Induction from Multi-label Examples

By

Hind Hazza Talal Alsharif

A thesis submitted for the requirements of the degree of Master on Computer Science

FACULTY OF COMPUTER AND INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY
JEDDAH – SAUDI ARABIA
1435 H – 2014 G (25-6-2014)

بسم الله الرحمن الرحيم

Induction from Multi-label Examples

By

Hind Hazza Talal Alsharif

A thesis submitted for the requirements of the degree of Master on Computer Science

Supervised By

Dr. Wadee Saleh Alhalabi

FACULTY OF COMPUTER AND INFORMATION TECHNOLOGY
KING ABDULAZIZ UNIVERSITY
JEDDAH – SAUDI ARABIA
1435 H – 2014 G (25-6-2014)

Induction from Multi-label Examples

By

Hind Hazza Talal Alsharif

This thesis has been approved and accepted in partial fulfillment of the requirements for the degree of Master on Computer Science

EXAMINATION COMMITTEE

	Name	Rank	Field	Signature
Internal Examiner	Dr.Fadi Fouz	Professor	Computer science	[illegible]
External Examiner	Dr. Ali Morfeq	Assistant professor	Computer engineering	[illegible]
Co-Advisor				
Advisor	Dr.Wadee Alhalabi	Assistant professor	Computer science	[illegible]

KING ABDULAZIZ UNIVERSITY

1435 H – 2014 G (25-6-2014)

Induction From Multi-label Examples

By: Hind Hazza Alsharif

Abstract

The task of text categorization is related to the assignation of one or more classes to a document. In order to solve this problem, the simplest machine learning approaches induce a binary classifier separately for each class, and then use these classifiers in parallel. An example of application belonging to this group and solved in this thesis is represented by a digital library collection which was classified into classes and sub-classes in a hierarchical order. Another important issue considered is the fact that a document can belong to more than one class, and therefore, a high performance multi-class label classifier was employed.

The main general objective of the current work is to point out the advantages of machine learning techniques for applications in text categorization area. Another aspect is related to the database requirements of these techniques in terms of training and testing, when high performance is desired. In this context, two situations were identified: i) 10 to 15% of the data used for training, and testing and ii) > 50% of the data set used for training and testing. In the latter case, the machine

learning may not have a significant contribution, as the computational effort and time consumed are very high. However, if 10 to 15% of the data set is needed, then, machine learning has a great contribution.

The last issue approached in this research is the inter-class relation, which means, if the example is classified to belong to a class C then the example belong to parents and grandparents of the class C. The main question arising was: is opposite way true too?

In order to answer all these important aspects, a framework to automatically classify documents was employed.

Dedicated to

I would like to express my gratitude towards my parents and family for their kind cooperation and encouragement, which has helped me in completion of this project.

I would like also to express my special thanks to my advisor for giving me his guidance and constant supervision as well as for providing necessary information regarding this work.

At last, I would like also to thank the examination committee for their time and attention.

Table of Contents

List of Tables

List of Figures

List of Symbols

HMC – Hierarchical Multi-label Classification

DAG – Directed Acyclic Graph

SVM – Support Vector Machine

KNN – K-Nearest Neighbor

MLNP- Mandatory Leaf Node Problem

NMLNP- Non Mandatory Leaf Node Problem

TAN – Tree Augmented Naïve Bayes

CL – Chow and Liu

NN – Neural Network

RF – Random Forest

OSH – Optimal Separating Hyper plane

DBT – Double Binary Tree

BT – Binary Tree

DST – Dempster Shafer Theory

SMO – Sequential Minimal Optimization

CSSA – Condensing Sort and Selection Algorithm

CSSAG- Condensing Sort and Selection Algorithm for the DAG-structured hierarchies

HiBLADE- Hierarchical Multi-label Boosting with Label Dependency

LP – Label Power set

RAKEL – Random K-Label set

MLKNN – Multi label K-Nearest Neighbor

LIFT – Multi label learning based on Label Specific Features

MLSF – Multi level Classification based on Specific Features

PIPL – Meta Path based Instance and Label Correlation

BSVM – Binary Support Vector Machine

ECC – Ensemble of Classifier Chains

MLCBR – Multi-label Classification based on Case-Based Reasoning

CBR – Case Based Reasoning

PRE – Probabilistic Reuse based on Experience

PR – Probabilistic Reuse

BCC – Bayesian Chain Classifier

MLP – Multi Layer Perceptron

HMCLMLP – Multi-label Classification with Local Multi-layer Perceptron

PRCURVES – Precision Recall Curves

PCT – Predictive Clustering Tree

LDA – Latent Dirichlet Allocation

TP – True Positive

TN – True Negative

FP – False Positive

FN – False Negative

CGI – Carnegie Group Inc.

PCA – Principle Component Analysis

ANN – Artificial Neural Networks

KDE – Kernel Dependency Estimation

DT – Decision Tree

IDF – Inverse Document Frequency

Introduction

The main goal of a classification induction process is to find the mechanism (rules) able to place an example or a stream of examples into sets of categories called classes. In the case of multi-label classification induction, an example is allowed to belong to more than one class at a time, and the classes are hierarchically ordered. This is referred to as Hierarchical Multi label Classification (HMC). The classification of a library collection (where book titles represent the examples and each of the scientific field represents a class) is an example of an HMC problem. The class-to-class relations are defined by a Directed Acyclic Graph (DAG) (Figure 1.1) which indicates that there are no cycles. The nodes and the edges define the structure of the network, and the conditional probabilities are the parameters to give the structure to the graph [1].

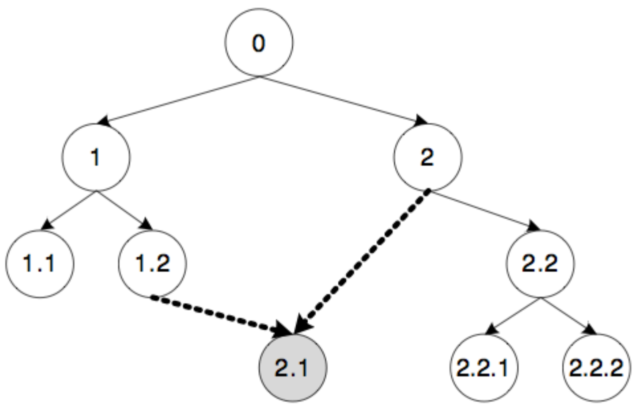


Figure 1.1: An example of a DAG-structured class hierarchy as presented in [2]

In this research, the focus is on the HMC problem, with emphasis on several case studies used for drawing observations and reaching general conclusions. Aiming to build a proper induction system for these problems, the top down approach was preferred. It started by inducing a classifier for each class of the highest level of the DAG and continued downward by employing the higher-level classifiers when creating the training sets for lower-level classifiers.

The scope was to develop a proprietary methodology and algorithm and compare it with a couple of many popular algorithms including Support Vector Machines (SVM) [3-5], K-Nearest Neighbor (KNN) [6] and Naïve Bayes [7]. The comparative study consisted in: i) classifying examples into hierarchically ordered classes and ii) finding their inter-class relation.

As the work in this research progressed and as recommended by [2], we realized that HMC's performance has to be evaluated along somewhat different criteria than those used in classical machine learning. For example, let C be a set of classes to which an example X belongs to. A perfect classifier will label X with all classes from C, never suggesting any class from outside C; moreover, an HMC usually requires that any X that has been labeled with C_i should also be labeled with all ancestors of C_i in the class hierarchy. To be able to reflect these requirements in performance evaluation, an adequate extensions of precision and recall introduced by Clare et al. [8] was used.

1.1. Problem Statement

A graph mainly consists of a set of nodes, N, and a set of edges, E, where an edge is an ordered pair of nodes, $(N_p, N_c) \in E \subseteq \{N \times N\}$. In this pair, N_p is known as a parent, and N_c as a child. A path ($N_a \rightarrow N_c$) from an ancestor (N_a) to a child (N_c) is referred to be a series of edges, $\{(N_1, N_2), (N_2, N_3), \ldots, (N_{n-1}, N_n)\}$ in a way that

N_1 = Na and $N_n = N_c$. The existence of a path in a DAG, is Na → Nc, guarantees the non-existence of the opposite-direction path, Nc → Na. A leaf node is known to be a node without any child, and a root node is known to be a node without any parent.

In this research, this problem is addressed by considering a set of class labels (C) whose mutual relations are specified by a class hierarchy (H) which has the form of a DAG in which each node represents only one class.

$X \subset R^p$ is a finite set of examples, each described by a set of p numeric attributes. We assume that each $x_i \in X$ is assigned a set of class labels, $L = \{C_1, ..., C_i\} \subseteq H$ (all classes belong to the given class hierarchy). An example belonging to class C_c is assumed to also belong to all C_c's ancestor classes (C_a). This property is called "hierarchical constraint".

There are two versions of the hierarchical classification task: i) the Mandatory Leaf-Node Problem (MLNP), where only the leaf-node classes are used and; ii) the Non-Mandatory Leaf Node Problem (NMLNP), where an example can be labeled with any class from the given class hierarchy. Considering the class hierarchy from Figure 1.1, MLNP permits an example to be labeled only with a subset of {C1.1, C2.1, C2.2.1, C2.2.2}, but NMLNP allows also the other class labels (e.g., C1 or C2.2). This research focuses on the general NMLNP, because the examples are assigned to any node in the hierarchy.

Two main questions were addressed in this research:

- Suppose that a machine learning algorithm has already induced classifiers for some highest-level classes. Does this facilitate any future attempts at the induction of lower-level classes? For instance, if an example was classified to a lower level class, can this example belong to the parent and grandparent classes?

- Turning this upside down, suppose we know the lowest-level classes. Can this be exploited in the induction of the parents of these classes? For instance, if an example was classified in the upper level class, can this example be a parent of the lower level classes?

Aiming to answer these questions, a proprietary algorithm will be built. It will test whether an example is classified into its corresponding child and grandchild, as well as if the grandchild is belonging to its accurate parent and grandparent. The focus is on the inter-class relations and we want to look at the parent-child and child-parent relations, this aspect representing the main contribution of this study.

The significance of the research is the motivation for the use of machine learning in digital libraries which can be defined as follow:

- The digital library needs to be able to identify all documents relevant to a user's query. This function is sometimes supported by an indexing system in which each document is tagged with the labels of all the topics it represents.

- The indexing system is relatively easy to create in a small collection: an expert reads each single document, and then decides which topics it represents.

- In large collections, this might be expensive and clearly impossible if hundreds of thousands of documents are added to the library every week, or even on a daily basis.

- In this latter case, one solution is to classify manually only a subset of the documents, and then employ the training set, to obtain the induction of a classifier.

- The induced classifier then labels those documents that have not been classified manually.

- The principle can be applied to other domains, not just digital libraries.

The main research question is the following: how many documents should we classify manually if we want to induce a high-performance classifier? To put the question in another format: How much can be gained from the use of machine learning? For example, suppose we have 10^6 documents. If we manually classify only a few, the induced classifier will over fit the training examples, and thus perform poorly on the remaining documents. The situation will improve if the training set consists of about 10% of the collection or more; but then, the price of manual classification will become prohibitive. This motivates an experimental study whose goal is to identify the right size of the training set, and this is what we want to do in this research.

Possible conclusions:

- It may turn out that only a small percentage of all examples are enough for the induction of a relatively high-performance classifier. In this case, the use of machine learning is justified.

- Conversely, it may turn out that even using 50% of the examples for training is not enough. In this case, machine learning does not seem to help.

- Most likely, the observed result will be somewhere between these two extremes.

- We might want to verify if the observation is the same in each of the studied experimental domains. This means, we want to repeat this experiment for several different domains.

Background

Over the past few years, studies of induction from multi-label examples have targeted two specific strategies: induction of sets of binary classifiers, and induction of one large multi-label classifier. For the induction of sets of binary classifiers, mechanisms based on Bayesian theory were studied by Friedman et al. [7], and McCallum and Nigam [9]. The latter was investigated by Baoli et al. [6], and the currently popular SVM were discussed by Joachimis [4] and Kwok [10]. Unfortunately, binary classifiers ignore inter-class relations, which sometimes lead to performance degradation. In this study, the focus is on these inter-class relations.

2.1 Bayesian Networks:

The Naïve Bayes classifier learns (from training data) the conditional probability of each attribute A_i given the class label C. Classification is then done by applying Bayes rule to compute the probability of C given the particular instance of $A_1 , . . . , A_n$, and then predicting the class with the highest posterior probability. This computation is rendered feasible by making a strong independence assumption: all the attributes A_i are conditionally independent given the value of the class C. The term independence indicates the probabilistic independence that is, A is independent of B.

A naive Bayesian classifier has the simple structure shown in Figure 2.1. This network captures the main assumption behind the naive Bayesian classifier, namely, that every attribute (every leaf in the network) is independent from the rest of the attributes, given the state of the class variable (the root in the network). Thus, it is said that the performance of naive Bayes is somewhat due its dependency [9].

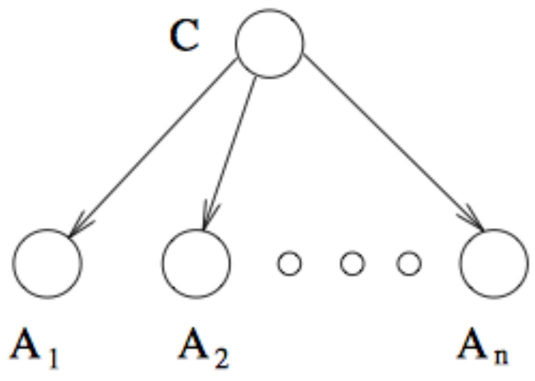


Figure 2.1: The structure of the naive Bayes network.

Friedman et al. [7] evaluated several approaches for inducing classifiers from data based on the theory of Bayesian networks. They presented a method they call Tree Augmented Naive Bayes (TAN), which outperforms the base algorithm, and at the same time maintains the computational simplicity (with no search involvement) and robustness that characterize Naive Bayes. Their empirical evaluation included TAN and Chow and Liu (CL) multi-net classifier [11]. CL describes a procedure for constructing a Bayesian network from data. Such procedure reduces the problem of constructing a maximum likelihood tree to finding a *maximal weighted spanning tree* in a graph. The problem of finding this type of tree is to select a subset of arcs from a graph such that the selected arcs constitute a tree and the sum of weights attached to the selected arcs is maximized. Both TAN and CL multi-nets reflect a good tradeoff between the quality of the approximation of correlations among attributes and the computational complexity in the learning stage. The learning procedures are guaranteed to find the optimal tree structure, and, as the experimental results show, they perform well in practice against state-of-the-art classification methods.

McCallum and Nigam [9] aimed to clarify the confusion between two first-order probabilistic models (making the naïve Bayes assumption applied for text classification) by describing their differences and details. The first uses a multivariate Bernoulli model (a Bayesian Network with no dependencies between words and binary word features as in [13], [14]), while the other uses a multinomial model (a unigram language model with integer word counts as presented in [15, 16]). The results on five text corpora indicated that the multivariate Bernoulli performs well with small vocabulary sizes, but that the multinomial version usually performs even better at larger vocabulary sizes providing on average a 27% reduction in error over the multivariate Bernoulli model at any vocabulary size.

2.2 K-Nearest Neighbor:

In a text categorization system based on the K-Nearest Neighbor algorithm (KNN), k is the most important parameter. To classify a new document, the k-nearest documents in the training set are first determined. The prediction of categories for this document can then be made according to the category distribution among the k nearest neighbors. Generally speaking, the class distribution in a training set is not even; some classes may have more samples than others. The system's performance is very sensitive to the choice of the parameter k. And it is very likely that a fixed k value will result in a bias for large categories, and will not make full use of the information in the training set [6].

Baoli et al. [6] studied a text categorization system based on KNN, and an improved KNN strategy (in which different numbers of nearest neighbors for different categories are used instead of a fixed number across all categories) was proposed. The numbers of nearest neighbors selected for different categories are adopted to their sample size in the training set. Experiments on two different datasets showed that the proposed approach is less sensitive to the parameter k than the traditional ones, and can properly classify documents belonging to smaller classes when employing a large k. The strategy is more efficient with cases where estimating the parameter K via cross-validation is not possible and the class distribution of a training set is skewed.

K- KNN method is often used in text document classification. It is considered as the simplest of all machine learning algorithms. An object is classified by a majority vote of its neighbors, with the object being assigned to the class most common amongst its k nearest neighbors. The accuracy of the k-NN algorithm is not guaranteed if the feature scales are not consistent with their importance. Therefore, it is sensitive to the local structure of the data. Random Forest (RF) classifier can handle irrelevant feature and gives estimate of what variables are important in the classification [17].

2.3 Random Forest:

Neural network (NN) based approaches are applicable to multilevel non-linear problems. The variables transformation is automated in the computational process. The main disadvantage of NNs is the fact that they are slow when many training data sets exist. Therefore, they are not suitable in all cases of classification problems. On the other hand, RF can handle thousands of input data, faster than other methods and in the same time avoid model over fitting [17].

In [17], Aung and Hla proposed a multi-category classification for web pages by using RF classifier. The accuracy of the proposed approach was compared with decision tree classifier using the same yahoo web pages, the results showing that the proposed approach was suitable for multi-category web page classification.

Wang et al. [18] goal was to predict DNA-binding residues directly from amino acid sequence data using RF. Previous methods have been reported for predicting DNA-binding. For example, Ahmad et al. [19] analyzed the structural data protein-DNA complexes, and used the amino acid sequences to train NNs for DNA-binding site prediction. Yan et al. [20] constructed Naïve Bayes classifiers using the amino acid identities of DNA-binding sites and their sequence neighbors. However, the prediction accuracy was low in these studies, and this is because amino acid sequences were directly used for classifier construction. It was found that classifier performance was significantly improved by the use of biochemical features for input encoding, and the SVM classifier outperformed the NN predictor [21, 22].

RF learning algorithm, has the capability of handling large number of input variables and avoiding model over fitting [23]. The results from this study indicate that DNA-binding site prediction can be significantly improved by using the RF-based approach with biochemical features and several new descriptors of evolutionary information for input encoding.

Yang et al. [24] investigated the possibilities of applying RF in machine fault diagnosis and proposed a hybrid method combined with genetic algorithm to improve the classification accuracy. The application research on RF is important and necessary because of its fast execution speed, the characteristics of tree classifier, and high performance in machine faults diagnosis. The proposed method is demonstrated by a case study on induction motor fault diagnosis. Experimental results indicate the validity and reliability of the RF-based fault diagnosis method, a high accuracy rate of diagnosis (98.89%) being obtained. The comparison result also showed that the optimized RF-based method is competitive with other classification method.

In his study [25], Osman solved a binary classification problem, where an ensemble of Decision Tree (DT) based classifiers is trained on-line, new images are always added and the recognition decision is made without delay. The ensemble of decision tree classifier is combined with forest classifier using averaging, generated on-line RF classifier. To represent an object visual features, they first employ object descriptor models based on bag of covariance matrices, and after that, they run their online RF learner to select object descriptors and to learn an object classifier. It was considered how machine learning models for object recognition categories can build 'incrementally' or 'on line', so that new images were continuously added and the recognition decision was made without delay. The main computational advantage in using RF classifier is that each DT classifier can be trained independently from each other and in parallel. Results showed superior performance with the standard RF, Adaboost, and SVM classifier.

