

Multi-class categorization task A categorization task with more than two classes: $|C| > 2$. (sect. 1.1.1)

Multi-label categorization task A categorization task where each document can be assigned to any number of categories from 0 to $|C|$. (sect. 1.1.1)

Offspring set For a category $p \in C$ and a class hierarchy $\mathcal{H} = \langle C, \leq \rangle$, $Offspring(p) = \{q \in C : q \leq p\}$. (sect. 1.2.1)

Ontology The hierarchical structuring of knowledge using a set of concepts that are specified in order to create an agreed-upon vocabulary for exchanging information. *[National Library of Canada]* (sect. 2.1)

Out-degree of a vertex The number of edges initiated in the vertex. (sect. 2.2)

Parent category Category $p \in C$ is a parent category of class $q \in C$ in a class hierarchy $\mathcal{H} = \langle C, \leq \rangle$, if $q < p$ and $\nexists r \in C : q < r < p$. (sect. 1.2.1)

Path in a directed graph A list of vertices of the graph where each vertex has an edge from it to the next vertex. (sect. 2.2)

Precision The percentage of correctly classified documents out of all documents classified to a category. (sect. 3.3)

Recall The percentage of correctly classified documents out of all documents in a category. (sect. 3.3)

Root (top) category An ancestor of all classes in the hierarchy: $\{Root(\mathcal{H})\} = \bigcap_{p \in C} Ancestors(p)$. (sect. 1.2.1)

Single-label categorization task A categorization task where each document must be assigned to exactly one category. (sect. 1.1.1)

Statistical consistency of evaluation measures The degree of consistency of two measures is greater than 0.5. (sect. 5.4)

Statistical discriminancy of evaluation measures Measure f is statistically more discriminating than measure g on domain Ψ, if the degree of discriminancy for f over g is greater than 1. (sect. 5.4)

Taxonomy A form of hierarchy representing a controlled vocabulary. (sect. 2.1)

Text categorization task The task of assigning a Boolean value to each pair $\langle d_j, c_i \rangle \in D \times C$, where D is a domain of documents and $C = \left\{c_1, \ldots, c_{|C|}\right\}$ is a set of predefined categories. *[Sebastiani, 2002]* (sect. 1.1.1)

Thesaurus An advanced controlled vocabulary that gives not only the relationships between the terms in the form of a hierarchy (narrower-broader terms), but also related and preferable terms. (sect. 2.1)

Tree A directed acyclic graph where each vertex (except the root) has exactly one parent. (sect. 2.2)

Bibliography

[Agrawal et al., 2000] Agrawal, R., Jr., R. B., and Srikant, R. (2000). Athena: Mining-Based Interactive Management of Text Database. In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, pages 365–379.

[Andrade and Bork, 2000] Andrade, M. and Bork, P. (2000). Automated Extraction of Information in Molecular Biology. *FEBS Letters*, 476:12–17.

[Ashburner et al., 2000] Ashburner, M. et al. (2000). Gene Ontology: Tool for the Unification of Biology. *Nature Genetics*, 25(1):25–29.

[Blaschke et al., 1999] Blaschke, C., Andrade, M., Ouzounis, C., and Valencia, A. (1999). Automatic Extraction of Biological Information from Scientific Text: Protein-Protein Interactions. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 60–67.

[Blockeel et al., 2002] Blockeel, H., Bruynooghe, M., Dzeroski, S., Ramon, J., and Struyf, J. (2002). Hierarchical Multi-Classification. In *Proceedings of the SIGKDD Workshop on Multi-Relational Data Mining (MRDM)*, pages 21–35.

[Bolshakova et al., 2005] Bolshakova, N., Azuaje, F., and Cunningham, P. (2005). A Knowledge-Driven Approach to Cluster Validity Assessment. *Bioinformatics*, 21(10):2546–2547.

[Breiman et al., 1984] Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and Regression Trees*. Wadsworth, Belmont, Ca.

[Brown et al., 1999] Brown, M., Grundy, W., Lin, D., Christianini, N., Sugnet, C., Jr, M., and Haussler, D. (1999). Support Vector Machine Classification of Microarray Gene Expression Data. Technical report, University California Santa Cruz.

[Bunescu et al., 2003] Bunescu, R. et al. (2003). Learning to Extract Proteins and their Interactions from MedLine abstracts. In *Proceedings of the ICML Workshop on Machine Learning in Bioinformatics*.

[Burhans et al., 2003] Burhans, D., Campbell, A., and Skuse, G. (2003). Exploring the Role of Knowledge Representation and Reasoning in Biomedical Text Understanding. In *Proceedings of the SIGIR Workshop on Text Analysis and Search for Bioinformatics*.