2.4 Support Vector Machine:

SVM aims to fit an Optimal Separating Hyperplane (OSH) between classes by focusing on the training samples that lie at the edge of the class distributions. The OSH is oriented such that it is placed at the maximum distance between the sets of support vectors, which leads to a more accurately generalization and training error minimization, similar to NNs [26].

Joachims [4] introduced the SVM for text categorization from examples by analyzing particular properties of learning with text data. Practical results showed that SVM's achieved good performance on text categorization tasks, substantial improvements over the currently best performing methods being observed. Kwok [10] studied SVM in text categorization because it allows easy incorporation of new documents into an existing trained system. Also, dimension reduction is optional with SVM's. Therefore, an SVM adapts efficiently in dynamic environments that require frequent additions to the document collection. The author investigated a different approach of integrating both dimension reduction and classification. The study showed the SVM's characteristics that make it very useful to the problem of text categorization and of information retrieval in general. SVMs can obtain better results by using them as pre-processing tools.

The binary SVM can be extended for a one-shot multiclass classification. The one-shot multiclass SVM has a relative advantage to the binary SVM-based approaches. This advantage is represented by the fact that it needs to be optimized only once. The multiclass SVM classification of all classes occurs in a single step [26].

Mathur and Foody [26], aimed to evaluate the multiclass and binary-based SVM approaches for the derivation of a multiclass land cover classification from remotely sensed data. They suggested two approaches for multi class classification by SVM:

- "one-against-all" approach, where a set of binary classifiers (each trained to separate one class from the rest) is undertaken and each pixel is allocated to the class for which the largest decision value was determined.

- "one-against-one" approach, where, a series of classifiers are applied to each pair of classes, with the most commonly computed class label kept for each pixel.

For the one-shot multiclass SVM approach, parameters need be optimized only once, while with the one-against-one and one-against-all binary strategies a series of analysis are required. The one-against-all and one-against-one strategies required five and ten optimizations, respectively. Also, the smallest number of support vectors was used with the one-shot multiclass SVM classifier, while other strategies for multiclass classification required a larger number of support vectors. From the results, it was obvious that the one-shot multiclass SVM classification yielded the most accurate classification. One problem observed was that the one-against-all approach was not able to label all cases appropriately.

Liu et al. [27] proposed a multi-class classification method of SVM based on double binary tree (DBT-SVM). Each sub-classifier of BT-SVM is modified and after unknown samples are classified by the modified BT-SVM, the negative output of its final sub-classifier can be classified again by adding an auxiliary BT-SVM. Thus, the misclassified samples mixed in the negative output can be classified correctly. Existing problem in BT-SVM method include:

- '*Irreversibility*', which occurs once a sub classifier mistakes a positive sample for a negative one, so the result is incorrect and there is no chance for re-classification.

- '*Error accumulation*' phenomenon. It appears when each sub-classifier in the hierarchy of a binary tree will mistake some positive samples for negative ones and then input them into the next-level sub-classifiers, resulting in an accumulation of the misclassified samples in the samples waiting for classification.

- The upper sub-classifiers in the binary tree structure of BT-SVM have greater influence over the generalization capacity of the overall classification model.

DBT-SVM turned to be able to provide a higher general classification accuracy compared with the BT-SVM. It improves the classification accuracy of the earlier classified classes while lowers the classification accuracy of the latter classified classes. Therefore, the classification order of the classes should be decided according to their importance without applying randomness.

Kubat et al. [28] proposed a new technique for induction in multi-label text classification domains. They applied a well-known boosting algorithm, AdaBoost.MH, as a “baseline induction algorithm” for the induction of a set of sub classifiers, each from the same training set. In addition, they developed a new fusion method around the principles of the Dempster-Shafer Theory, called DST-fusion. Experiments showed that DST-fusion can lead to impressive savings in the computational time without impairing the classification performance. DST-Fusion and “weighted sum” outperformed the more traditional methods of plain voting and weighted majority voting. Moreover, when comparing DST-Fusion with a more traditional approach, it was observed that, the Multi-Label C4.5 (based on induction of decision trees), might be a better choice.

2.5 Hierarchical Multi-label Classification Strategies

Silla and Freitas [29] explored the solutions to the HMC problems and presented three fundamental strategies: 1) flat classification, 2) top-down approach "local classification", 3) the "big-bang" approach or global classification.

2.5.1 Flat classification

The advantage of this strategy is that it enables the use of traditional machine-learning techniques such as neural networks, decision trees, or SVM to be implemented in the HMC as reported by [30-32]. Basically it ignores the class hierarchy and deals only with the leaf-node classes (as if the problem were MLNP), whether by a single multi-label classifier or by a set of binary classifiers (a separate one for each leaf node). If the leaf-node class label is known for each example, this strategy is possible. Besides, if the nature of the application seems to allow the user to afford the inability to identify non-terminal classes.

2.5.2 Top-down approach (local classifier):

The most common approach in HMC induction is the local classifier. In the simplest scenario, for each node in the DAG-specified class hierarchy, a separate (local) classifier is induced, and the processing is started by creating a whole hierarchy of classifiers, from top levels going downwards.

The main advantage of this method is simplicity. On the other hand, the approach tends to suffer from "error propagation", which means that misclassifications of the higher- level classes are propagated to the lower levels.

The first experiments with this approach were provided by Koller and Sahami

[14] by choosing Naive Bayes to induce each individual class. The authors experimented with tree-structured class hierarchies with no more than one parent for any node and limited to just two levels.

Fagni and Sebastiani [78, 79] compared four different policies (Sibling, ALL, BestGlobal, and BestLocal) to generate a set of binary training data. Tree-structured hierarchical versions of boosting and SVM called TreeBoost and TreeSVM were used. The best results were obtained with the Sibling policy in which the negative training examples of the i^{th} node are all positive examples of its Sibling nodes in the hierarchy.

This strategy was applied to text classification by Sun and Lim [33], where the class hierarchy was a plain tree structure. They induced two SVMs for each class: a local classifier and a sub-tree classifier. An example is labeled as C_i by the local classifier, while the sub-tree classifier decides whether or not this example should be passed to c_i's sub-classifiers. This approach was extended to domains with DAG-structured class hierarchies, by Nguyen et al. [34], the DAG hierarchy being transformed into a set of tree hierarchies. Experimental results indicated high classification performance as well as high computational costs.

Looking to further improve the performance, Secker et al. [35] used several induction algorithms for each node of the hierarchy: Naive Bayes, SMO, 3-NN, etc. Ten classifiers were trained for each node, and the one with the best classification results was selected. This improved classification accuracy, but the computational costs were even higher than in the previous attempt.

Bi and Kwok [36] applied the Kernel Dependency Estimation (KDE) to reduce the number of classes in the hierarchy during the training process. This procedure was applied because the number of classes in the hierarchy is usually unmanageable. The authors proposed an algorithm called “Condensing Sort and Selection Algorithm (CSSA)” for the tree structured hierarchies and, then, extended it to the CSSAG algorithm for the DAG-structured hierarchies. However, they did not report experimental results regarding induction time and the number of reduced classes.

Alaydie et al. [37] proposed a framework called “HiBLADE (Hierarchical multi-label Boosting with Label Dependency),” applied to tree-structured hierarchies. The classifier for each class is a boosting-type algorithm, such as ADABOOST, where the new model for each boosting iteration is updated by utilizing the proposed Baysian correlation.

2.5.3 The “big-bang” approach (global classifier):

Some authors preferred to induce one big (global) classification model to cover the entire class hierarchy, instead of inducing a separate binary classifier for each node. In this manner, mutual interdependencies of the classes can be easily taken into account, and the global classifier is often smaller than the total of the local classifiers.

Clare and King [8] developed a hierarchical extension to the decision-tree generator C4.5 [38] and applied it to functional-genomics data. Their system is known as HC4.5, a mechanism for weighing the entropy formula (in order to give higher priority to more specific classes) being induced.

Seeking to make the decision-tree paradigm applicable to hierarchical domains, an attempt was reported by Blockeel et al. [39] whose Clus-HMC is a hierarchical version of the earlier "predictive clustering tree" (PCT) [40]. Ven et al. [80] improved Clus-HMC so it could be used in DAG-specified class hierarchies. Schietgat et al. [12] proposed an ensemble version of the algorithm Clus-HMC-ENS. Although the ensemble concept can improve classification accuracy, its computational costs are much higher than those of the original Clus-HMC.

A global-approach hierarchical framework based on the K-Nearest Neighbor classifier (k-NN), was proposed by Pandey et al. [41]. The system's improvements include: i) a Lin's semantic similarity measure used as a distance measure; ii) the prediction function of the i-th class incorporates the inter-relationship score of the i-th class to other classes in the hierarchy; and iii) the mechanism to filter insignificant class inter-relationships was suggested.

Lo et al. [42] proposed a basis expansion model for multi-label classification, where a basis function is a Label Power set (LP) classifier trained on a random k-label set. LP [43] method is a multi-label learning algorithm which basically reduces the multi-label classification problem to a single-label multi-class classification problem by dealing with each distinct combination of labels in the training set as a different class. Random k-Label sets (RAKEL) [44] has introduced to overcome the drawback of the LP method. It randomly selects a number of label subsets from the original set of labels and then uses LP for training the corresponding multi-label classifiers. Experiments were conducted on ten benchmark datasets belonging to different domains, including: scene, enron, cal500, major miner, medical bibtex, and four versions of delicious (from dlc1 to dlc4). More details on these data sets are available at the MULAN library website [45].

Qu et al. [46] proposed a Multi-Label classification algorithm based on label-Specific Features (MLSF). The feature density on the positive and negative instances set of each class was first computed and after that, the features of high density from the positive and negative instances set of each class were selected. The intersection was taken as the label-specific features of the corresponding class. Finally, the multi-label data was classified on the basis of label-specific features. The classifiers induction process of MLSF is similar to the original binary classifiers. Given an unlabeled instance $x_u \in U$, the feature sets for each class label are first rebuild based on the label-specific features, and then the corresponding classifier is used to predict whether it has the label or not. The proposed MLSF is compared with three multi-label learning algorithms, including ML-KNN, LIFT, and Rank-SVM. The experiments were employed on both regular-scale and large-scale. For the results, common evaluation criteria for multi-label classification were used (hamming loss, one-error, coverage, and average precision). It is observed, that the performance of MLSF is comparable to that of LIFT on the regular-scale data sets and large-scale data sets and that MLSF and LIFT algorithms perform significantly better than ML-KNN and Rank-SVM.

Kong et al. [47], used the heterogeneous information networks to simplify the multi-label classification process. They focused on extracting the relationships among different class labels and data samples by mining the linkage structure of heterogeneous information networks. These relationships can be then used to effectively infer the correlations among different class labels in general, as well as the dependencies among the label sets of data examples that are inter-connected in the network. The proposed multi-label collective classification algorithm (called PIPL) was tested on a bio-informatic dataset SLAP [48], which is a heterogeneous network containing integrated data related to chemical compounds, genes, diseases, side effects, pathways etc.

2.6. Other Classification Methods

Other existing multi-label classification methods include:

- BSVM (binary SVM); ECC (multi-label classification + ensemble);
- PISl (binary decomposition + meta-path based instance correlation):a collective classification approach [49], where instance correlations are from heterogeneous network;
- Icml (simple label correlation + instance correlation in homogeneous network): this method was proposed by Kong et al. [50] which exploit relational features for inter-instance dependencies based on homogeneous network for multi-label collective classification;
- PIml (simple label correlation + meta-path based in- stance correlation): a multi-label collective classification approach extended from PIsl [49] by adding relational features according to inter- instance-cross-label dependencies for multi-label collective classification [50];
- PIPL (meta-path based instance and label correlation): a method for multi-label collective classification in heterogeneous information networks. The only difference between PIPL and PIml is that PIml does not consider the meta-path based label correlation.

In order to achieve a reduction in time costs without compromising accuracy, Nicolas et al. [51] proposed the MLCBR algorithm which is a system for multi-label classification based on Case-Based Reasoning. In their study, they have investigated the characteristics of the most popular systems in this area, MLKNN (Multi-Label K Nearest Neighbor) and RAKEL (Random L Label sets), where they have observed that the main drawback of these specific systems is the time required. The focus was on the retrieval and reuse stages of CBR because these are the features that lead toward their objectives, namely the reduction of computational time and improving the accuracy. The retrieval stage of Multi-label Case-Based Reasoning Algorithm (MLCBR) is based on MLKNN where the K most similar cases to the case study are recovered of the case memory. In the reuse phase two approaches were proposed. Probabilistic Reuse (PR) is the first option, where the final classification is made through a voting process in which all the recovered cases are equally weighted. The second option is Probabilistic Reuse, which is based on Experience (PRE). It adds the concept of experience to better weight the recovered cases. Results of the proposed model were compared with other two competitive multi-label learning systems, MLKNN and RAKEL, using seven synthetic dataset and three real-world datasets used as benchmark by multi-label classification community (scene, emotions and yeast). All the phases of the experimentation process obtain the accuracy values with an average of the standard deviation of 10 independent executions of ten-fold cross-validation process with different randomness seed.

Experiments show that, a level of accuracy equivalent to that obtained by a competent system (MLKNN) and statistically has better results than the benchmark (RAKEL). In both comparisons the computational time of the model is lower than the one performed by previous platforms.

Chain classifiers have been recently proposed to address some problems such as, high computational complexity, and ignoring possible dependencies among classes. In chain classifier each classifier in the chain learns and predicts the label of one class given the attributes and all the predictions of the previous classifiers in the chain.

Sucar et al. [52], introduced a method for chaining Bayesian classifiers that combines the strengths of classifier chains and Bayesian networks for multi-label classification. A Bayesian network is induced from data to represent the probabilistic dependency relationships between classes, and constrain the number of class variables used in the chain classifier by considering conditional independence conditions. A Bayesian Chain Classifier (BCC) makes two basic assumptions that are, a Bayesian network can represent the class dependency structure given the features, and the total abduction is approximated by the concatenation of the most probable individual classes. A chain classifier can be constructed by inducing first the class that does not depend on any other class and then proceed with its children. Thus, the constructed steps are: create an order of classes in the chain based on the dependencies between classes given the features. These dependencies can be represented as a BN, and therefore simpler base classifiers can be created by considering conditional independencies between classes. Different Bayesian chain classifiers were tested on 9 benchmark multi-label data sets; each of them with different dimensions. For performance evaluation, several metrics were used to evaluate the performance of multi-label classifiers: Mean accuracy over the d class variables (accuracy per label), Global accuracy over the d-dimensional class variable (accuracy per example, also called subset zero-one loss), Multi-label accuracy, also called Jaccard measure, and finally F-measure. The results showed that a random chain order considering the constraints imposed by a Bayesian network with a simple tree-based structure could have very competitive results in terms of predictive performance and time complexity against related state of the art approaches.

Cerri et al. [53] investigated a new local-based classification method that incrementally trains a Multi-Layer Perceptron (MLP) for each level of the classification hierarchy. In a given level, a neural network makes predictions that are used as inputs to the neural network responsible for the prediction in the next level.

Hierarchical Multi-label Classification with Local Multi-Layer Perceptron (HMC-LMLP) is a local-based HMC method that associates one Multi-Layer Perceptron (MLP) to each classification hierarchical level. This method is basically designed to be used in tree-structured hierarchies. The method trains the MLPs incrementally on each level, and after the training process of one neural network for a specific level, the predictions of this network for the training dataset are used as inputs for the training of the next neural network associated with the next hierarchical level. This process is continuous until reaching the last level of the hierarchy.

The experiments used twelve free available [54] datasets associated with the task of protein function prediction. Precision–Recall curves (PR-curves) are used as the evaluation measure for the methods.

The proposed method results were compared with the results obtained by one state-of-the-art decision tree learner and two other decision-tree based methods, all three based on Predictive Clustering Trees (PCT). The experimental data showed that

HMC-LMLP can achieve competitive results compared with the global (state-of-the-art) version of the PCT-based methods, Clus-HMC, regarding $AU\overline{(PRC)}$.

Latent Dirichlet allocation (LDA) [55] is a generative probabilistic model. The LDA underlying idea is that documents are represented as random mixtures over latent topics, and each topic is characterized by a distribution over words. In the text classification, a document is classified into two or more classes. As in any classification problem, by using LDA module for each class, it could obtain a generative model for classification. LDA model is one of the most successful topic discovery models used in the statistical text analysis literatures as it uses plenty of words generative approach to automatically find topic for documents.

Cross-validation [56] is a technique that estimates how a specific classifier will generalize when used with a data set that is different than the one that the model has trained. It basically partitions the data into n subsets and then uses n-1 of the subsets for training the model, and the remaining set for testing the model. This procedure will be repeated n times so by this, each of the subsets is used as a testing set only once. Then, the results are averaged over the rounds to find the final estimation.

2.7 Performance Evaluation

In order to evaluate the multi-label classifiers, different methods than the ones specific to single-label problems are used because an example can be partially correct or incorrect [57]. According to [43], the measures used for evaluation of multi-label classification can be organized into two classes: i) bipartition based (includes example based measures and label based measures) and ii) ranking based (evaluates measures based on the ground truth of multi-label dataset). The example based measures evaluate the bipartitions over all examples of the evaluation dataset, while the label based measures divide the evaluation process into evaluations of each label [57].

In classical machine learning, the classifiers are usually evaluated by error-rate estimates. This error is obtained by comparing the testing examples having a pre-determined class labels with those class labels recommended by the classifier. This, however, is not quite enough when dealing with domains where one class significantly outnumbers the other [58]. For instance, if only 1% of the examples are positive, then a classifier that labels all examples as negative will achieve 99% accuracy.

For this latter case, other criteria are used, the most popular among them being precision and recall. Let us denote by TP the number of true positives, by FN the number of false negatives, by FP the number of false positives, and by TN the number of true negatives. ***Precision*** and ***recall*** (which are example based measures) are defined as follows:

$$Pr = \frac{TP}{TP+FP} \tag{1}$$

$$Re = \frac{TP}{TP+FN} \tag{2}$$

Precision is the percentage of truly positive examples among those labeled as such by the classifier; recall is the percentage of positive examples that have been recognized as such ("recalled") by the classifier. Which of the two is more important depends on the given domain. In order to combine them in a single formula, [59] proposed $\boldsymbol{F\beta}$, where the user-specified parameter, $\beta\epsilon[0,\infty)$, quantifies each component's relative importance:

$$F\beta = \frac{(\beta^2+1) \times Pr \times Re}{\beta^2 \times Pr + Re} \quad \textbf{(3)}$$

It would be easy to show that $\beta > 1$ apportions more weight to recall while $\beta < 1$ emphasizes precision. Moreover, F β converges to recall if $\beta \rightarrow \infty$, and to precision if $\beta = 0$. If we do not want to give more weight to either of them, we use the neutral $\beta = 1$:

$$F_1 = \frac{2 \times \mathrm{Pr} \times Re}{Pr+Re} \quad \textbf{(4)}$$

All this, however, applies only to domains where each example is labeled with one and only one class.

F-measure is the harmonic mean between precision and recall [52]**:**

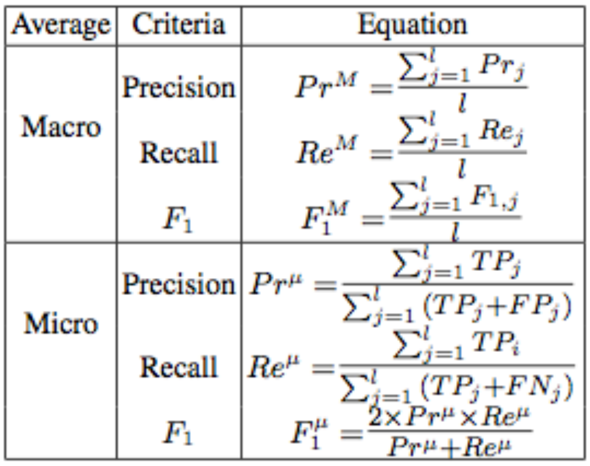
$$F - measure = \frac{1}{d}\sum_{j=1}^{d} \frac{2p_j r_j}{(p_j + r_j)} \quad \textbf{(5)}$$

where p_j and r_j are the precision and recall for C_j. Here, the F-measure is calculated per label and then averaged.

Yang [60] proposed two methods to average the above metrics over multiple classes: (1) ***macro-averaging***, where precision and recall are first computed separately for each class and then averaged; and (2) ***micro-averaging***, where precision and recall are obtained by summing over all individual decisions. Which of the two approaches is better depends on the concrete application. Generally speaking, micro-F_1 weighs the classes by their relative frequency, whereas macro-F_1 gives equal weight to each class. The formulas are summarized in Table 2.1, where Pr_j, Re_j, and

$F_{1,j}$, stand for precision, recall, and F_1 for the jth class (from l classes).

Table 2.1 Macro-averaging and micro-averaging of the performance criteria on the data set with l classes [60].



Average	Criteria	Equation
Macro	Precision	$Pr^M = \frac{\sum_{j=1}^{l} Pr_j}{l}$
	Recall	$Re^M = \frac{\sum_{j=1}^{l} Re_j}{l}$
	F_1	$F_1^M = \frac{\sum_{j=1}^{l} F_{1,j}}{l}$
Micro	Precision	$Pr^\mu = \frac{\sum_{j=1}^{l} TP_j}{\sum_{j=1}^{l} (TP_j+FP_j)}$
	Recall	$Re^\mu = \frac{\sum_{j=1}^{l} TP_i}{\sum_{j=1}^{l} (TP_j+FN_j)}$
	F_1	$F_1^\mu = \frac{2\times Pr^\mu \times Re^\mu}{Pr^\mu + Re^\mu}$

Hamming loss (an example based measure) [61] evaluates how many times an example-label pair is misclassified, i.e., label not belonging to the example is predicted or a label belonging to the example is not predicted. The smaller the value of *hamming_loss*(h), the better the performance. The performance is perfect when *hamming_loss*(h) = 0. This metric is defined as:

$$hamming_{loss(h)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{Q}|h(x_i)\Delta y_i| \qquad \textbf{(6)}$$

where Δ stands for the symmetric difference between the two sets, N is the number of examples and Q is the total number of possible class labels. *Yi* denotes the set of true labels of examples xi and h(xi) denotes the set of predicted labels for the same examples.

The ***ranking loss*** [62] evaluates the average fraction of label pairs that are disordered for the example. The metric is defined as:

$$Ranking\ loss(h, x, P_x) = \frac{|\{(\lambda,\lambda') \in P_x \times N_x | \tau(\lambda) > \tau(\lambda')\}|}{|P_x||N_x|} \quad \textbf{(7)}$$

where h is a ranking model, x is a given instance, P the set of relevant labels, $\tau\ (\lambda_i)$ denotes the position of λ_i in the predicted ranking , $\tau^{-1}(i)$ the label λ having assigned position i.

Set error [42] evaluates a multi-label prediction as a whole. It evaluates the percentage of predicted label sets that do not exactly match the true label sets.

The ***one-error*** [62] evaluates the performance from a restricted perspective, since it only determines when the top-ranked label is relevant. In this case, the best performance is reached when one-error is equal to 0, the smaller the value of the error, the better the classification algorithm.

$$One\ error(h, x, P_x) = \begin{cases} 1\ if\ \tau^{-1}(1) \notin P_x, \\ 0\ otherwise. \end{cases} \quad \textbf{(8)}$$

where h is a ranking model, x is a given instance, $\tau\ (\lambda_i)$ denotes the position of λ_i in the predicted ranking , $\tau^{-1}(i)$ the label λ having assigned position i, and P the set of relevant labels.

Average precision [42] evaluates, for each relevant label, the percentage of relevant labels among all labels that ranked above it. It can evaluate the algorithm as a whole and, unlike the case if one-error, the higher its value, the higher the performance is.

The ***coverage*** [63] evaluates on average how many steps are needed, to move down the label list in order to cover all the proper labels of the example. Along with one-error and average precision, this measure metric belongs to the ranking-based measure class.

$$Coverage(h) = \frac{1}{p}\sum_{i=1}^{p} \max_{y \in Y_i} rank^h (x_i, y) - 1 \qquad \textbf{(9)}$$

Here, h(xi) returns a set of proper labels of xi; h(xi,y) returns a real-value indicating the confidence for y to be a proper label of xi; $rank^h$(xi,y) returns the rank of y derived from h(xi,y).

Another error that can be used for multi-label classification is the Mean accuracy over the d class variables (accuracy per label) defined by Eq. 10 and Global accuracy over the d-dimensional class variable (accuracy per example, also called subset zero-one loss) defined by Eq. 11 [52].

$$M - Acc = \frac{1}{d}\sum_{j=1}^{d} Acc_j = \frac{1}{d}\sum_{j-1}^{d}\frac{1}{N}\sum_{i=1}^{N}\delta(c'_{ij}, c_{ij}) \qquad \textbf{(10)}$$

where $\delta(c'_{ij}, c_{ij}) = 1$ if $c'_{ij} = c_{ij}$ and 0 otherwise, and c'_{ij} denotes the C_j class value outputted by the model for instance i and c_{ij} is its true value.

$$G - Acc = \frac{1}{N}\sum_{i=1}^{N}\delta\,(c'_i, c_i) \qquad \textbf{(11)}$$

where $\delta\,(c'_i, c_i) = 1$ if $c'_i = c_i$ and 0 otherwise. Therefore, we call for a total coincidence on all the components of the vector of predicted classes c'_i and the vector of real classes c_i

In [64], a multi-label accuracy measure called ***Jaccard*** measure was defined:

$$ML - Acc = \frac{1}{N} \sum_{N=1}^{N} \frac{|c_i \wedge c'_i|}{|c_i \vee c'_i|} \quad \textbf{(12)}$$

where in the numerator, we count the number of coincidences of the two vectors (real and predicted), and in the denominator, we count the number of labels covered by some of both vectors.

In case of decision-tree methods, in order to obtain the final predictions, a threshold value is employed. When classifying an example, if the corresponding output value for a given class is equal or larger than the threshold, the class is assigned to the example. Otherwise, it is not assigned to the example. Thus, the choice of the "optimal" threshold value is a difficult task, because low threshold values lead to many classes being assigned to the examples, resulting in high recall and low precision. Moreover, larger threshold values lead to very few examples being classified, resulting in high precision and low recall. In order to deal with this problem, ***Precision–Recall curves*** (PR-curves) are used as the evaluation measure [53]. To obtain a PR-curve for a given classification method are applied to the outputs of the methods, and thus different values of precision and recall are obtained for each threshold. Each threshold represents a point within the PR-space. The union of these points forms a PR-curve, and then the area below the curve (*AUPRC*) is calculated. Different methods can be compared based on their areas below the PR-curves.

a) Area under the average PR-curve:

Given a threshold value, a precision–recall point $(\overline{Prec}, \overline{Rec})$ in the PR-space can be obtained using the following equations:

$$\overline{Prec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FP_i} \quad \textbf{(13)}$$

$$\overline{Rec} = \frac{\sum_i TP_i}{\sum_i TP_i + \sum_i FN_i} \qquad \textbf{(14)}$$

where i ranges over all available classes, corresponding to the micro-average of precision and recall.

b) Weighted average of the areas under the individual PR-curves:

In order to calculate the weighted average of the areas under the individual PR-curves, we first calculate the *AUPRCi* for each class separately, with *i* ranging from 1 to |*C*|. Afterwards, we obtain the $\overline{AUPRC_w}$using the following equation:

$$\overline{AUPRC_w} = \sum_i w_i . AUPRC_i \qquad \textbf{(15)}$$

where w_i is used to weight the contribution of a class according to its frequency.

Methodology

Digital libraries provide a huge range of information including text, movies, speeches, images, photos, books and others. This digital data provides large collections of content which naturally leads to the need of powerful tools that efficiently process, analyze, navigate, and browse the digital data [65]. Therefore, in this work, different data sets from books digital libraries and other contents were used. There are many digital libraries available online such as, Internet archive [66], Google books [67], Open library [68], The New York public library [69], and Wiley online library [70], Routers-21578 [71]. The Wiley Online Library [70] hosts the world's broadest and deepest multidisciplinary collection of online resources covering life, health and physical sciences, social science, and the humanities. Routers-21578 [71] is a collection appeared on the Reuters newswire in 1987.

From the available sources Wiley Online Library [70], Routers-21578 text categorization collection data set [71], and the 20 Newsgroups data set [72] were chosen for the conducted experiments. Wiley Online Library hosts the world's broadest and deepest multidisciplinary collection of online resources. It delivers seamless integrated access to over 4 million articles in 1500 journals, over 14,000 online books, and hundreds of reference works, laboratory protocols and databases.

The documents in Routers-21578 [71] are organized and indexed with categories by personnel from Reuters Ltd. In 1990, Reuters and CGI made the documents available for research purposes to the Information Retrieval Laboratory of the Computer and Information Science Department at the University of Massachusetts at Amherst. There are multiple categories, and there are relationships among the categories, therefore are many possible feature sets can be extracted from the text

The 20 Newsgroups data set [72] is a collection of approximately 20,000 newsgroup documents, partitioned across 20 different newsgroups, each corresponding to a different topic. It has become a popular data set for experiments in text applications of machine learning techniques, such as text classification and text clustering.

Since the data set that is provided by the digital library is considered as raw data, it may contain nominal attributes (un-necessary). Nominal attributes are defined by providing a <nominal-specification> listing the possible values: {the, for, in, on, edition, processes, systems...}. Also, a raw data set may contain many values that may be missing, so it is necessary to do some pre-processing. Once pre-processing was finished, a proprietary algorithm for multi-label class was implemented and compared with some existing algorithms.

3.1. Data pre-processing:

This phase consists of the following: i) data cleaning; ii) feature extraction; and iii) nominal to numerical conversion.

3.1.1 Data cleaning:

Removing un-necessary and meaningless words such as “introduction”, “handbook”, “edition” etc., is done in this stage. Its role is to reduce the dimensions of the dataset and to eliminate the elements that can create errors in the classification algorithm.

Meaningless words with very high frequency are considered as stop words [73], and these words are added to the Stop Word list. Removing such words will result in better results and it will not affect the classification efficiency at the same time. The Stop Word list are shown in Table 3.1.

Table 3.1: The list of Stop Word used in the classification procedure

a	couldn't	his	or	third
about	cry	how	other	this
above	de	however	others	those
across	describe	hundred	otherwise	though
after	detail	i	our	three
afterwards	do	ie	ours	through
again	done	if	ourselves	throughout
against	down	in	out	thru
all	due	inc	over	thus
almost	during	indeed	own	to
alone	each	interest	part	together
along	eg	into	per	too
already	eight	is	perhaps	top

also	either	it	please	toward
although	eleven	its	put	towards
always	else	itself	rather	twelve
am	elsewhere	keep	are	twenty
among	empty	last	same	two
amongst	enough	latter	see	un
amongst	etc	latterly	seem	under
amount	even	least	seemed	until
an	ever	less	seeming	up
and	every	ltd	seems	upon
another	everyone	made	serious	us
any	everything	many	several	very
anyhow	everywhere	may	she	via
anyone	except	me	should	was
anything	few	meanwhile	show	we
anyway	fifteen	might	side	well
anywhere	fifty	mill	since	were

are	fill	mine	sincere	what
around	find	more	six	whatever
as	fire	moreover	sixty	when
at	first	most	so	whence
back	five	mostly	some	whenever
be	for	move	somehow	where
became	former	much	someone	whereafter
because	formerly	must	something	whereas
become	forty	my	sometime	whereby
becomes	found	myself	sometimes	wherein
becoming	four	name	somewhere	whereupon
been	from	namely	still	wherever
before	front	neither	such	whether
beforehand	full	never	system	which
behind	further	nevertheless	take	while
being	get	next	ten	whither
below	give	nine	than	who

beside	go	no	that	whoever
besides	had	nobody	the	whole
between	has	none	their	whom
beyond	Hasn't	noone	them	whose
bill	have	nor	themselves	why
both	he	not	then	will
bottom	hence	nothing	thence	with
but	her	now	there	within
by	here	nowhere	thereafter	without
call	hereafter	of	thereby	would
can	hereby	off	therefore	yet
cannot	herein	often	therein	you
cant	hereupon	on	thereupon	your
co	hers	once	these	yours
computer	herself	one	they	yourself
con	him	only	thick	yourselves
could	himself	onto	thin	

In data cleaning, the input file is parsed line by line and each line is being split into words by space character as a delimiter. Then each is getting its stem using the Porter stemming algorithm [74].

The Porter stemming algorithm is a process for removing the commoner morphological and inflexional endings from words in English. It is mainly used as part of the normalization process that is usually done during processing information retrieval systems. After the stemming process, each root is being searched in the list of unwanted words and if that root exists in the unwanted words file, then the word it's derivatives will be deleted from the input file. Finally, the line that has unwanted words eliminated is reconstructed and pushed in a new file (Intermediate). The scheme of this algorithm is presented in Figure 3.1.

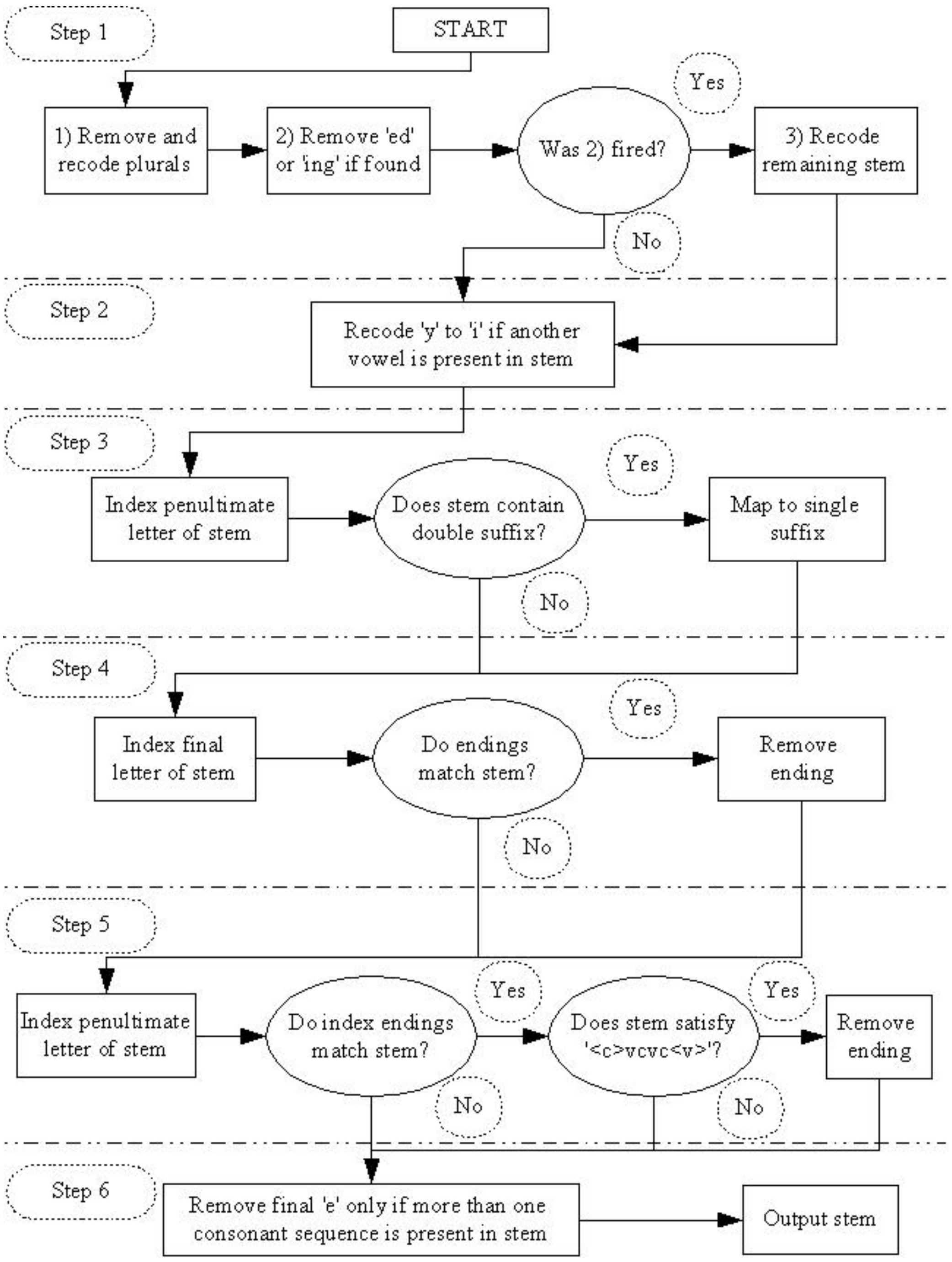

Step 1
START
1) Remove and recode plurals
2) Remove 'ed' or 'ing' if found
Was 2) fired?
Yes
3) Recode remaining stem
No
Step 2
Recode 'y' to 'i' if another vowel is present in stem
Step 3
Index penultimate letter of stem
Does stem contain double suffix?
Yes
Map to single suffix
No
Step 4
Index final letter of stem
Do endings match stem?
Yes
Remove ending
No
Step 5
Index penultimate letter of stem
Do index endings match stem?
Yes
Does stem satisfy '<c>vcvc<v>'?
Yes
Remove ending
No
No
Step 6
Remove final 'e' only if more than one consonant sequence is present in stem
Output stem

Figure 3.1: Flowchart of the Porter Stemming algorithm [81]

3.1.2 Feature extraction:

Transforming the input data into the set of features is called feature extraction. The features have to be chosen carefully. By that, the features set will extract the relevant information from the input data in order to perform the desired task using this reduced representation instead of the full size input [75].