[Burke, 2003] Burke, W. (2003). Genomics as a Probe for Disease Biology. *New England Journal of Medicine*, 349(10):969–972.

[Cai and Hofmann, 2004] Cai, L. and Hofmann, T. (2004). Hierarchical Document Categorization with Support Vector Machines. In *Proceedings of the ACM Conference on Information and Knowledge Management*, pages 78–87.

[Caropreso et al., 2001] Caropreso, M., Matwin, S., and Sebastiani, F. (2001). A Learner-Independent Evaluation of the Usefulness of Statistical Phrases for Automated Text Categorization. In Chin, A., editor, *Text Databases and Document Management: Theory and Practice*, pages 343–354. Idea Group Publishing.

[Catona et al., 2004] Catona, E., Srinivasan, P., and Street, W. N. (2004). Protein Annotation with GO Codes. In *Proceedings of the MEDINFO 2004*.

[Chakrabarti et al., 1997] Chakrabarti, S., Dom, B., Agrawal, R., and Raghavan, P. (1997). Using Taxonomy, Discriminants, and Signatures for Navigating in Text Databases. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*.

[Chen et al., 2002] Chen, C., Chen, M., and Sun, Y. (2002). PVA: A Self-Adaptive Personal View Agent. *Journal of Intelligent Information Systems*, 18(2/3):173–194.

[Chen and Dumais, 2000] Chen, H. and Dumais, S. (2000). Bringing Order to the Web: Automatically Categorizing Search Results. In *Proceedings of the Conference on Human Factors in Computing Systems (CHI)*, pages 145–152.

[Cheng et al., 2001] Cheng, C., Tang, J., Wai-chee, A., and King, I. (2001). Hierarchical Classification of Documents with Error Control. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD)*, pages 433–443.

[Chiang and Yu, 2004] Chiang, J.-H. and Yu, H.-C. (2004). Extracting Functional Annotations of Proteins Based on Hybrid Text Mining Approaches. In *Proceedings of the BioCreAtIvE Challenge Evaluation Workshop*.

[Chuang et al., 2000] Chuang, W., Tiyyagura, A., Yang, J., and Giuffrida, G. (2000). A Fast Algorithm for Hierarchical Text Classification. In *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery (DaWaK)*, pages 409–418.

[Clare, 2003] Clare, A. (2003). *Machine Learning and Data Mining for Yeast Functional Genomics*. PhD thesis, University of Wales.

[Collier et al., 2000] Collier, N., Nobata, C., and Tsujii, J. (2000). Extracting the Names of Genes and Gene Products with a Hidden Markov Model. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 201–207.

[Consortium, 2002] Consortium, F. (2002). The Flybase Database of the Drosophila Genome Projects and Community Literature. *Nucleic Acids Research*, 30:106–108.

[Cooper and Miller, 1998] Cooper, G. and Miller, R. (1998). An Experiment Comparing Lexical and Statistical Methods for Extracting MeSH Terms from Clinical Free Text. *Journal of the American Medical Association*, 5(1):62–75.

[Cooper, 2003] Cooper, J. (2003). An Evaluation of Unnamed Relations Computation for Discovery of Protein-Protein Interactions. In *Proceedings of the SIGIR Workshop on Text Analysis and Search for Bioinformatics*.

[Couto et al., 2005] Couto, F., Silva, M., and Coutinho, P. (2005). Finding Genomic Ontology Terms in Text using Evidence Content. *BMC Bioinformatics*, 6 (Supplement 1).

[Crammer and Singer, 2001] Crammer, K. and Singer, Y. (2001). On the Algorithmic Implementation of Multi-Class Kernel-based Vector Machines. *Journal of Machine Learning Research*, 2:265–292.

[Craven and Kumlien, 1999] Craven, M. and Kumlien, J. (1999). Constructing Biological Knowledge Bases by Extracting Information from Text Sources. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 77–86.

[D'Alessio et al., 2000] D'Alessio, S., Murray, K., R.Schiaffino, and Kershenbaum, A. (2000). The Effect of Using Hierarchical Classifiers in Text Categorization. In *Proceeding of the International Conference Recherche d'Information Assistee par Ordinateur (RIAO)*, pages 302–313.

[Davies and Bouldin, 1979] Davies, J. and Bouldin, D. (1979). A Cluster Separation Measure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1:224–227.

[de Bruijn and Martin, 2002] de Bruijn, B. and Martin, J. (2002). Getting to the (C)ore of Knowledge: Mining Biomedical Literature. *Int. Journal of Medical Informatics*, 67:7–18.