In features extraction, the process starts by reading the intermediate file line by line. Then, we find the stem of each word by searching in the stem file. If the stem is found then the word is ignored and the process forwards to the next word. If that stem is not found, then the count of features is increased by one and that stem is added into the feature set. After that, that original word of that stem is written in the output file. In order to find the relationship between the number of features and number of example, the feature in this step are counted.

3.1.3 Nominal-to-numerical conversion:

To make the classification less computational expensive, the classes are numbered and their corresponding meaning are defined. Also the extracted features are transformed into numerical features usable for machine learning.

Figure 3.2 shows an example of the pre-processing phases, where un-necessary word (Handbook) is removed in the data cleaning and the remaining words are extracted as representing the features. Figure 3.3 provides sample of pre-processed training examples where the numbers before the “:” represents the book title features and the numbers after the “:” represents the three level class representation corresponding to the book titles.

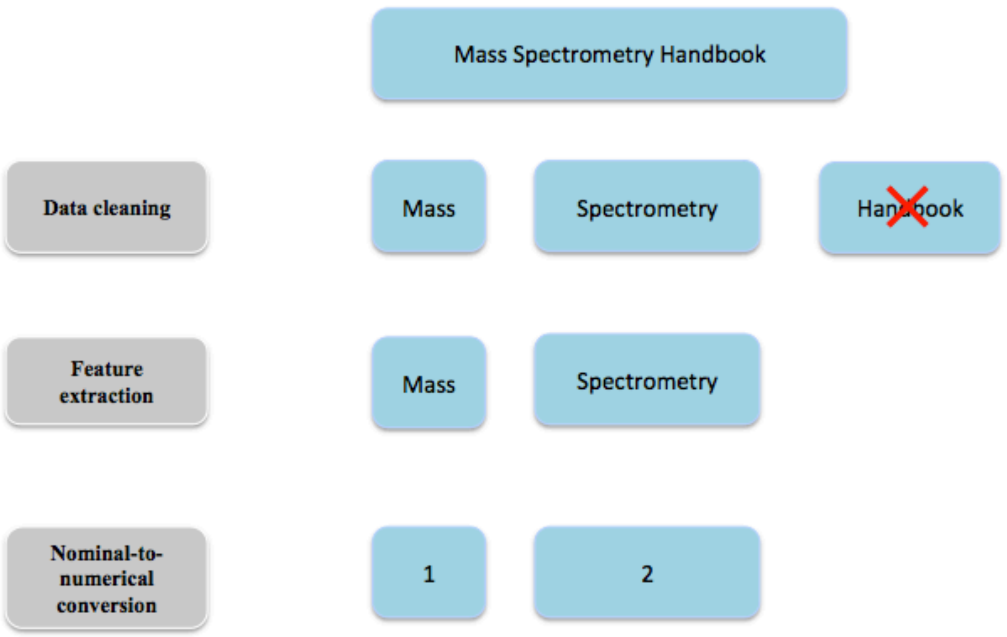
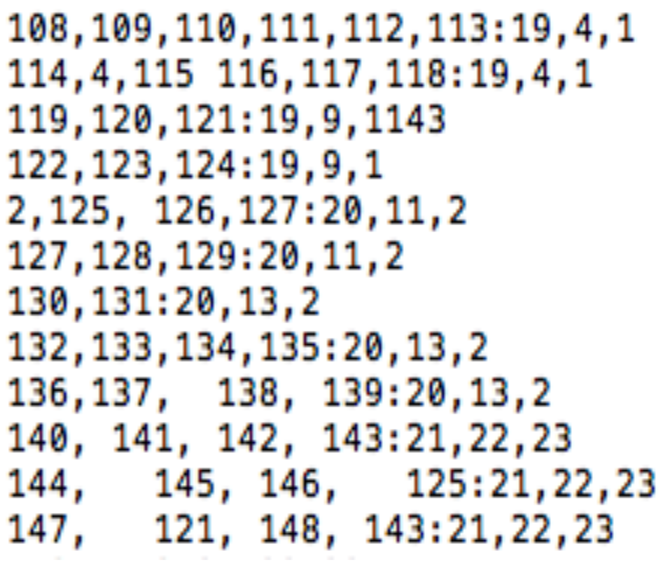


Figure 3.2: Pre-Processing phases on an example



```
108,109,110,111,112,113:19,4,1
114,4,115 116,117,118:19,4,1
119,120,121:19,9,1143
122,123,124:19,9,1
2,125, 126,127:20,11,2
127,128,129:20,11,2
130,131:20,13,2
132,133,134,135:20,13,2
136,137,  138, 139:20,13,2
140, 141, 142, 143:21,22,23
144,   145, 146,   125:21,22,23
147,   121, 148, 143:21,22,23
```

Figure 3.3: Sample of pre-processed training examples

3.2 The proposed algorithm

Once data pre-processing is completed, the data is stored in a pre-processed file to be handled later as clean data. The proposed classifiers read from this data set as follow: First, the system reads a set of the data set X. Then, the system reads another set of data, let's call it Z. This data is used for testing. After the classification process is done, the error rate and the classification accuracy will be observed.

Let's call the set of misclassified examples Y. Then classifier must be trained again. Because Y is smaller than Z a number of examples (E) must be added, where $E = Z - Y$ and the new training set is N where $N = E + Y$. The following example clarifies how (N) is computed. If we have X=1000 examples for training, Z=1000 examples for testing, and we came out with Y=200 misclassified examples. So, we need to include Y in the next training session. But because $Y < Z$, then we must to add new set of data (E) where $E = Z - Y$ and then we add Y to E to form (N).

The system reads another set of data, let's call it (V), and this is going to be used for testing, so every example in the testing iteration $i \in V_i$.

For every classification iteration, a training session will start again and a new testing session V_i will also go through the classifier. In Figure 3.4 a simplified schema of the proposed approach is presented.

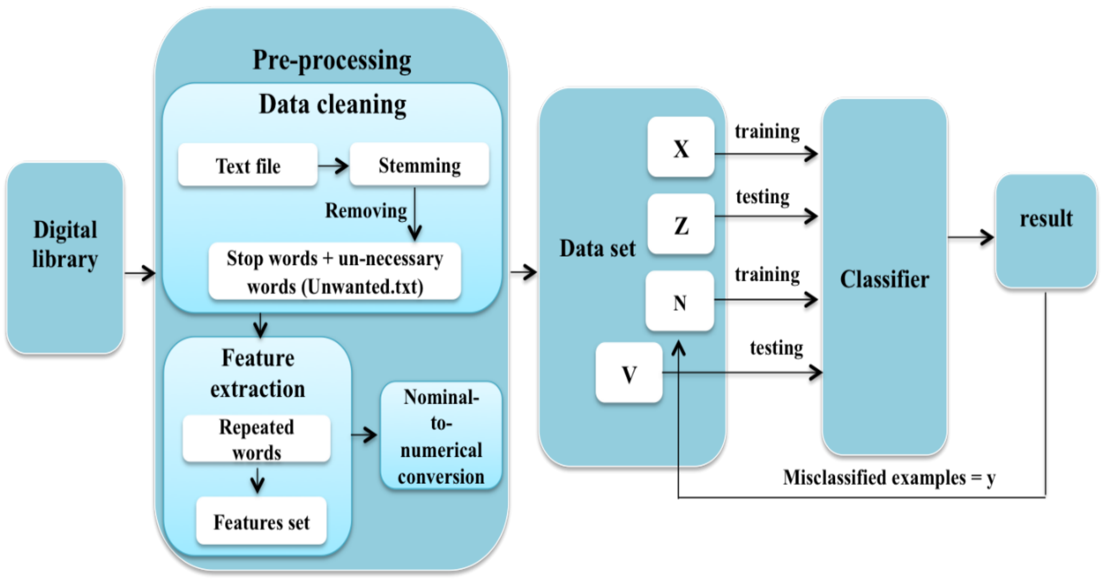


Figure 3.4: Methodology work flow

A common approach for building a reliable classifier is to split a data set in to a training set and an independent test set, where the training set is used to develop the classifier and the testing set is used to evaluate its performance. The common used strategy is allocating 2/3rd of cases for training is nearly optimal for reasonable sized data sets ($n \geq 100$) with strong signals [76]. According to this principle the workflow is as following:

- Once the data it is cleaned, the algorithm reads it.

- The algorithm trains the classifier by assigning the feature numbers with every class in the classification tree.

  For example if data has a set of features (Computer = 1, Science = 2, Machine = 3, Learning = 4, Algorithm = 5, Engineering = 6, Biology = 7, Chemistry = 8), these features are assigned to each class according to a pre-designed classification tree:

Class 1, features [1, 2, 3, 5, 8, 10]

Class 2, features [1, 5, 8]

Class 3, features [5, 8, 10]

and so on.

Then, the classification tree might look as Figure 3.5.

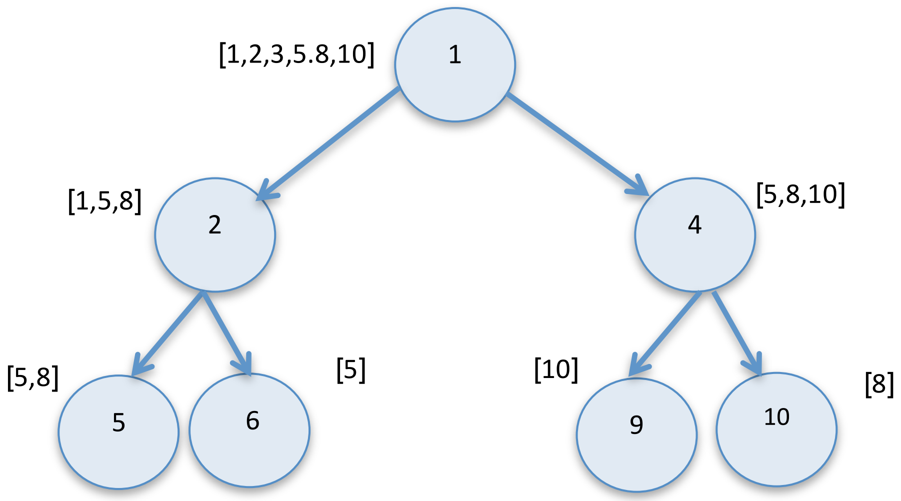


Figure 3.5: Classification tree

- Once the features are assigned to classes, the testing set is introduced, every word in the title being assigned to a class. The word might be assigned to more than one class, but only to the ones belonging to the same grandparent. Once a word cannot be classified in any class, it means, the feature of the word is new, and the classifier needs to be re-trained. The error rate is calculated if a word feature was miscounted or if a word was classified in a wrong class. Macro- and Micro- averaging are used to calculate the error rate in case a word was classified into a wrong class.

The pseudo-code of the proposed algorithm is the following:

```
TitleList = Import_Titels_List( )
CategoriesList  = Import_Categories_List( )
StemList = Import_Words_List( )
ErrorsCounter = 0
WordsCounter = 0
For each title in TitleList
    WordsIn Title = extractWordsFromTitles(title)
    WordsCounter = WordsCounter + NumberOfElement(WordsIn Title)
    TitleCategories(title) = emplylist( )
        for each wordintitle in WordsInTitle
                for each stem in StemList
                    StemIsFound = false
                    if StemOf (wordintiltle) == stem
            TitleCategories(titles) = union( TitleCategories(title), CategoriesOfStem(stem))
                        StemIsFound = true
                        GoToNextWordIntitle( )
                    end
                end
                if StemIsFound == false
OutputWarning("The word"wordintitle "in the title"title "has not  a matching in the list of stems")
                        ErrorsCounter = ErrorsCounter + 1
               end
        end
end
ErrorRate = ErrorCounter / WordsCounter
Output("The error rate is" ErrorRate)
```

3.3. Complexity analysis

To analyze the complexity of the algorithm the following symbols are used:

n : number of titles to be analyzed

m : number of stems present in the database

t : number of categories per word (mean value)

p : number of words per title (mean value)

q : number of characters per word (mean value)

w : total number of categories

The overall number of instructions is

$$f(n,m,t,p,q) = npq + mp + n\big(pm(t+t+q)\big) + p + pq \qquad \textbf{(16)}$$

That is in the expanded form

$$f(n,m,t,p,q) = mnpq + 2mnpt + mp + npq + pq + p \qquad \textbf{(17)}$$

Analyzing the expression above we can note that the increasing the size of inputs the dominants terms are $mnpq$ and $2mnpt$. Then, considering that the number of categories per word (t) is generally lower than the number of characters per word (q) the time-complexity of the algorithm is $O(mnpq)$.

Considering that the number of word per title (p), the number of characters

(q) and the number of categories per word (t) does not increase by increasing the input size (as they mainly depend on the language the words belong) they can be treated as constant (the medium value is considered) and can be neglected in the evaluation of the time - complexity of the algorithm.

In the end, the time complexity of the algorithm result using the big-O notation.

$$O(mn) \tag{18}$$

The space complexity of the algorithm is calculated considering the bytes of memory needed for the execution of the algorithm. Therefore, the number of bytes is defined by Equation 19, described in its extended form by Equation 20:

$$f(n,p,q,m,w) = npq + m(p+q) + pq + wq \tag{19}$$

$$f(n,p,q,m,w) = npq + mp + mq + pq + wq \tag{20}$$

Considering that the number of categories (w) is generally lower than the number of titles to be analyzed (n) and lower than the number of stems (m), in the asymptotic analysis the last term (wq) can be neglected. In these conditions, the space required when increasing n and q can be approximated as:

$$f(n,p,q,m,w) = npq + mp + mq \tag{21}$$

Considering that the number of words per title p and the number of characters per word q do not increase increasing the input size (as mentioned in the previous paragraph, they mainly depend on the language used), the space required can be approximated to:

$$f(n,p,q,m,w) = npq + mp + mq = npq + m(p+q) \quad \textbf{(22)}$$

In the end, the space complexity of the algorithm result using the big-O notation

$$O(n+m) \quad \textbf{(23)}$$

3.4. Case studies

The same experiment is conducted for full domain, and sub-domains of the library collections which are shown in Figure 3.6.

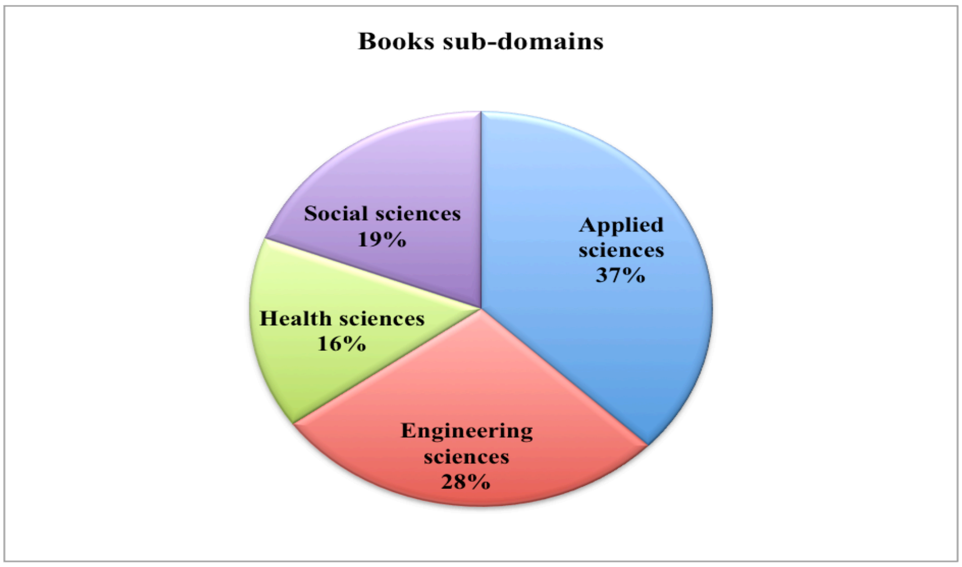


Figure 3.6: Books domain and sub-domains

After finishing with the books domain, the algorithm is applied to another domain such as animal domain (Figure 3.7).

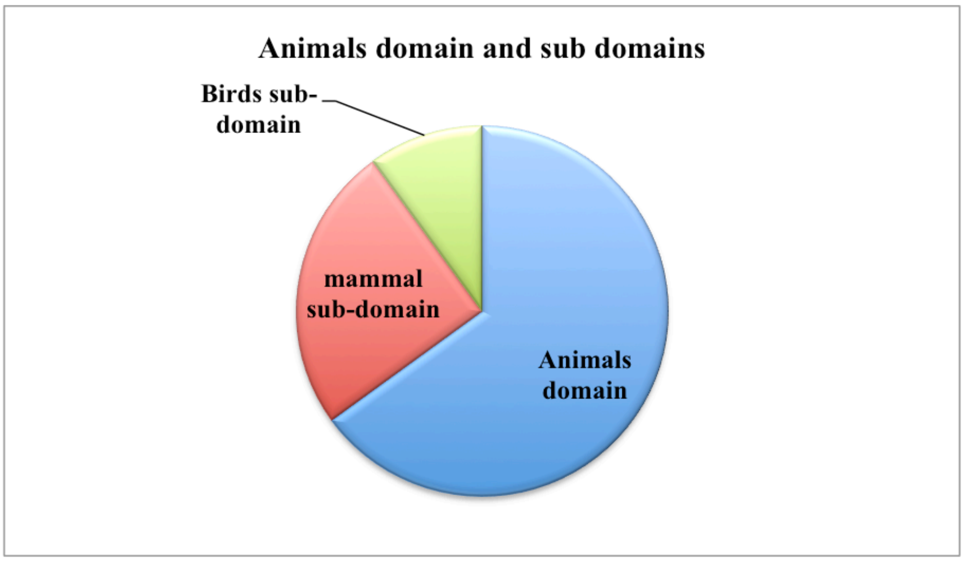


Figure 3.7: Representing animal domain and sub-domains

In order to assess the performance of the proposed algorithm, a comparison with Naïve Bayes and K-Nearest Neighborhood is performed.

3.4.1. Case study one (Wiley online library)

The data set of the first case study is represented by the Willey online library [70]. It has collection of books (examples) described by different attributes. These books were collected from different fields and disciplines. The characteristics of this database are the following:

- Dataset name: Wiley online library [57]
- Number of attributes: 5888
- Number of examples: 8842
- Number of classes: 64

- Number of hierarchical levels: 3

The data set already contains nominal attributes, many values were missing. Therefore, pre-processing was necessary. According to the workflow of the proposed algorithm, before training and testing, a data cleaning step and nominal to numerical conversion steps are performed.

In the data cleaning step, rare classes or classes that may have a representation of less than 1% of the data set will be ignored as 1% is really a small number of examples. In case the data set is a large one, 1% might be taken into consideration in other data sets. Some examples of books titles that might be ignored due to the low class representation are: “It Happened One Night”, “Top Hat”, “Hairspray”, “The Act of Remembering”, and “Women at the Top”.

In the nominal to numerical conversion, numbers are manually assign to each class as those classes are already induced and defined to their corresponding meaning. Also, the transformation of the extracted features into numerical features is useful for machine learning since its easy to handle when coding. For example if we have the word “science = 1” in the feature set and we got a book titled with “computer science algorithms”, the word “computer” will be assigned to 2, and the word “algorithms” will be assigned to 3 as we already have “science” assigned to 1. So, the example representation will be: “2,1,3.”

The data set has thousands of examples. To insure precise performance evaluation, a 5-fold cross validation was used. The training examples are described by thousands of attributes, thus it becomes easy to classify discriminant classes, but that means that a large number of examples is required in this case.

This data is cleaned and all un-necessary words and stop words are removed. The result of this phase is shown in table 3.2.

Table 3.2: Sample of cleaned books titles

Mass Spectrometry
Ray Powder Diffractometry
Ray Fluorescence Spectrometry
Reflection ATR Spectroscopy
HPLC
Electrospray MALDI Mass Spectrometry Biological
Forensic Chemistry
Chiroptical Spectroscopy Simulations
Chiroptical Spectroscopy Stereochemical Biomolecules
Chemistry
Atomic Microscopy
Condensed Molecular Spectroscopy Photophysics

Thus, using stemming, every word is associated with its family. Consequently, words like computer, computing, computers, and compute will have only one stem number (Table 3.3).

Table 3.3: Sample of the words and their associated stem numbers.

Word	**Stem number**	**Word**	**Stem number**
mass	1	Biological	14
Spectrometry	2	Chemistry	15
ray	3	Chiroptical	16
powder	4	Simulations	17
Diffractometry	5	Stereochemical	18
Fluorescence	6	Biomolecules	19
Reflection	7	Atomic	20
ATR	8	Microscopy	21
Spectroscopy	9	Condensed	22
HPLC	10	Molecular	23
Electrospray	11	Photophysics	24
Forensic	12	ADME	25
MALDI	13	Drug	26

We select $x_1, x_2 \dots . x_n \in X$, and X is a set of examples that consists of n examples for training. In the training stage, the features are manually assign with each class. This is called a class feature vector. Once this stage is achieved, the classifier is trained and becomes ready for testing.

For this case study, the scope was to test if the classifier can be trained and what would be the error rate. The dataset considered is represented by the entire book domain [70]. The entire data set including all major classes and sub-classes was used. The main classes are: Applied Science, Engineering Science, Health and Social Sciences.

The data set was divided into training and testing examples each training set having 200 examples and each testing 200 examples. For the training set, the examples are manually classified and the class label is updated with every example. Figure 3.8 shows the feature extraction workflow.

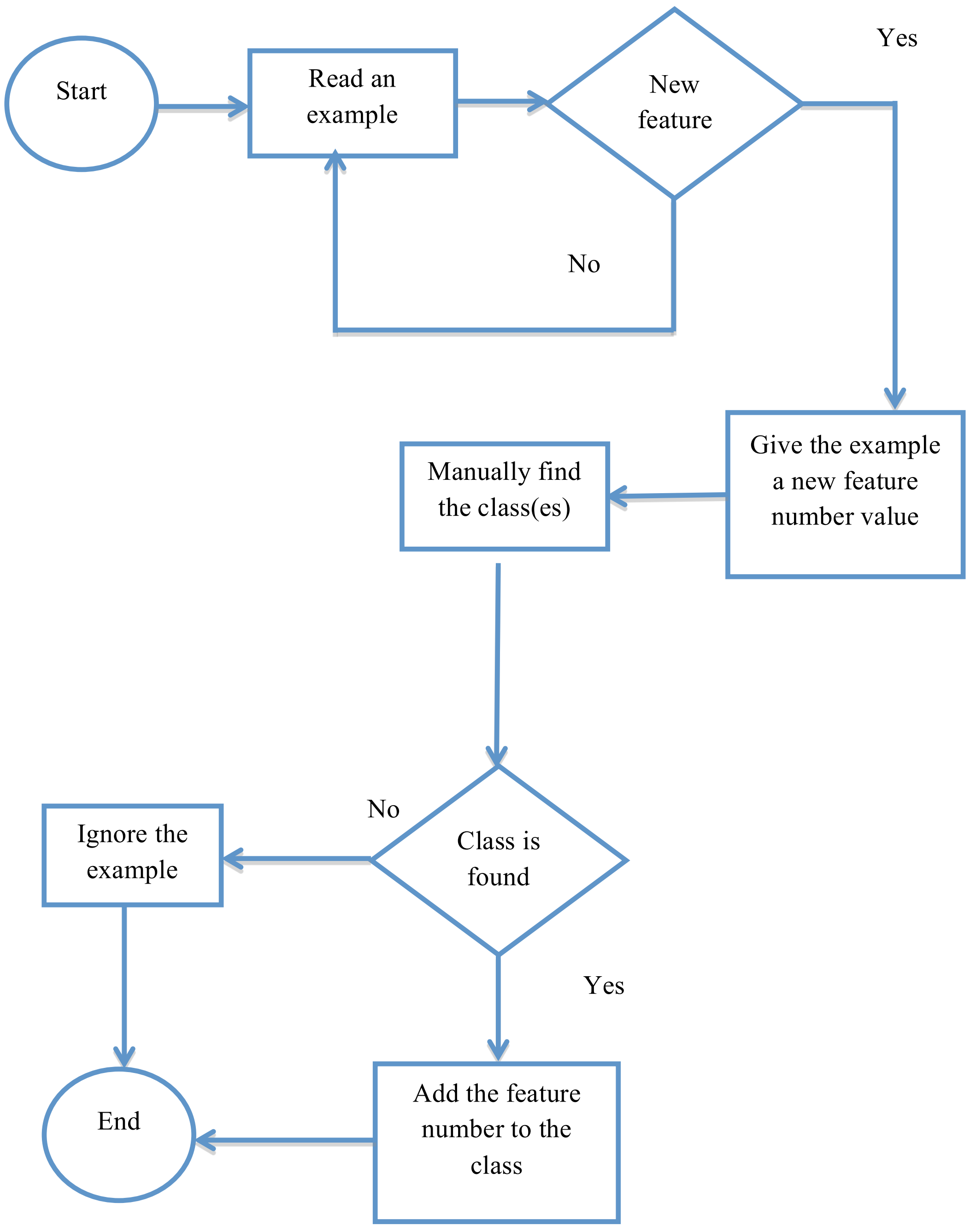

Start
Read an example
New feature
Yes
No
Give the example a new feature number value
Manually find the class(es)
No
Ignore the example
Class is found
Yes
End
Add the feature number to the class

Figure 3.8: Feature extraction workflow

3.4.2. Case study two (Engineering domain)

The structure of the engineering domain is presented in Table 3.4.

Table 3.4: Structure of the Engineering domain

Engineering science	Engineering	Chemical Engineering
		Civil & Construction Engineering
		Communication Technology & Networks
		Computer Science & Information Technology
		Electrical & Electronics Engineering
		Industrial Engineering
		Mechanical Engineering
		Mobile & Wireless Communications

Very similar experiment setup will be used as the one used in case study one, but the data is different. In this case, we will only apply the engineering domain. The purpose is to observe if there is any change in the result compared with case one.

3.4.3 Case study three (Social Sciences and Humanities domain)

In this case, the same data set as in the case study two was used [70]. However, only the social sciences and humanities books collection domain was studied. The structure of the database is presented in Table 3.5.

Table 3.5: Structure of the Social Sciences and Humanities domain

Social Sciences	Social Sciences & Humanities	Ancient History & Classical Studies
		Anthropology & Archaeology
		Architecture & Planning
		Business, Economics, Finance, Accounting
		History
		Language & Linguistics
		Literature
		Philosophy
		Public Administration & Management
		Religion & Theology
		Sociology, Media, Communications, & Cultural Studies

3.4.4 Case study four (Health science domain)

The Health Science domain has a number of 1300 examples from the same data set that we are using [70]. The structure of this domain is presented in Table 3.6.

In this experiment, the scope was to explore what happens (in terms of percentages of obtaining new features in every news groups) if we introduce sequence of examples in groups of 100-200 examples for each group.

Table 3.6: Structure of the Health Science domain

Health	Health Sciences	Allergy & Respiratory Medicine
		Anatomy & Physiology
		Cardiology & Cardiovascular Medicine
		Clinical Psychology
		Dentistry
		Dermatology
		Endocrinology & Diabetes
		Gastroenterology & Hepatology
		Hematology
		Neurology
		Neuroscience
		Nursing
		Obstetrics & Gynecology
		Oncology & Radiotherapy
		Pharmacology
		Psychiatry
		Psychology
		Public Health/General
		Surgery
		Veterinary Medicine

3.4.5 Case study five (Physical Science domain)

This domain has 1200 examples. The structure of the Applied Science area (including the Physical Sciences domain) is presented in Table 3.7. Similarly to the case study, in this experiment, the scope was to explore that happens (in terms of percentages of obtaining new features in every news groups) if we introduce sequence of examples in groups of 100-200 examples for each group. Distinctively from case study four, different domains were used. Therefore, the experiment will show (when the domain is changed), if there will be the same percentage of getting new features with every new group or no.

Table 3.7: Structure of the Applied Science area

Applied science	Chemistry	Analytical Chemistry
		Biochemistry
		Environmental Chemistry
		General & Physical Chemistry
		Industrial Chemistry
		Inorganic Chemistry
		Organic Chemistry & Catalysis
		Pharmaceutical & Medicinal Chemistry
	Physical Sciences	Energy
		Food Science & Technology
		Materials Science

		Mathematics
		Nanotechnology
		Physics
		Polymer Science & Technology
		Statistics

3.4.6 Case study six (Analytical Chemistry domain)

The Analytical Chemistry domain has a number of 250 examples. Similar to the previous two case studies (four and five), in this experiment, was explored the situation in which sequence of examples in groups of 50-100 examples for each group are introduced.

3.4.7 Case study seven (Routers-21578)

In this case the Routers-21578 data set [71] is used. It is represented by a collection of documents (news articles), and the documents are classified into classes. In this experiment, the scope was to use the whole text in the document, and not only the title of each document. Although this approach (due to the dimensionality) may require a longer computational time, it will show if the number of features to number of examples would have a direct impact on the error rate.

3.4.8 Case study eight (20 newsgroup)

In this experiment, the "20 newsgroup" dataset was employed. It has about 19,000 documents, the full document text, not only the titles being used for classification. Section 4.1.10 discuss the result of this experiment in details. Very similar to the previous experiment "case study seven," in this experiment we want to use the whole text in the document, and not only the titles of each documents, the difference being that the algorithm is run on a different data set.

3.4.9 Case study nine

In this experiment, we are looking at the error rate in general. The scope was to check if all class hierarchy has the same error rate or different error rate. In addition, the distribution of the error rate over class hierarchy was shown. In other words, it was tested if the error rate in the parents classes are similar or different than the error rate in the children or grandchildren classes. Therefore, two cases were considered: i) testing the error rate at a specific error rate during the training-testing phase; and ii) taking the readings for the whole experiment from start to finish.

3.5 Existing algorithms

In order to assess the performance of the proposed algorithm, a series of existing algorithms (Naïve Bayes and K-NN) were implemented and compared in terms of performance and efficiency.

3.5.1 Naïve Bayes algorithm

This method is based on the Bayes theorem and is a simple probabilistic classifier. It is suited for high dimensionality inputs, having good performance. If it is used with text classification, we need to calculate the error rate on the same data set and compare it with our suggested algorithm.

3.5.2 K-NN algorithm

The very popular K-NN is compared in this part with our algorithm.

Results

In this chapter, the results of the simulations performed with the proposed algorithm and the algorithms chosen for comparison are presented and discussed. Several experiments were conducted on real world data sets from different fields including library collections [70], Reuters- 21578 [71], and 20 Newsgroup [72] data sets. The final goal was to correctly classify a library collection into classes where the examples (books) are classified into classes and the classes are hierarchically ordered.

4.1 The proposed algorithm

4.1.1 Feature extraction

If a new word is received and the word is not found in any class vector, then this is considered as an error. The follow pseudo code calculates the error rate and do the training and testing steps.

```
Data = importdata('Data.xlsx'); // n*p*q + m*(p+[q])
errcnt = 0; //1
wrdscnt = 0; //1
for i=1:length(Data.Titles)
    WordsInTitle=regexp(Data.Titles{i}, ' ', 'split'); //p*q
    WordsInTitle=setdiff(WordsInTitle,{''});
    wrdscnt = wrdscnt + length(WordsInTitle);
    TitleCategories{i}=[]; //0
    for j=1:length(WordsInTitle)
        for k=1:length(Data.Words)
```

```
                    found=0;
                    match=strncmpi(WordsInTitle{j},Data.Words{k},length(Data.Words
                    {k}));
                    if match==1
                            TitleCategories{i}=union(TitleCategories{i},Data.Words(
                                                            k,2:end)); //w*q
                            TitleCategories{i}=setdiff(TitleCategories{i},{''});
                            found=1;
                            break;
                    end
            end
            if found==0
            warning(['The word ',WordsInTitle{j},' in the title ',num2str(i),' has not a
                            matching in the list of stems']) //1
            errcnt = errcnt +1; //1
            end
    end
end
errrate = errcnt / wrdscnt; // 1
disp(['The error rate is ', num2str(errrate*100) ,' %']);
```

4.1.2 5-fold cross validation

Along with this approach, the 5-fold cross validation procedure was also tested. Let's consider a scenario in which 250 examples are classified in order to estimate the error rate. This classification is performed by taking 50 examples at a time. The first iteration contained a number of 87 features in the feature set. In the second iteration, 65 new features were obtained. Having new features that do not exist in the features set will result in examples miss-classification, and this occurs during every iteration. Therefore, the new features were added manually to the feature set, so these new features will be assigned automatically in the feature set in the following classification iteration. By this, it can be noticed that the error rate is dropping with each iteration (Figure 4.1).

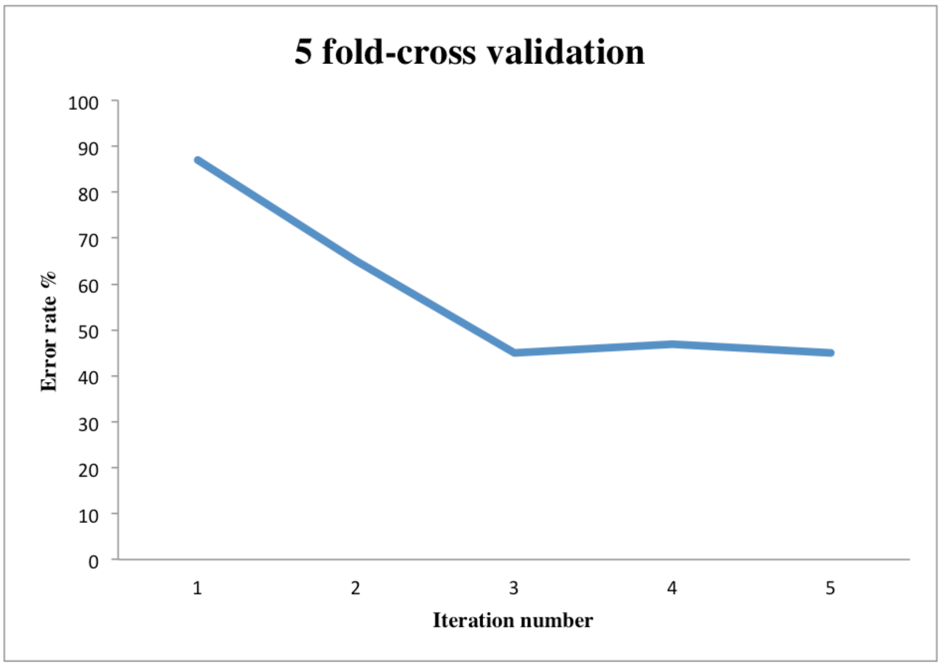


Figure 4.1: 5 fold – cross validation

4.1.3 Case study one (Wiley online library)

In order to compute the error rate for this dataset, the macro-micro averaging were used. When multiple class labels are to be retrieved, averaging the evaluation measures can give a view on the general results. For example, consider a binary evaluation measure B(TP,TN,FP,FN) that is calculated based on the number of true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN), and 2 labels c_1 and c_2. In this case, the metrics are the ones presented Table 4.1.

Table 4.1: Macro-Micro averaging measures

Label	TP	FP	FN	Precision	Recall
c_1	20	20	20	0.5	0.5
c_2	80	20	20	0.8	0.8
Total	100	40	40		
Macro-averaged = 0.65 Micro-averaged = 0.71					

Figure 4.2 shows the result of the experiment with a very noisy training, until 1000 examples are reached. At this point, the error rate starts to drop below 80%. The learning curve shows an error rate of 40% when almost 4000 examples are used. We expect that the curve will keep improving and the error rate keeps dropping as more examples are added. This experiment was conducted on the whole books domain. The number of attributes used in this experiment was 8555.

Figure 4.2 shows the result of applying the proposed algorithm, where every class is labeled with the features that best describe the class. These all the examples were classified to all classes that may have their features as shown in Figure 3.5

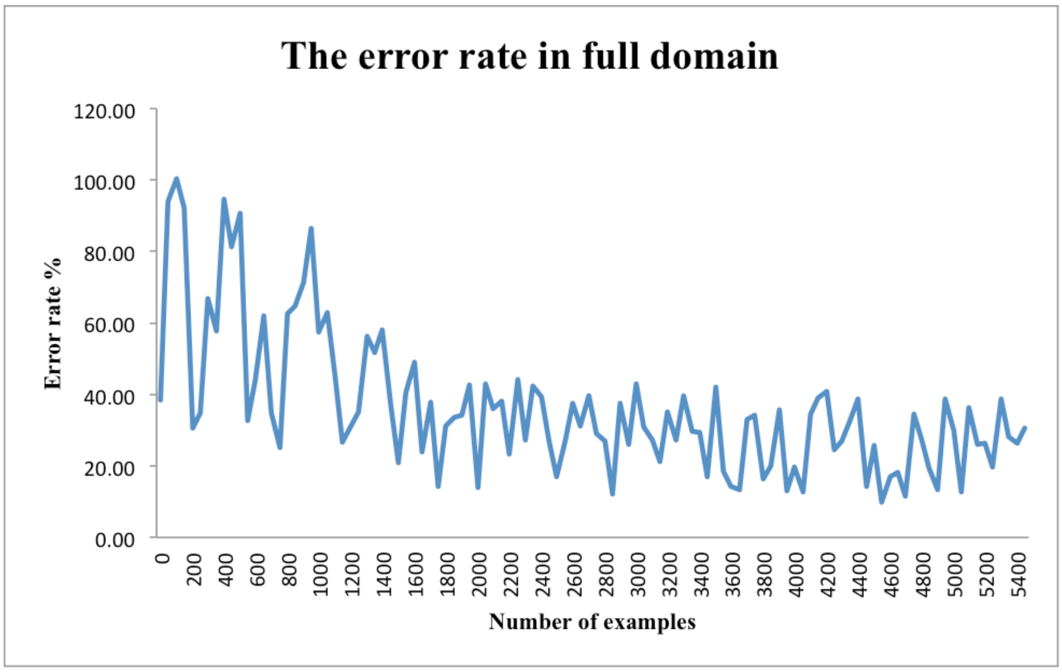


Figure 4.2: Books full domain error rate.