[Dekel et al., 2004] Dekel, O., Keshet, J., and Singer, Y. (2004). Large Margin Hierarchical Classification. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 209–216.

[Dempster et al., 1977] Dempster, A., Laird, N., and Rubin, D. (1977). Maximum Likelihood from Incomplete Data via the EM. *Journal of the Royal Statistical Society, Series B*, 39:1–38.

[Dietterich, 1997] Dietterich, T. (1997). Machine Learning Research: Four Current Directions. *AI Magazine*, 18(4):97–136.

[Dietterich and Bakiri, 1995] Dietterich, T. and Bakiri, G. (1995). Solving Multiclass Learning Problems via Error-Correcting Output Codes. *Journal of Artificial Intelligence Research*, 2:263–286.

[Do and Poulet, 2003] Do, T.-N. and Poulet, F. (2003). Incremental SVM and Visualization Tools for Bio-Medical Data Mining. In *Proceedings of the European Workshop on Data Mining and Text Mining for Bioinformatics*, pages 14–19.

[Dobrokhotov et al., 2003] Dobrokhotov, P., Goutte, C., Veuthey, A.-L., and Gaussier, E. (2003). Combining NLP and Probabilistic Categorization for Document and Term Selection for Swiss-Prot Medical Annotation. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 91–94.

[Dolinski et al., 2003] Dolinski, K. et al. (2003). Saccharomyces Genome Database. http://www.yeastgenome.org/.

[Domingos, 1999] Domingos, P. (1999). MetaCost: A General Method for Making Classifiers Cost-sensitive. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining*, pages 155–164.

[Dumais and Chen, 2000] Dumais, S. and Chen, H. (2000). Hierarchical Classification of Web Content. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 256–263.

[ECC, 1992] ECC (1992). Enzyme Nomenclature. Academic Press, San Diego, California.

[Ehrler et al., 2005] Ehrler, F., Geissbuhler, A., Jimeno, A., and Ruch, P. (2005). Data-poor Categorization and Passage Retrieval for Gene Ontology Annotation in Swiss-Prot. *BMC Bioinformatics*, 6 (Supplement 1).

[Eisen et al., 1998] Eisen, M., Spellman, P., Brown, P., and Botstein, D. (1998). Cluster Analysis and Display of Genome-wide Expression Patterns. *Proceedings of the National Academy of Sciences*, 95:14863–14868.

[Famili et al., 2004] Famili, A., Liu, G., and Liu, Z. (2004). Evaluation and Optimization of Clustering in Gene Expression Data Analysis. *Bioinformatics*, 20(10):1535–1545.

[Freund and Schapire, 1996] Freund, Y. and Schapire, R. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning*, pages 148–156.

[Friedman et al., 2000] Friedman, J., Hastie, T., and Tibshirani, R. (2000). Additive Logistic Regression: a Statistical View of Boosting. *Annals of Statistics*, 2:337–407.

[Frommholz, 2001] Frommholz, I. (2001). Categorizing Web Documents in Hierarchical Catalogues. In *Proceedings of the European Colloquium on Information Retrieval Research*.

[Fukuda et al., 1998] Fukuda, K., Tsunoda, T., Tamura, A., and Takagi, T. (1998). Toward Information Extraction: Identifying Protein Names from Biological Papers. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 707–718.

[Furey et al., 2000] Furey, T., Duffy, N., Cristianini, N., Bednarski, D., Schummer, M., and Haussler, D. (2000). Support Vector Machine Classification and Validation of Can-

cer Tissue Samples Using Microarray Expression Data. *Bioinformatics*, 16(10):906–914.

[Gaussier et al., 2002] Gaussier, E., Goutte, C., Popat, K., and Chen, F. (2002). A Hierarchical Model for Clustering and Categorising Documents. In *Proceedings of the European Colloquium in IR Research (ECIR)*, pages 229–247.

[Ghani, 2000] Ghani, R. (2000). Using Error-Correcting Codes for Text Classification. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[Gibbons and Roth, 2002] Gibbons, F. and Roth, F. (2002). Judging the Quality of Gene Expression-Based Clustering Methods Using Gene Annotation. *Genome Research*, 12:1574–1581.

[Glenisson et al., 2003] Glenisson, P., Mathys, J., and De Moor, B. (2003). Meta-Clustering of Gene Expression Data and Literature-Extracted Information. *ACM SIGKDD Explorations, Special Issue on Microarray Data Mining*, 5(2).

[Golub et al., 1999] Golub, T., Slonim, D., Tamayo, P., Huard, C., Gaasenbeek, M., Mesirov, J., Coller, H., Loh, M., Downing, J., Caligiuri, M., Bloomfield, C., and Lander, E. (1999). Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring. (286):531–537.

[Goutte and Gaussier, 2005] Goutte, C. and Gaussier, E. (2005). A Probabilistic Interpretation of Precision, Recall and F-score, with Implication for Evaluation. In *Proceedings of the 27th European Conference on Information Retrieval*, pages 345–359.

[Greiner et al., 1997] Greiner, R., Grove, A., and Schuurmans, D. (1997). On Learning Hierarchical Classifications.

[Gruber, 1993] Gruber, T. (1993). A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, 5(2):199–220.

[Guttmacher and Collins, 2002] Guttmacher, A. and Collins, F. (2002). Genomic Medicine A Primer. *New England Journal of Medicine*, 347(19):1512–1521.

[Hanisch et al., 2002] Hanisch, D., Zien, A., Zimmer, R., and Lengauer, T. (2002). Co-clustering of Biological Networks and Gene Expression Data. *Bioinformatics*, 18 (Supplement 1):S145–S154.

[Hartigan, 1975] Hartigan, J. (1975). *Clustering Algorithms*. John Wiley & Sons.

[Hersh and Bhupatiraju, 2003] Hersh, W. and Bhupatiraju, R. (2003). Of Mice and Men (and Rats and Fruit Flies): The TREC Genomics Track. In *Proceedings of the SIGIR Workshop on Text Analysis and Search for Bioinformatics*.

[Hersh et al., 2004] Hersh, W., Bhupatiraju, R., Ross, L., Johnson, P., Cohen, A., and Kraemer, D. (2004). TREC 2004 Genomics Track Overview. In *Proceedings of the TREC 2004 Conference*.

[Hersh et al., 1994] Hersh, W., Buckley, C., Leone, T., and Hickman, D. (1994). OHSUMED: an Interactive Retrieval Evaluation and New Large Text Collection for Research. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 192–201.

[Hersh et al., 2005] Hersh, W., Cohen, A., Yang, J., Bhupatiraju, R., Roberts, P., and Hearst, M. (2005). TREC 2005 Genomics Track Overview. In *Proceedings of the TREC 2005 Conference*.

[Hettich et al., 1998] Hettich, S., Blake, C., and Merz, C. (1998). UCI repository of machine learning databases.

[Huang and Ling, 2005] Huang, J. and Ling, C. (2005). Using AUC and Accuracy in Evaluating Learning Algorithms. *IEEE Trans. on Data and Knowledge Engineering*, 17(3)(3):299–310.

[Hull, 1994] Hull, D. (1994). Improving Text Retrieval for the Routing Problem Using Latent Semantic Indexing. In *Proceedings of the 17th ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 282–289.

[Hvidsten et al., 2003] Hvidsten, T., Laegreid, A., and Komorowski, J. (2003). Learning Rule-based Models of Biological Process from Gene Expression Time Profiles Using Gene Ontology. *Bioinformatics*, 19(9):1116–1123.

[ICD-10, 1992] ICD-10 (1992). *International Statistical Classification of Diseases and Related Health Problems*. World Health Organization, Geneva, 1989 revision edition.

[Ipeirotis et al., 2001] Ipeirotis, P., Gravano, L., and Sahami, M. (2001). Probe, Count, and Classify: Categorizing Hidden Web Databases. In *Proceedings of the ACM SIGMOD Conference*.

[Itskevitch, 2001] Itskevitch, J. (2001). Automatic Hierarchical Email Classification Using Association Rules. Master's thesis, Simon Fraser University.

[Jenssen et al., 2001] Jenssen, T.-K., Laegreid, A., Komorowski, J., and Hovig, E. (2001). A Literature Network of Human Genes for High-throughput Analysis of Gene Expression. *Nature Genetics*, 28:21–28.

[Jiang and Conrath, 1998] Jiang, J. and Conrath, D. (1998). semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *Proceedings of International Conference on Research in Computational Linguistics*.

[Joslyn, 2004] Joslyn, C. (2004). Poset Ontologies and Concept Lattices as Semantic Hierarchies. In *Proceedings of the 12th International Conference on Conceptual Structures (ICCS)*, pages 287–302.

[Joslyn et al., 2005] Joslyn, C., Cohn, J., Verspoor, K., and Mniszewski, S. (2005). Automatic Ontological Function Annotation: Towards a Common Methodological Framework. In *Proceedings of the 8th Annual Bio-Ontologies Meeting held at ISMB-05*.

[Karp et al., 1999] Karp, P., Riley, M., Paley, S., Pellegrini-Toole, A., and Krummenacker, M. (1999). EcoCyc: Encyclopedia of Escherichia Coli Genes and Metabolism. *Nucleic Acids Research*, 27:55–58.