This algorithm works as follow:

- Read a data set.
- Perform data pre-processing phase.
- Extract the features, which are the attributes. If the feature is exist in any class, then the word is classified in to that class, else, if it does not have a class, then it is considered as an error and we classify it manually.

4.1.4 Case study two (Engineering domain)

With this sub-domain, the error rate drop below 20% with only 400 examples. This can be explained by the fact that the number of attributes associated with this domain is very small and equals to 1831 for the engineering domain. The error rate was calculated using the Macro-Micro averaging metric. Figure 4.3 shows the result of this experiment.

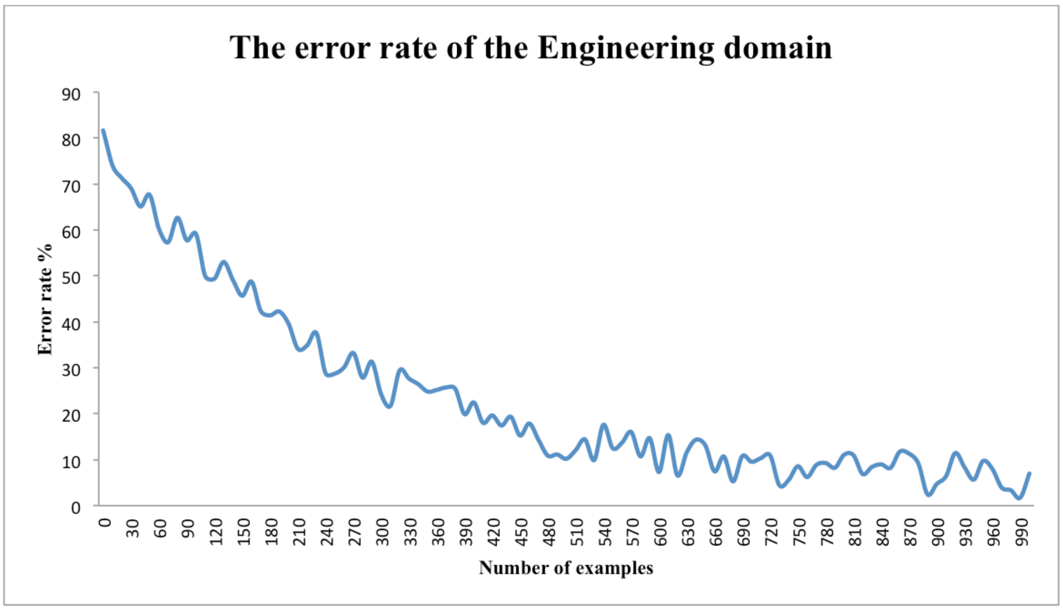


Figure 4.3: Engineering domain error rate

4.1.5 Case study three (Social sciences and humanities domain)

Figure 4.4 shows the output of this case study. As observed, the error descends rapidly due to the fewer number of features. In this experiment, we also used the Macro-Micro averaging metric to calculate the error rate.

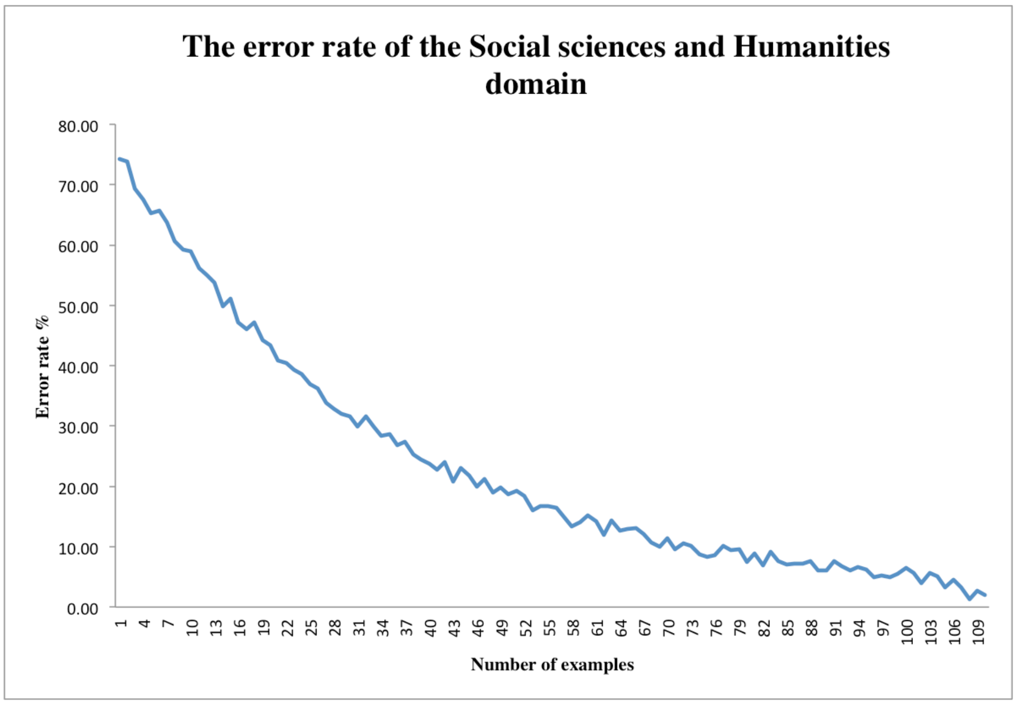


Figure 4.4: Social sciences and Humanities error rate.

4.1.6 Case study four (Health domain)

Considering health sciences domain, which have a number of 1300 examples and taking 100 examples on each iteration, the result shows an obvious decrease in the number of features whenever new set of examples are added (Figure 4.5).

The Figure 4.5 shows that, the first 100 examples has 154 new features then the second 100, shows 152 new features, and the third 100, shows another new features of 125, and so on.

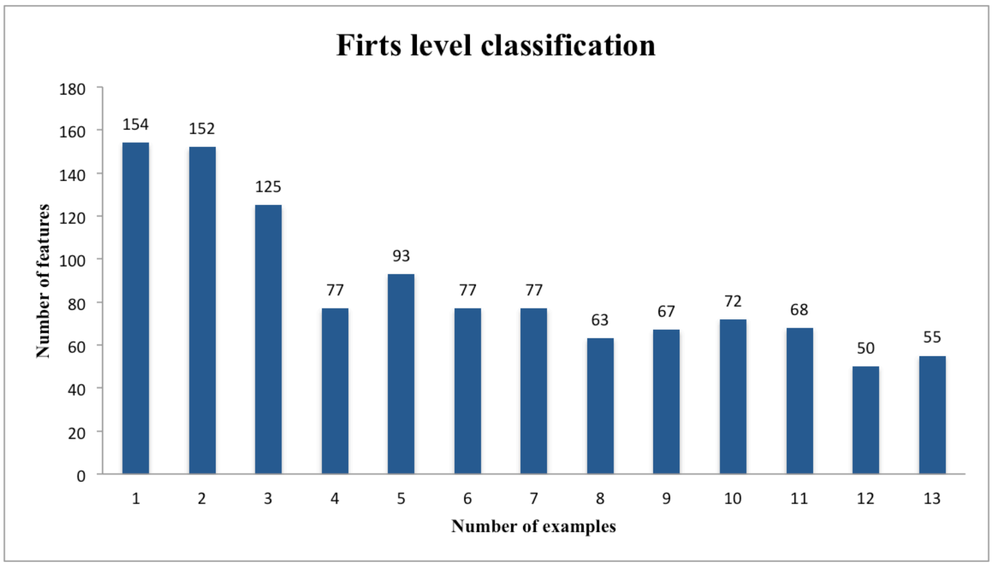


Figure 4.5: First level classification

4.1.7 Case study five (Physical sciences domain)

Considering physical sciences domain which has a number of 1200 examples and taking 100 examples on each iteration, the result shows an obvious decrease in the number of features (Figure 4.6.)

The Figure 4.6 shows that, the first 100 examples have 87 new features then the second 100 have 65 new features, and the third 100 have 45 new features, and so on.

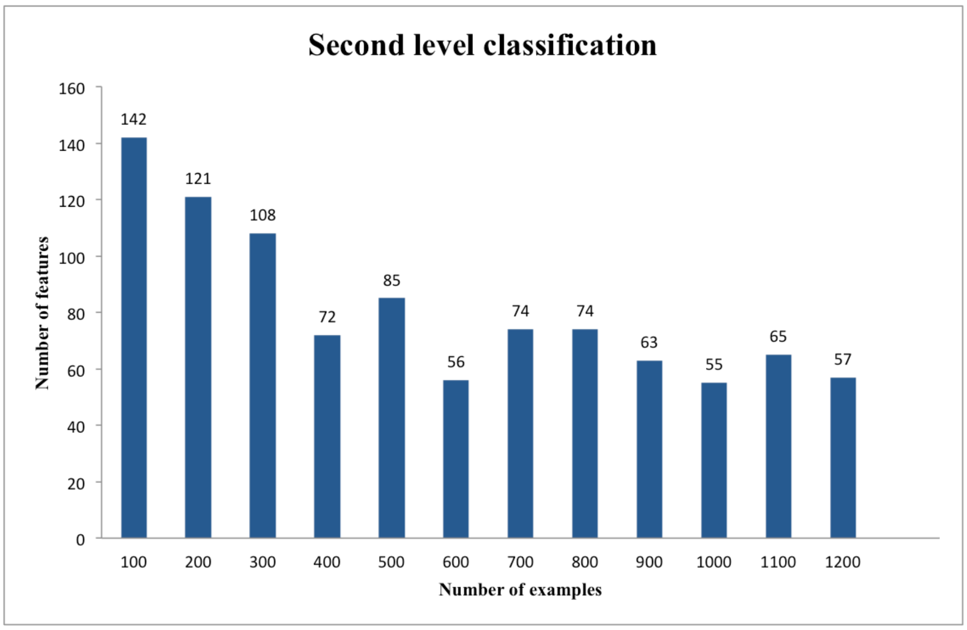


Figure 4.8: Second level classification

4.1.8 Case study six (Analytical chemistry domain)

Considering analytical chemistry domain which has a number of 250 examples and taking 50 examples on each iteration, the results show an obvious decrease in the number of features (Figure 4.7)

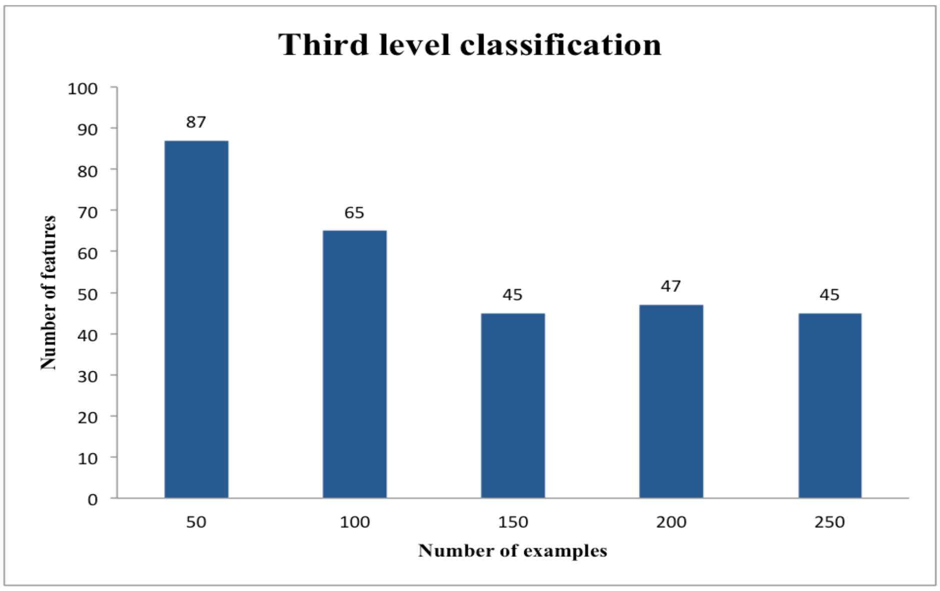


Figure 4.7: Third level classification

4.1.9 Case study seven (Routers- 21578)

The results of the previous case studies showed that a book collection could be trained and the error rate can be reduced if more examples are added. The error rate in this case was calculated using the hamming loss [61]. Figure 4.8 shows the result of the experiment. As it can be observed, the error rate is very low starting at ≈ 1.5% to less than 0.5%.

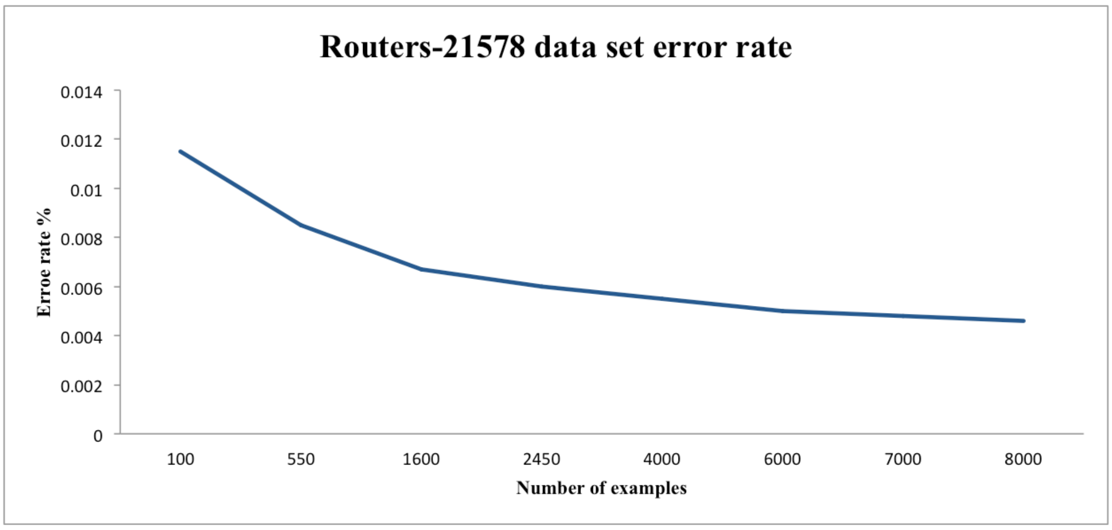


Figure 4.8: Routers-21578 data set error rate

4.1.10 Case study eight (20 newsgroup)

In this experiment, we used another popular algorithm in documents classification called Term Frequency Inverse Document Frequency (TF-IDF) [8] and the data set "20 newsgroup" was used. As explained in section 3.4.8 the data set has about 19,000 documents. It is expressed in terms of the document-term matrix. Rows are represented by the document examples, and columns represent words. A matrix entry (i,j) represents the frequency of occurrence of a word j in a document i. Word frequencies for about 60,000 words are specified for each document. The item (1,1) means document #1, word #1, and the #4 means word#1 has frequency = 4 in document #1 and so on.

In an attempt to reduce the dimensionality of dataset, the following steps were preformed:

- Removing features that do not help in discriminating between class i.e., words like 'a', 'the' that appear in all documents.
- Using Principle Component Analysis PCA [39] for dimensionality reduction

Words with high Inverse Document Frequency (IDF) counts are removed, where IDF represents the ratio of the number of documents in which a particular word appears, to the total number of documents. A high value indicates that the word is present in most of the documents across classes, and hence does not help much in discriminating between the classes. But it was noticed that we were left with a large number of words even after removing the ones with counts above a certain threshold. Since discarding information can affect classifier performance later and setting too low threshold is not a good thing, new alternatives were searched.

4.1.11 Case study nine (The distribution of the error rate)

Figure 4.9 shows the error distributions over the class hierarchy. It shows that as we go down in the hierarchy the error increases. It was taken at error rate = 20 and as it can be observed, the error rate is 20% * 9.1 = 1.82% in the parent level, 20% * 27.3% = 5.46% in the child level, and 20% * 63.6% = 12.72% in the grandchild level or level 3.

Further investigations were conducted in order to determine if there is a constant relation between the class hierarchies when the error rate is 20%. A discrete distribution of the error over the hierarchy was determined. However, we want to find a continuous function of the distribution of the error over the hierarchy at any point where error rate start at maximum to minim.

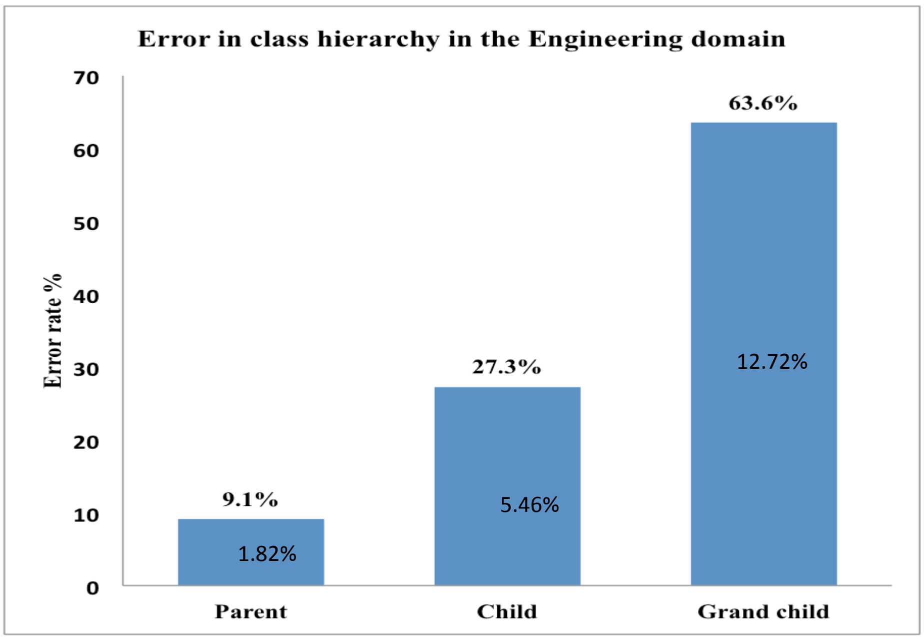


Figure 4.9: Error in class hierarchy in the Engineering domain.

Figure 4.10, shows the distribution of the error rate over class hierarchy. The results show that, the parent classes most of the time has less error rate and the grand children class has more error rate. However, this is not true for all values as rarely the parent class has more errors.