[Kim et al., 2001] Kim, W., Aronson, A., and Wilbur, W. (2001). Automatic MeSH Term Assignment and Quality Assessment. In *Proceedings of the AMIA 2001 Annual Symposium*.

[King et al., 2003] King, O., Foulger, R., Dwight, S., White, J., and Roth, F. (2003). Predicting Gene Function from Patterns of Annotation. *Genome Research*, 13:896–904.

[Kiritchenko et al., 2004] Kiritchenko, S., Matwin, S., and Famili, A. (2004). Hierarchical Text Categorization as a Tool of Associating Genes with Gene Ontology Codes. In *Proceedings. of the Second European Workshop on Data Mining and Text Mining for Bioinformatics*, pages 26–30.

[Kiritchenko et al., 2005a] Kiritchenko, S., Matwin, S., and Famili, A. (2005a). Functional Annotation of Genes Using Hierarchical Text Categorization. In *Proceedings of the BioLINK SIG: Linking Literature, Information and Knowledge for Biology*.

[Kiritchenko et al., 2005b] Kiritchenko, S., Matwin, S., Nock, R., and Famili, A. (2005b). Learning and Evaluation in the Presence of Class Hierarchies: Application to Text Categorization. *Submitted*.

[Koller and Sahami, 1997] Koller, D. and Sahami, M. (1997). Hierarchically Classifying Documents Using Very Few Words. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 170–178.

[Krallinger et al., 2005] Krallinger, M., Padron, M., and Valencia, A. (2005). A Sentence Sliding Window Approach to Extract Protein Annotations from Biomedical Articles. *BMC Bioinformatics*, 6 (Supplement 1).

[Krymolowski et al., 2004] Krymolowski, Y., Alex, B., and Leidner, J. (2004). BioCreative Task 2.1: The Edinburgh-Stanford System. In *Proceedings of the BioCreAtIvE Challenge Evaluation Workshop*.

[Lang, 1995] Lang, K. (1995). NewsWeeder: Learning to Filter Netnews. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 331–339.

[Lewis, 1992] Lewis, D. (1992). An Evaluation of Phrasal and Clustered Representation on a Text Categorization Task. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 37–50.

[Lewis et al., 2004] Lewis, D., Yang, Y., Rose, T., and Li, F. (2004). RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5:361–397.

[Li and Roth, 2002] Li, X. and Roth, D. (2002). Learning Question Classifiers. In *Proceedings of the International Conference on Computational Linguistics (COLING)*.

[Lin, 1998] Lin, D. (1998). An Information-Theoretic Definition of Similarity. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 296–304.

[Lin et al., 2002] Lin, S.-H., Chen, M., Ho, J.-M., and Huang, Y.-M. (2002). ACIRD: Intelligent Internet Document Organization and Retrieval. *Knowledge and Data Engineering*, 14(3):599–614.

[Liu et al., 2004] Liu, J., Wang, W., and Yang, J. (2004). Gene Ontology Friendly Biclustering of Expression Profiles. In *Proceedings of the IEEE Computational Systems Bioinformatics Conference (CSB)*.

[Long and Vega, 2003] Long, P. and Vega, V. (2003). Boosting and Microarray Data. *Machine Learning Journal*, 52(1-2):31–44.

[Lord et al., 2003] Lord, P., Stevens, R., Brass, A., and Goble, C. (2003). Investigating Semantic Similarity Measures Across the Gene Ontology: the Relationship between Sequence and Annotation. 19(10):1275–1283.

[Maron, 1961] Maron, M. (1961). Automatic Indexing: an Experimental Inquiry. *Journal of the Association for Computing Machinery*, 8(3):404–417.

[Masys et al., 2001] Masys, D., Welsh, J., Fink, J., Gribskov, M., Klacansky, I., and Corbeil, J. (2001). Use of Keyword Hierarchies to Interpret Gene Expression Patterns. *Bioinformatics*, 17(4):319–326.

[McCallum, 1999] McCallum, A. (1999). Multi-label Text Classification with a Mixture Model Trained by EM. In *Proceedings of the AAAI Workshop on Text Learning*.

[McCallum et al., 1998] McCallum, A., Rosenfeld, R., Mitchell, T., and Ng, A. (1998). Improving Text Classification by Shrinkage in a Hierarchy of Classes. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 359–367.

[McKusick, 1994] McKusick, V. (1994). *Mendelian Inheritance in Man, Catalog of Human Genes and Genetic Disorders*. Johns Hopkins University Press.