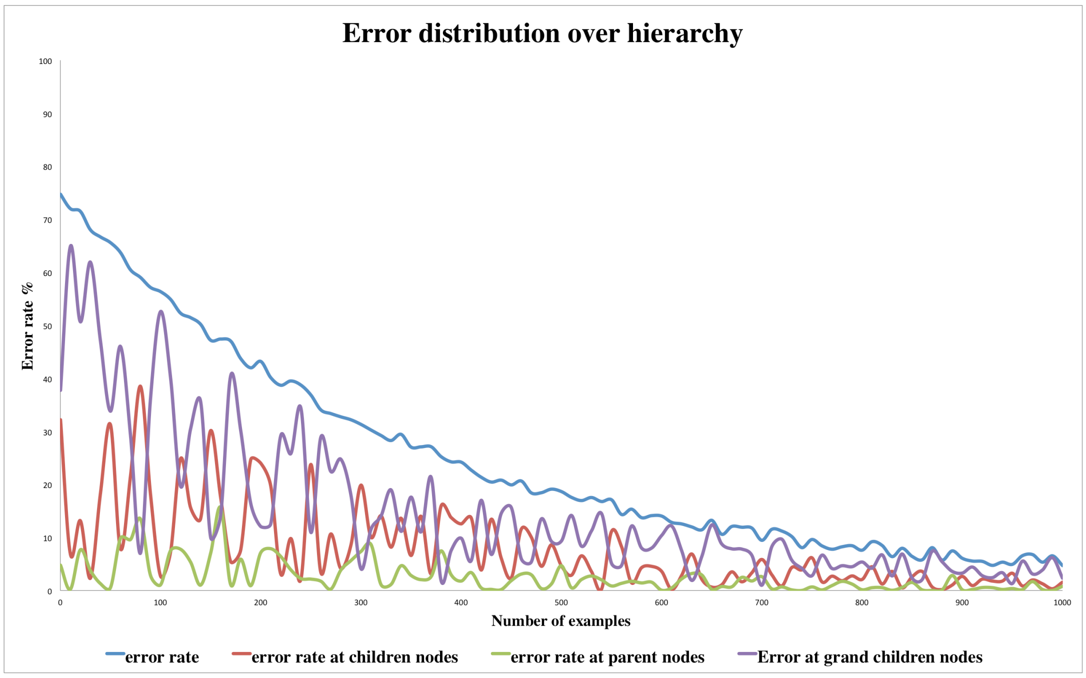


Figure 4.10: Error distributions over hierarchy

4.2. Existing algorithm (Method A: Naïve Bayes algorithm)

4.2.1 Training stage

The first task in the training stage is to separate 10% data for testing purposes from each class. 10% data is separated for each class out of total data for that class. For example, if 100 samples are available for class 1, 10 samples were taken out for testing. This 10% amount is standard in literature and in normal circumstances 10-15% data is taken out for testing. If 50% of data is taken out, too less remains for the training stage and the classifier may not generalize well. After segregating, there were 9012 items in training data and 969 items in testing data, including roughly 10% from each class.

On the second stage, the methodology of data cleaning presented in section 3.1is applied, all the unnecessary words being removed from the titles of the books. These include articles such as, (a, and), prepositions (of, for etc.) and other common meaningless words like volume, edition, e-book etc.

On the third stage, all the remaining words were extracted from the books' titles and each word was assigned a unique number and another identifier to show which class it belongs to. Actually wordID is unnecessary and we may ignore it. For each word, we have a list of classes it can belong to, e.g. chemical may belong to class 1, 3 and 5 so its class ID will be (1,3,5). So the format was like:
word class ID wordID

For instance, the word *Horticulture*, the word ID may be 5 and if it belongs to class 10, its class ID is 10. Then all words are converted to uppercase so that while comparing the words later we do not have to deal with case-sensitivity issue.

Finally, all the words are sorted alphabetically so that when comparing in the testing stage, we do not have to compare with all words. Thus, there is no need to compute distances with all the words but only with those which start with the specified letter. For instance, if the word is Horticulture, then we only need to compare it with the words starting with the letter H.

Once all is done, all the words, sorted alphabetically and in the format mentioned above, are stored in a data file.

4.2.2 Testing stage

In the testing stage, all the titles, along with the class to which it belongs to are passed through the testing function. For instance:

Class	**title**
Agriculture	**Horticultural Reviews, Volume 1**

In the testing stage, the book title goes through the same steps as the training data start the data cleaning process again. Useless words are removed and the useful words are extracted and separated, then converted into uppercase.

After pre-processing, the final shape of the title will be: (HORTICULTURAL REVIEWS and VOLUME and 1) all of those words being removed as they are very generic words and cannot be associated with a particular class. The only word kept is HORTICULTURAL

Next, the training data is loaded and as discussed before, only the words starting with the same letter as the testing word are selected for comparison. For instance, when we want to see which class the word FUNGI belong to, we will compare only with the words starting with the letter F. We do the same with the example above HORTICULTURAL which starts with H, we will compare it only with the words starting with the letter H. We call this a stemming process, where we associate each word with its own stem words only.

The testing word is compared with the above selected words and its distance is computed from them. Two types of matching techniques are used; one is the Lavenstein Distance [77], the number of edits required to convert one string to another. For instance, if HORTICULTURE and HORTICULTURAL are compared, then Lavenstein Distance will be **2**, since 2 edits are required to convert the first string into the second one.

The second one is the number of mismatched characters with respect to the shorter of the two strings e.g. if HORTICULTURE and HORTICULTURAL are compared, the shorter string is HORTICULTURE and with respect to its length, the number of mismatched characters are only 1 (A instead of E), so the distance between the two strings is 1.

Even though both techniques use slightly different and give different results, when we take the minimum in the following stage, the final result is invariably the same.

Once distances from all the words are computed, we select only those with the minimum distance. If multiple words give the same minimum distance (as they do because of repetition), then all of them are selected and the IDs of the classes they belong to, are stored in a cell array. For instance, if we give the label 'Horticultural Reviews, Volume 1' we know that after preprocessing, we are left with only HORTICULTURAL REVIEWS.

Now, when classifier compares HORTICULTURAL with the training words and then computes the distance from them and chooses those words with minimum distances. Since in this case, HORTICULTURAL is present as it is in our data set, the minimum distance will be 0. Now HORTICULTURAL has 35 occurrences in class 1 – (AGRICULTURE) and 1 in class 55 – (Plant Science), it would return an array of 36 elements, with 35 elements as 1 and 1 element as 55 i.e. array = [1 1 1 .. 1 1 55]

The process is repeated for all the words in the label and in the end resulting a cell array of class IDs to which these words may belong. So same process will be repeated for reviews and the class IDs it returns will be appended in the previous array.

In the final step, the most frequent class ID are picked as the class to which the label should belong. If multiple class IDs have the same frequency, then the one which comes first is picked. For instance, in the above case, decision is easy. Since there are 35 instances of class ID 1 and only 1 of class ID 55, mode function will return 1 and we classify the label to class 1 – AGRICULTURE. But suppose if there are 6 instances of class 1 and 15 instances of class 55 and 15 instances of class 20; now there are 2 class IDs with most frequent entries – 20 and 55, mode will return the class ID which is smaller so in that case class 20 will be the answer and label will be classified to class 20.

4.2.3 Results

4.2.3.1 Single label classification

The testing is done for all the classes and the error rate for each class is compared. The error rate for most of the classes is quite high, the reason being that out of the 49000 total words in the 9012 training titles, only about 8500 are unique and the rest are just repetitions. This indicates a huge overlap of data among different classes and as a consequence, the classifier gets confused while testing and therefore misclassify the data.

The last step, in which we pick the ID of the most frequent class, normally has number of IDs with the same frequency, and just picking the first one also introduces errors. The error rate for all the different classes is reported in Table 4.2.

Table 4.2: The error rate for all the different classes using single label classification

Class name	Total number of examples	Correct	Incorrect	Error rate
'Agriculture'	10	4	6	60
'Allergy & Respiratory Medicine'	2	0	2	100
'Analytical Chemistry'	25	6	19	76
'Anatomy & Physiology'	1	0	1	100
'Ancient History & Classical Studies'	10	0	10	100
'Anthropology & Archaeology'	10	1	9	90
'Aquaculture & Fisheries'	10	4	6	60
'Architecture & Planning'	3	1	2	66
'Biochemistry'	12	1	11	91
'Business, Economics, Finance, Accounting'	28	12	16	57
'Cardiology & Cardiovascular Medicine'	14	6	8	57
'Cell & Molecular Biology'	21	4	17	80
'Chemical Engineering '	24	10	14	58
'Civil & Construction Engineering'	28	11	17	60
'Clinical Microbiology'	1	0	1	100
'Clinical Psychology'	20	7	13	65
'Communication Technology & Networks'	25	8	17	68

'Computer Science & Information Technology'	27	4	23	85
'Dentistry'	1	0	1	100
'Dermatology'	2	0	2	100
'Earth & Environmental Sciences'	17	2	15	88
'Ecology'	10	3	7	70
'Electrical & Electronics Engineering'	76	55	21	27
'Endocrinology & Diabetes'	5	2	3	60
'Energy'	13	1	12	92
'Environmental Chemistry'	12	0	12	100
'Food Science & Technology'	23	13	10	43
'Gastroenterology & Hepatology'	6	1	5	83
'General & Physical Chemistry'	20	0	20	100
'Genetics & Evolution'	6	0	6	100
'Geography'	12	3	9	75
'Hematology'	5	0	5	100
'History'	12	2	10	83
'Industrial Chemistry'	23	1	22	95
'Industrial Engineering'	13	0	13	100
'Inorganic Chemistry'	6	0	6	100
'Language & Linguistics'	8	0	8	100
'Literature'	21	15	6	28

'Materials Science'	39	24	15	38
'Mathematics'	16	2	14	87
'Mechanical Engineering'	22	2	20	90
'Microbiology, Virology & Immunology'	7	0	7	100
'Mobile & Wireless Communications'	29	18	11	37
'Nanotechnology'	10	1	9	90
'Neurology'	3	0	3	100
'Neuroscience'	2	0	2	100
'Nursing'	12	10	2	16
'Obstetrics & Gynecology'	4	1	3	75
'Oncology & Radiotherapy'	2	0	2	100
'Organic Chemistry & Catalysis'	35	26	9	25
'Pharmaceutical & Medicinal Chemistry'	20	6	14	70
'Pharmacology'	4	0	4	100
'Philosophy'	26	9	17	65
'Physics'	32	10	22	68
'Plant Science'	9	8	1	11
'Polymer Science & Technology'	12	5	7	58
'Psychiatry'	10	3	7	70
'Psychology'	26	10	16	61
'Public Administration & Management'	18	4	14	77

'Public Health/General'	1	0	1	100
'Religion & Theology'	11	2	9	81
'Sociology, Media, Communications, & Cultural Studies'	19	2	17	89
'Statistics'	26	17	9	34
'Surgery'	1	0	1	100
'Veterinary Medicine'	11	7	4	36

The average error rate for all the classes can be computed by

$$\text{Avg. Error Rate} = \frac{\sum \text{(Error Rate of class) x (No. Words in the class)}}{\text{Total No. of Words}}$$

where ∑ indicates the sum over all the classes. This is just the concept of weighted average. e.g. if we have 3 classes; class 1 has an error rate of 60%, class 2 has an error rate of 40% and class 3 has an error rate of 50%.

Then using the simple average formula, the error rate obtained is 50%. ((60+40+50)/3 = 50). But now suppose there are total of 10 words; 5 belong to class 1, 3 to class 2 and 2 to class 3. Since more words belong to class, logically its error rate should have more contribution in the overall error rate. So we do weighted average, weight for class 1 is 5/10 = 0.5 (no. of words in the class / total words). Similarly weight for class 2 is 3/10 = 0.3 and weight for class 3 is 2/10 = 0.2. Now we multiply with respective error rates and sum them up; so the error rate becomes (0.5 x 60) + (0.3 x 40) + (0.2 x 50) = 30 + 12 +10 = 52%. This error rate is more indicative of the overall behavior of all the classes as it gives more weight to the classes with more amount of data. Using the above formula, the Avg. Error Rate for all classes comes out to be 64%.

As evident from Table 4.2, the error rate is low only for those classes which have a sufficient amount of data. Classes with high amount of data are more likely to be classified correctly just because the probability of occurrence is high. Probability statistics has low error rate because it contains more unique words and the overlapping of data with other classes is low.

4.2.3.2 Multi-label classification

The error rate can be reduced by number of different techniques. One simple way is to get all the different class IDs in the last step of testing stage, each of which have the same probability to be assigned to the given title – this is the Multi-Label Classification and in this case each title can belong to multiple classes. Given that the actual class is among the final set, this step can eliminate all the non-probable classes and another classifier can be used in the next step to choose the final class or a human can do that provided the number of such instances are small.

Alternatively, unique words can be extracted and for each word, the class to which it belongs most frequently can be identified (the bag-of-words approach). Then, instead of assigning the class IDs of all the classes to which the word may belong, we assign only those class (or classes) IDs to which it belongs the most. But this technique is biased towards the class having more training samples, and the error rates for the classes which have the lesser data may increase more. The overall error rate will surely decrease as the classifier is now more biased towards the classes which are more frequent and more likely to come. The results of this technique are slightly better (Table 4.3).

Table 4.3: The error rate for all the different classes using multi-label classification

Class name	**Total number of examples**	**Correct**	**Incorrect**	**Error rate**
'Agriculture'	10	4	6	60
'Allergy & Respiratory Medicine'	2	2	0	0
'Analytical Chemistry'	25	14	11	44
'Anatomy & Physiology'	1	0	1	100
'Ancient History & Classical Studies'	10	7	3	30
'Anthropology & Archaeology'	10	5	5	50
'Aquaculture & Fisheries'	10	7	3	30
'Architecture & Planning'	3	1	2	66
'Biochemistry'	12	1	11	91
'Business, Economics, Finance, Accounting'	28	22	6	21
'Cardiology & Cardiovascular Medicine'	14	11	3	21
'Cell & Molecular Biology'	21	7	14	66
'Chemical Engineering '	24	12	12	50
'Civil & Construction Engineering'	28	14	14	50
'Clinical Microbiology'	1	0	1	100
'Clinical Psychology'	20	14	6	30

'Communication Technology & Networks'	25	11	14	56
'Computer Science & Information Technology'	27	10	17	62
'Dentistry'	1	0	1	100
'Dermatology'	2	0	2	100
'Earth & Environmental Sciences'	17	4	13	76
'Ecology'	10	5	5	50
'Electrical & Electronics Engineering'	76	54	22	28
'Endocrinology & Diabetes'	5	1	4	80
'Energy'	13	1	12	92
'Environmental Chemistry'	12	1	11	91
'Food Science & Technology'	23	13	10	43
'Gastroenterology & Hepatology'	6	2	4	66
'General & Physical Chemistry'	20	2	18	90
'Genetics & Evolution'	6	0	6	100
'Geography'	12	4	8	66
'Hematology'	5	0	5	100
'History'	12	5	7	58
'Industrial Chemistry'	23	2	21	91
'Industrial Engineering'	13	2	11	84
'Inorganic Chemistry'	6	1	5	83
'Language & Linguistics'	8	1	7	87

'Literature'	21	14	7	33
'Materials Science'	39	25	14	35
'Mathematics'	16	5	11	68
'Mechanical Engineering'	22	4	18	81
'Microbiology, Virology & Immunology'	7	0	7	100
'Mobile & Wireless Communications'	29	17	12	41
'Nanotechnology'	10	0	10	100
'Neurology'	3	1	2	66
'Neuroscience'	2	0	2	100
'Nursing'	12	5	7	58
'Obstetrics & Gynecology'	4	0	4	100
'Oncology & Radiotherapy'	2	0	2	100
'Organic Chemistry & Catalysis'	35	23	12	34
'Pharmaceutical & Medicinal Chemistry'	20	8	12	60
'Pharmacology'	4	1	3	75
'Philosophy'	26	9	17	65
'Physics'	32	8	24	75
'Plant Science'	9	5	4	44
'Polymer Science & Technology'	12	2	10	83
'Psychiatry'	10	2	8	80
'Psychology'	26	8	18	69

'Public Administration & Management'	18	8	10	55
'Public Health/General'	1	0	1	100
'Religion & Theology'	11	5	6	54
'Sociology, Media, Communications, & Cultural Studies'	19	2	17	89
'Statistics'	26	14	12	46
'Surgery'	1	0	1	100
'Veterinary Medicine'	11	8	3	27

The Avg. Error Rate for this technique comes out to be 57% which is an improvement over the previous technique.

4.3 Existing algorithm (Method B: KNN algorithm)

Suppose the classifier was asked to classify some sample X, and after computation classifier finds that it can belong to any one of the class 1, class 2 and class 3. In this case, we need a rule to break a tie and the one we used in previous section was to pick the lowest class ID. Therefore, the sample will be classified to class 1 even though it may belong to class 2 or class 3. Suppose that sample originally belonged to class 2; then the classification will be wrong and will account as an error. But if we don’t use any tie-breaker and outputs all the equally probable classes, i.e. class 1, class 2 and class 3, then there will be no error as sample does belong to one of these classes. This is the whole idea of multi-label classification or multi-output classification, in which input X is not mapped to a single scalar class *y*, but rather a vector of classes Y

The algorithm was modified to accommodate the multi-label classification and an error occurs only if the actual class ***y*** was not among the vector of classes Y given by the classifier. The error rate is reduced in this case compared to the case of single-output classification. The results are shown in Table 4.4.

Table 4.4: Results obtained with KNN algorithm

Class name	**Total number of examples**	**Correct**	**Incorrect**	**Error rate**
'Agriculture'	10	4	6	60
'Allergy & Respiratory Medicine'	2	2	0	0
'Analytical Chemistry'	25	14	11	44
'Anatomy & Physiology'	1	0	1	100
'Ancient History & Classical Studies'	10	7	3	30
'Anthropology & Archaeology'	10	5	5	50
'Aquaculture & Fisheries'	10	7	3	30
'Architecture & Planning'	3	1	2	66
'Biochemistry'	12	2	10	83
'Business, Economics, Finance, Accounting'	28	22	6	21
'Cardiology & Cardiovascular Medicine'	14	11	3	21
'Cell & Molecular Biology'	21	8	13	61
'Chemical Engineering '	24	14	10	41
'Civil & Construction Engineering'	28	20	8	28

'Clinical Microbiology'	1	0	1	100
'Clinical Psychology'	20	14	6	30
'Communication Technology & Networks'	25	13	12	48
'Computer Science & Information Technology'	27	12	15	55
'Dentistry'	1	0	1	100
'Dermatology'	2	1	1	50
'Earth & Environmental Sciences'	17	6	11	64
'Ecology'	10	6	4	40
'Electrical & Electronics Engineering'	76	61	15	19
'Endocrinology & Diabetes'	5	3	2	40
'Energy'	13	7	6	46
'Environmental Chemistry'	12	2	10	83
'Food Science & Technology'	23	15	8	34
'Gastroenterology & Hepatology'	6	4	2	33
'General & Physical Chemistry'	20	8	12	60
'Genetics & Evolution'	6	2	4	66
'Geography'	12	9	3	25
'Hematology'	5	3	2	40
'History'	12	6	6	50
'Industrial Chemistry'	23	4	19	82

'Industrial Engineering'	13	4	9	69
'Inorganic Chemistry'	6	2	4	66
'Language & Linguistics'	8	4	4	50
'Literature'	21	16	5	23
'Materials Science'	39	28	11	28
'Mathematics'	16	8	8	50
'Mechanical Engineering'	22	8	14	63
'Microbiology, Virology & Immunology'	7	2	5	71
'Mobile & Wireless Communications'	29	19	10	34
'Nanotechnology'	10	2	8	80
'Neurology'	3	2	1	33
'Neuroscience'	2	0	2	100
'Nursing'	12	11	1	8
'Obstetrics & Gynecology'	4	1	3	75
'Oncology & Radiotherapy'	2	1	1	50
'Organic Chemistry & Catalysis'	35	29	6	17
'Pharmaceutical & Medicinal Chemistry'	20	11	9	45
'Pharmacology'	4	2	2	50
'Philosophy'	26	20	6	23
'Physics'	32	15	17	53
'Plant Science'	9	8	1	11

'Polymer Science & Technology'	12	8	4	33
'Psychiatry'	10	5	5	50
'Psychology'	26	17	9	34
'Public Administration & Management'	18	10	8	44
'Public Health/General'	1	0	1	100
'Religion & Theology'	11	7	4	36
'Sociology, Media, Communications, & Cultural Studies'	19	7	12	63
'Statistics'	26	20	6	23
'Surgery'	1	0	1	100
'Veterinary Medicine'	11	11	0	0

Green blocks indicate the classes with error rate of less than 50% and it is visible that now such instances are lot more than in previous tables. The overall Avg. Error Rate is **41%** which is an improvement of nearly 16% from the previous method.