[Mewes et al., 1997] Mewes, H., Albermann, K., Bahr, M., Frishman, D., Gleissner, A., Hani, J., Heumann, K., Kleine, K., Maierl, A., Oliver, S., Pfeiffer, F., and Zollner, A. (1997). Overview of the Yeast Genome. *Nature*, 387:7–8.

[Mitchell, 1998] Mitchell, T. (1998). Conditions for the Equivalence of Hierarchical and Flat Bayesian Classifiers. Technical report, Center for Automated Learning and Discovery, Carnegie-Mellon University.

[Mladenic and Grobelnik, 1998] Mladenic, D. and Grobelnik, M. (1998). Feature Selection for Classification Based on Text Hierarchy. In *Working notes of Learning from Text and the Web, Conference on Automated Learning and Discovery (CONALD)*.

[Nédellec et al., 2001] Nédellec, C., Vetah, M., and Bessières, P. (2001). Sentence Filtering for Information Extraction in Genomics, a Classification Problem. In *Proceedings of the 5th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 326–337.

[Ng et al., 1997] Ng, H., Goh, W., and Low, K. (1997). Feature Selection, Perceptron Learning, and a Usability Case Study for Text Categorization. In *Proceedings of the ACM International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 67–73.

[Nobata et al., 1999] Nobata, C., Collier, N., and Tsujii, J. (1999). Automatic Term Identification and Classification in Biology Texts. In *Proceedings of the Natural Language Pacific Rim Symposium (NLPRS)*, pages 369–374.

[Ogata et al., 1999] Ogata, H., Goto, S., Fujibuchi, W., Bono, H., and Kanehisa, M. (1999). KEGG: Kyoto Encyclopedia of Genes and Genomes. *Nucleic Acids Research*, 27:29–34.

[Ohta et al., 1997] Ohta, Y., Yamamoto, Y., Okazaki, T., Uchiyama, I., and Takagi, T. (1997). Automatic Construction of Knowledge Base from Biological Papers. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 218–225.

[Porter, 1980] Porter, M. (1980). An Algorithm for Suffix Stripping. *Program*, 14(3):130–137.

[Pretschner and Gauch, 1999] Pretschner, A. and Gauch, S. (1999). Ontology Based Personalized Search. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 391–398.

[Proux et al., 1998] Proux, D., Rechenmann, F., Julliard, L., Pillet, V., and Jacq, B. (1998). Detecting Gene Symbols and Names in Biological Texts: A First Step Toward Pertinent Information. In *Proceedings of the Ninth Workshop on Genome Informatics*, pages 72–80.

[Proux et al., 2000] Proux, D., Rechenmann, F., and Laurent, J. (2000). A Pragmatic Information Extraction Strategy for Gathering Data on Genetic Interactions. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 279–285.

[Quinlan, 1993] Quinlan, J. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

[Ray and Craven, 2005] Ray, S. and Craven, M. (2005). Learning Statistical Models for Annotating Proteins with Function Information using Biomedical Text. *BMC Bioinformatics*, 6 (Supplement 1).

[Raychaudhuri and Altman, 2003] Raychaudhuri, S. and Altman, R. (2003). A Literature-based Method for Assessing the Functional Coherence of a Gene Group. *Bioinformatics*, 19(3):396–401.

[Raychaudhuri et al., 2002] Raychaudhuri, S., Chang, J., Sutphin, P., and Altman, R. (2002). Associating Genes with Gene Ontology Codes Using a Maximum Entropy Analysis of Biomedical Literature. *Genome Research*, 12:203–214.

[Raychaudhuri et al., 2003] Raychaudhuri, S., Schütze, H., and Altman, R. (2003). Inclusion of Textual Documentation in the Analysis of Multidimensional Data Sets: Application to Gene Expression Data. *Machine Learning*, 52:119–145.

[Renner and Aszódi, 2000] Renner, A. and Aszódi, A. (2000). High-throughput Functional Annotation of Novel Gene Products Using Document Clustering. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 54–68.

[Resnik, 1995] Resnik, P. (1995). Using Information Content to Evaluate Semantic Similarity in a Texonomy. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 448–453.

[Resnik and Yarowsky, 1997] Resnik, P. and Yarowsky, D. (1997). A Perspective on Word Sense Disambiguation Methods and their Evaluation. In *Proceedings of the ACL SIGLEX Workshop on Tagging Text with Lexical Semantics: Why, What, and How?*

[Rice et al., 2005] Rice, S., Nenadic, G., and Stapley, B. (2005). Mining Protein Function from Text using Term-based Support Vector Machines. *BMC Bioinformatics*, 6 (Supplement 1).

[Riley, 1998] Riley, M. (1998). Genes and Proteins of Escherichia Coli K-12 (GenProtEC). *Nucleic Acids Research*, 26:54.

[Rison et al., 2000] Rison, S., Hodgman, T., and Thornton, J. (2000). Comparison of Functional Annotation Schemes for Genomes. *Functional and Integrative Genomics*, 1:56–69.

[Rousseeuw, 1987] Rousseeuw, P. (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. *Journal of Computational Application in Mathematics*, 20:53–65.

[Ruiz and Srinivasan, 2002] Ruiz, M. and Srinivasan, P. (2002). Hierarchical Text Categorization Using Neural Networks. *Information Retrieval*, 5:87–118.

[Schapire et al., 1998] Schapire, R., Freund, Y., Bartlett, P., and Lee, W. (1998). Boosting the Margin: a New Explanation for the Effectiveness of Voting Methods. *Annals of Statistics*, 26:1651–1686.

[Schapire and Singer, 1999] Schapire, R. and Singer, Y. (1999). Improved Boosting Algorithms Using Confidence-rated Predictions. *Machine Learning*, 37:297–336.

[Schapire and Singer, 2000] Schapire, R. and Singer, Y. (2000). BoosTexter: A Boosting-based System for Text Categorization. *Machine Learning*, 39(2/3):135–168.

[Sebastiani, 2002] Sebastiani, F. (2002). Machine Learning in Automated Text Categorization. *ACM Computing Surveys (CSUR)*, 34(1):1–47.

[Sehgal et al., 2003] Sehgal, A., Qui, X., and Srinivasan, P. (2003). Mining MEDLINE Metadata to Explore Genes and their Connections. In *Proceedings of the SIGIR Workshop on Text Analysis and Search for Bioinformatics*.

[Sekimizu et al., 1998] Sekimizu, T., Park, H., and Tsujii, J. (1998). Identifying the Interaction between Genes and Gene Products Based on Frequently Seen Verbs in Medline Abstracts. In *Proceedings of the Ninth Workshop on Genome Informatics*, pages 62–71.

[Shamir and Sharan, 2002] Shamir, R. and Sharan, R. (2002). Algorithmic Approaches to Clustering Gene Expression Data. In Jiang, T., Smith, T., Xu, Y., and Zhang, M., editors, *Current Topics in Computational Biology*, pages 269–299. MIT Press.

[Shatkay et al., 2000] Shatkay, H., Edwards, S., Wilbur, W., and Boguski, M. (2000). Genes, Themes, and Microarrays: Using Information Retrieval for Large-Scale Gene Analysis. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*.

[Speer et al., 2004a] Speer, N., Spieth, C., and Zell, A. (2004a). A Memetic Clustering Algorithm for the Functional Partition of Genes Based on the Gene Ontology . In

Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB), pages 252–259.

[Speer et al., 2004b] Speer, N., Spieth, C., and Zell, A. (2004b). A Memetic Co-Clustering Algorithm for Gene Expression Profiles and Biological Annotation. In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC)*, pages 1631–1638.

[Speer et al., 2005] Speer, N., Spieth, C., and Zell, A. (2005). Biological Cluster Validity Indices Based on the Gene Ontology. In *Proceedings of the 6th International Symposium on Intelligent Data Analysis (IDA)*, pages 429–439.

[Srihari et al., 2003] Srihari, R., Ruiz, M., and Srikanth, M. (2003). Concept Chain Graphs: A Hybrid IR Framework for Biomedical Text Mining. In *Proceedings of the SIGIR Workshop on Text Analysis and Search for Bioinformatics.*

[Stapley and Benoit, 2000] Stapley, B. and Benoit, G. (2000). Biobibliometrics: Information Retrieval and Visualization from Co-occurrences of Gene Names in Medline Abstracts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 529–540.

[Stephens et al., 2001] Stephens, M., Palakal, M., Mukhopadhyay, S., Raje, R., and Mostafa, J. (2001). Detecting Gene Relations from Medline Abstracts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB).*

[Stevens et al., 2000] Stevens, R., Goble, C., and Bechhofer, S. (2000). Ontology-based Knowledge Representation for Bioinformatics. *Briefings in Bioinformatics*, 1(4):398–416.

[Sun and Lim, 2001] Sun, A. and Lim, E.-P. (2001). Hierarchical Text Classification and Evaluation. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 521–528.

[Suzuki et al., 2001] Suzuki, E., Gotoh, M., and Choki, Y. (2001). Bloomy Decision Tree for Multi-Objective Classification. In *Proceedings of the International Conference on Principles of Data Mining and Knowledge Discovery (PKDD)*, pages 436–447.

[Swanson, 1986] Swanson, D. (1986). Fish Oil, Raynaud's Syndrome, and Undiscovered Public Knowledge. *Perspectives in Biology and Medicine*, 30:7–18.

[Swanson, 1988] Swanson, D. (1988). Migraine and Magnesium: Eleven Neglected Connections. *Perspectives in Biology and Medicine*, 31:526–557.

[Tamames et al., 1998] Tamames, J., Ouzounis, C., Casari, G., Sander, C., and Valencia, A. (1998). EUCLID: Automatic Classification of Proteins in Functional Classes by their Database Annotations. *Bioinformatics*, 14(6):542–543.

[Thomas et al., 2000] Thomas, J., Milward, D., Ouzounis, C., Pulman, S., and Carroll, M. (2000). Automatic Extraction of Protein Interactions from Scientific Abstracts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, pages 541–552.

[Toutanova et al., 2001] Toutanova, K., Chen, F., Popat, K., and Hofmann, T. (2001). Text Classification in a Hierarchical Mixture Model for Small Training Sets. In *Proceedings of the International Conference on Information and Knowledge Management*, pages 105–113.

[Tsochantaridis et al., 2004] Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. (2004). Support Vector Machine Learning for Interdependent and Structured Output Spaces. In *Proceedings of the International Conference on Machine Learning (ICML)*.

[van Rijsbergen, 1979] van Rijsbergen, C. (1979). *Information Retrieval*. Butterworths, second edition.

[Verspoor et al., 2005] Verspoor, K., Cohn, J., Joslyn, C., Mniszewski, S., Rechtsteiner, A., Rocha, L., and Simas, T. (2005). Protein Annotation as Term Categorization in the Gene Ontology using Word Proximity Networks. *BMC Bioinformatics*, 6 (Supplement 1).

[Wang et al., 2004] Wang, H., Azuaje, F., Bodenreider, O., and Dopazo, J. (2004). Gene Expression Correlation and Gene Ontology-Based Similarity: an Assessment of Quantitative Relationships. In *Proceedings of the IEEE 2004 Symposium on Computational Intelligence in Bioinformatics and Computational Biology*, pages 25–31.

[Wang et al., 2001] Wang, K., Zhou, S., and He, Y. (2001). Hierarchical Classification of Real Life Documents. In *Proceedings of the SIAM International Conference on Data Mining*.

[Wang et al., 1999] Wang, K., Zhou, S., and Liew, S. (1999). Building Hierarchical Classifiers Using Class Proximities. In *Proceedings of the International Conference on Very Large Data Bases (VLDB)*, pages 363–374.

[Weigend et al., 1999] Weigend, A., Wiener, E., and Pedersen, J. (1999). Exploiting Hierarchy in Text Categorization. *Information Retrieval*, 1(3):193–216.

[Wibowo and Williams, 1999] Wibowo, W. and Williams, H. (1999). On Using Hierarchies for Document Classification. In *Proceedings of the Australian Document Computing Conference*, pages 31–37.

[Wibowo and Williams, 2002a] Wibowo, W. and Williams, H. (2002a). Simple and Accurate Feature Selection for Hierarchical Categorisation. In *Proceedings of the ACM Symposium on Document Engineering*, pages 111–118.

[Wibowo and Williams, 2002b] Wibowo, W. and Williams, H. (2002b). Strategies for Minimising Errors in Hierarchical Web Categorisation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 525–531.

[Wiener et al., 1995] Wiener, E., Pedersen, J., and Weigend, A. (1995). A Neural Network Approach to Topic Spotting. In *Proceedings of the Annual Symposium on Document Analysis and Information Retrieval (SDAIR)*, pages 317–332.

[Wooster and Weber, 2003] Wooster, R. and Weber, B. (2003). Breast and Ovarian Cancer. *New England Journal of Medicine*, 348(23):2339–2347.

[Yandell and Majoros, 2002] Yandell, M. and Majoros, W. (2002). Genomics and Natural Language Processing. *Nature Reviews Genetics*, 3:601–610.

[Yeh et al., 2003] Yeh, A., Hirschman, L., and Morgan, A. (2003). Evaluation of Text Data Mining for Database Curation: Lessons Learned from the KDD Challenge Cup. In *Proceedings of the International Conference on Intelligent Systems for Molecular Biology (ISMB)*, pages 331–339.

[Zhdanova and Shishkin, 2002] Zhdanova, A. and Shishkin, D. (2002). Classification of Email Queries by Topic: Approach Based on Hierarchically Structured Subject Domain. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning*, pages 99–104.