4.4 Classification of parent class

The classification of Parent Classes is much less error prone as there are few parent classes and the margin for error is smaller. In this case, there are 6 Parent classes corresponding to 65 Child classes. The original classifier (in multi-label case) returns a set of child classes, which may or may not belong to the same parent class. So we get the parent class for each of these child classes and then compare them one by one to the original parent class. An error occurs only if none of the parent classes match the original parent class. The error rates for all 6 parent classes are shown in Table 4.5.

Table 4.5: Errors obtained in case of parent classification

Class name	**Total number of examples**	**Correct**	**Incorrect**	**Error rate**
'Chemistry'	153	94	59	38
'Engineering'	244	212	32	13
'Health Sciences'	132	114	18	13
'Life, Earth & Environmental Sciences '	103	65	38	36
'Physical Sciences'	171	125	46	26
'Social Sciences & Humanities'	166	147	19	11

The highlighted entries indicate the classes with error rate of less than 30%. The overall weighted Average Error Rate is **21%**.

Discussion

Large collections of a library of documents may include hundreds of thousands of documents and more are added every week. In this case, there is a need for finding out if a machine learning algorithms might be helpful. So, we can answer this question by the following this simple idea: a small percentage of all examples are enough for the induction of a high performance classifier. If this is the case, the use of machine learning is helpful. However, if 50% of the collection is not enough, machine learning is not an adequate approach as the resource consumed (human and computational time) are two high and the advantages it brings are insignificant compared with the drawbacks.

In order to test this idea, a new algorithm is developed and implemented. When applying the proposed algorithm for the case study one, if a perfect training is performed, then the error rate is very small. Also, the relation between the number of iteration and the error rate was shown, if more samples are added, then error is decreasing.

In the case study two, where a collection of about 9000 book titles is available, when using 100% of the collection, an error rate of 40% $\pm$ 5% is obtained. The result of the experiment is very pessimistic. However, if a closer look and careful attention is paid to Figure 4.2, it can be observed that the curve trend is going slowly toward a lower error rate. It can be assumed, that a collection of 50,000 book titles is available, then the error rate will be 10% $\pm$ 2% from the number of training example (close to 20,000 book titles).

This problem is due to: i) the large number of features and low number of examples; and ii) the high frequency of each feature in relation to documents, which makes the classifier confused about the examples. For example, for the case study three, when analyzing the results from Figure 4.3, it can be observed that error rate drops to less than 10% with number of examples less than 1000. This is due to the small number of features, as all the examples belong to the engineering domain and thus; the features are very limited and related only to this domain.

Figure 4.6 shows the social sciences and humanities domain (case study four). This experiment gives a result very similar to the result found in the engineering domain (experiment three). Therefore, it can be concluded that if each domain is classified separately, a lesser number of examples can be obtained than in the case of classifying the entire collection. So, according to experiment 2, 3, and 4, we can see that, if the domain is small enough such as the engineering domain or the social sciences and humanities domain, then the error rate drops to 10% at much faster rate. Figure 5.1 summarizes the finding for these case studies.

Figure 5.1 shows a disparity in the error rate between different domains in the number of examples needed to take the error rate down to 10%. So, in case of the whole domain 4000 examples are required, while the engineering or social sciences and humanities domain require 550 examples and 650 examples respectively.

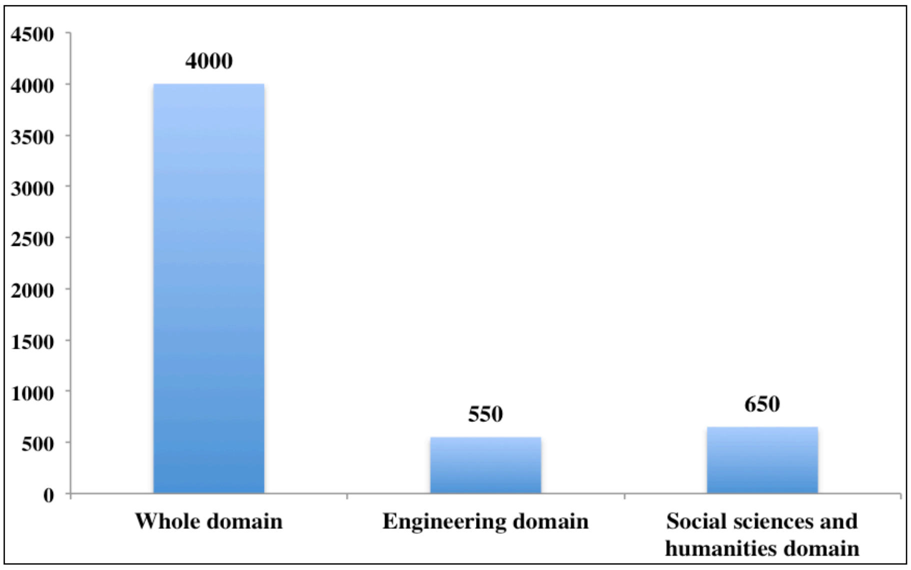


Figure 5.1: Findings of experiments 2, 3, and 4.

The second research question presented in chapter 1 is, if a classifier was induced at higher level, does this implies to a lower class classifier? Which means, if an example was classified at higher level class, would this example be a parent and grandparent to all its documents? In order to test this hypothesis different case studies were tested.

For example, in the case study three, Figure 4.4 shows that when we investigated the error at 20% we found that 9.1% of the error is in the parent level, which means the error at parent level is 9.1 * 20% = 1.82%, then it is 5.46% at the children level and 12.72% at the grand children level. This experiment shows that there are more errors in the lower level than in the higher level of the class hierarchy.

Figure 4.5 shows that if the examples are correctly classified at the parent level and it has x% error in the lower level, this means that there are x % misclassified examples in the lower level. However, the error of the examples that where classified in the upper level to the classes that are correctly classified in the lower level is very low and it is found to be << x%. This concludes that, if we have 100 examples correctly classified in the upper level, those examples will have x% misclassified examples in the lower level. And if we have 100 examples, correctly classified in the lower level they will have << x% misclassified examples in the upper level. Also, it was shown that parents and grandparents classes might have miss-classified children in the lower level more than misclassified grandparent.

Another data set used is Routers data set (case study eight.) This document classification was done using Naïve Bayes technique and the error rate drops to less than 1 % at a very early stage and with a number of examples = 100 documents the error rate drops to a number close to 200 at 8000 examples. This behavior is due to large size of information as we used to full text in this case, and we use the documents' contents and not only the documents' titles. This concludes that if more information is used in the document, the error rate will drop dramatically. This result may address the first research question as follow: If we can use more information about the document, will it give a much better result? In order to answer this question a set of 3 more case studies (five, six and seven) were used for simulations.

The results of those experiments showed that when the training is started, the number of features is high and the relation of number of examples to the number of features is almost 1:1.5. However, more examples are added, the number of new features drops to point where it gets to 1 to 0.5. This means we can have a much better error rate if we add more examples. The number of new features is decreasing in all levels, parent, grandparents and children at almost the same rate as presented in Figures 4.7, 4.8, and 4.9. This means if we add more example we will have a great improvement in the error rate at all levels.

In summary, a new algorithm using machine learning was proposed and compared to two of the very popular methods Naïve Bayes and KNN methods. As shown in Chapter 3, in order to perform the comparisons between algorithms, two well-known and highly recognized evaluation metrics (macro-micro averaging and hamming loss) were used.

Simulations showed that using only book titles to classify a large collection of library contents is very time consuming and costly too. Alternatively, it was suggested that either use the sub domain approach and classify each sub domain separately or include as much information as possible such as; abstract of documents, an introduction, or the whole document.

Another major finding is that a parent node can be a parent of all documents with a small and acceptable error rate e_1. However, a child node is a child if all ancestors with a very small error rate e_2 where $e_2 \gg e_1$.

The problems encountered when conducted the research presented in this work is that more data sets must be used to verify the findings. Although, five different data sets from different domains were used, the author still feels that more experiments on different domains may be required, this aspect being a possible weakness of the research. Another weakness is related to the fact that only one machine learning method for each data set was used and the proposed algorithm was compared only with KNN and Naïve Bayes.

Different results could have been obtained if different methods were used, but the limited time allocated for this research made it impossible, as a big percent of it was spent on formulating the problem and finding the suitable data set. Selecting totally different data sets and approaching the problem from different perspective gives novelty to this research. A set of aspects were difficult to handle as many problems were encountered (such as having a data set that may need a huge memory to solve which is turn ends up in to “out of memory” error).

The significance of this research is very clear as it addresses the first research question clearly and informs that a library collection or a similar collection can be classified automatically using machine learning algorithms.

Conclusion

This study is focused on the HMC with emphasis on several case studies to draw the research observations. This is done by conducting various experiments using a few popular machine learning algorithms: KNN and Naïve Bayes algorithms, along with the proposed algorithm based on SVM. The research also aimed to identify the child-parent relation, and parent-child relation. To this goal, a proprietary software was built to test whether an example is classified into its corresponding child and grandchild as well as if the grandchild belonged to its accurate parent and grandparent. The significance of the research is the motivation for the use of machine learning in digital libraries which were the primary resource that were used in the study. We have also used 20 newsgroup and Routers data set to compare the performance.

The performance analysis was done using Macro and Micro averaging and hamming loss metrics. Based on the results, it was found that, it is very time consuming and costly to use only book titles to classify a large collection of library contents. Thus, it was suggest that either use the sub domain approach or classify each sub domain separately, or include more information such as abstract of documents, an introduction of the document. Another major finding is that a parent node can be a parent of all documents with a small and acceptable error rate. In general, these findings are very similar to the many recent published studies.

In future, we plan to generalize the proposed algorithm for the hierarchical case where the interrelation of the class labels can be specified by a generalization tree of a directed acyclic graph (DAG) as in Vateekul et al. [2] study.

References

1. E. Alpaydin, *Introduction to machine learning*: The MIT Press; second edition (December 4, 2009).

2. P. Vateekul, M. Kubat, and K. Sarinnapakorn, "Hierarchical Multi-Label Classification with SVMs: a Case Study in Gene Function Prediction." . unpublished work.

3. V. Vapnik, *The nature of statistical learning theory*: Springer; 2nd edition (December 1, 1999)

4. T. Joachims, *Text categorization with support vector machines: Learning with many relevant features*: Springer, 1998.

5. L. Wang, *Support Vector Machines: theory and applications,* 177: Springer, 2005.

6. L. Baoli, L. Qin, and Y. Shiwen, "An adaptive k-nearest neighbor text categorization strategy," *ACM Transactions on Asian Language Information Processing (TALIP),* vol. 3, no. 4. pp.215-226, 2004.

7. N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning,* vol. 29, no. 2-3. pp.131-163, 1997.

8. A. Clare and R. D. King, "Predicting gene function in Saccharomyces cerevisiae," *Bioinformatics,* vol. 19, no. suppl 2. pp.ii42-ii49, 2003.

9. A. McCallum and K. Nigam, "A comparison of event models for naive bayes text classification." *AAAI-98 workshop on learning for text categorization* vol. 752, pp. 41-48. 1998. Citeseer.

10. J. T.-Y. Kwok, "Automated text categorization using support vector machine." *In Proceedings of the International Conference on Neural Information Processing (ICONIP)* . 1998. Citeseer.

11. C. Chow and C. Liu, "Approximating discrete probability distributions with dependence trees," *Information Theory, IEEE Transactions on,* vol. 14, no. 3. pp.462-467, 1968.

12. L. Schietgat, C. Vens, J. Struyf, H. Blockeel, D. Kocev, and S. Dzeroski, "Predicting gene function using hierarchical multi-label decision tree ensembles." *Bmc Bioinformatics*, vol. 11, 2010.

13. L. S. Larkey and W. B. Croft, "Combining classifiers in text categorization." *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval* , pp. 289-297. 1996. ACM.

14. D. Koller and M. Sahami, "Hierarchically classifying documents using very few words," *Proceedings of the 14th International Conference on Machine Learning ML, Nashville, page 170--178.* (*1997*).

15. D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers." *Proceedings of the 17th annual international ACM SIGIR*

conference on Research and development in information retrieval , pp. 3-12. 1994. Springer-Verlag New York, Inc.

16. T. M. Mitchell, "Machine learning," *Burr Ridge, IL: McGraw Hill,* vol. 45, 1997.

17. W. T. Aung and K. H. M. S. Hla, "Random forest classifier for multi-category classification of web pages." *Services Computing Conference, 2009.APSCC 2009.IEEE Asia-Pacific* , pp. 372-376. 2009. IEEE.

18. L. Wang, M. Q. Yang, and J. Y. Yang, "Prediction of DNA-binding residues from protein sequence information using random forests," *Bmc Genomics,* vol. 10, no. Suppl 1. pp.S1, 2009.

19. S. Ahmad, M. M. Gromiha, and A. Sarai, "Analysis and prediction of DNA-binding proteins and their binding residues based on composition, sequence and structural information," *Bioinformatics,* vol. 20, no. 4. pp.477-486, 2004.

20. C. Yan, M. Terribilini, F. Wu et al., "Predicting DNA-binding sites of proteins from amino acid sequence," *BMC bioinformatics,* vol. 7, no. 1. pp.262, 2006.

21. L. Wang and S. J. Brown, "Prediction of DNA-binding residues from sequence features," *Journal of bioinformatics and computational biology,* vol. 4, no. 06. pp.1141-1158, 2006.

22. L. Wang and S. J. Brown, "BindN: a web-based tool for efficient prediction of DNA and RNA binding sites in amino acid sequences," *Nucleic acids research,* vol. 34, no. suppl 2. pp.W243-W248, 2006.

23. L. Breiman, "Random forests," *Machine learning,* vol. 45, no. 1. pp.5-32, 2001.

24. B. S. Yang, X. Di, and T. Han, "Random forests classifier for machine fault diagnosis," *Journal of mechanical science and technology,* vol. 22, no. 9. pp.1716-1725, 2008.

25. H. E. Osman, "A binary classification and online vision." *Neural Networks, 2009.IJCNN 2009.International Joint Conference on* , pp. 1142-1148. 2009. IEEE.

26. A. Mathur and G. M. Foody, "Multiclass and binary SVM classification: Implications for training and classification users," *Geoscience and Remote Sensing Letters, IEEE,* vol. 5, no. 2. pp.241-245, 2008.

27. G. Liu, X. Zhang, and S. Zhou, "Multi-class Classification of Support Vector Machines Based on Double Binary Tree." *Natural Computation, 2008.ICNC'08.Fourth International Conference on* vol. 2, pp. 102-105. 2008. IEEE.

28. M. Kubat, K. Sarinnapakorn, and S. Dendamrongvit, "Induction in Multi-Label Text Classification Domains." *Advances in Machine Learning II.* pp. 225-244. 2010. Springer.

29. C. N. Silla Jr and A. A. Freitas, "A survey of hierarchical classification across different application domains," *Data Mining and Knowledge Discovery,* vol. 22, no. 1-2. pp.31-72, 2011.

30. L. J. Jensen, R. Gupta, N. Blom et al., "Prediction of human protein function from post-translational modifications and localization features," *Journal of molecular biology,* vol. 319, no. 5. pp.1257-1265, 2002.

31. M. Riley, "Functions of the gene products of Escherichia coli," *Microbiological reviews,* vol. 57, no. 4. pp.862, 1993.

32. W. R. Weinert and H. S. r. Lopes, "Neural networks for protein classification," *Applied Bioinformatics,* vol. 3, no. 1. pp.41-48, 2004.

33. A. Sun and E. P. Lim, "Hierarchical text classification and evaluation." *Data Mining, 2001.ICDM 2001, Proceedings IEEE International Conference on* , pp. 521-528. 2001. IEEE.

34. C. D. Nguyen, T. A. Dung, and T. H. Cao, "Text classification for DAG-structured categories." *Advances in Knowledge Discovery and Data Mining*. pp. 290-300. 2005. Springer.

35. A. Secker, M. N. Davies, A. A. Freitas et al., "An experimental comparison of classification algorithms for hierarchical prediction of protein function," *Expert Update (Magazine of the British Computer Society's Specialist Group on AI),* vol. 9, no. 3. pp.17-22, 2007.

36. W. Bi and J. T. Kwok, "Multi-label classification on tree-and dag-structured hierarchies." *Proceedings of the 28th International Conference on Machine Learning (ICML-11)* , pp. 17-24. 2011.

37. N. Alaydie, C. K. Reddy, and F. Fotouhi, "Exploiting label dependency for hierarchical multi-label classification." *Advances in Knowledge Discovery and Data Mining*. pp. 294-305. 2012. Springer.

38. J. R. Quinlan, "Induction of decision trees," *Machine learning,* vol. 1, no. 1. pp.81-106, 1986.

39. "Dimensionality reduction." http:// en.wikipedia.org/ wiki/ Dimensionality_reduction, 2014.

40. H. Blockeel, L. De Raedt, and J. Ramon, "Top-down induction of clustering trees," *arXiv preprint cs/0011032*, 2000.

41. G. Pandey, C. L. Myers, and V. Kumar, "Incorporating functional inter-relationships into protein function prediction algorithms," *BMC bioinformatics,* vol. 10, no. 1. pp.142, 2009.

42. H. Lo, S. Lin, and H. M. Wang, "Generalized k-Labelsets Ensemble for Multi-Label and Cost-Sensitive Classification,", 2013.

43. G. Tsoumakas, I. Katakis, and I. Vlahavas, "Mining multi-label data." *Data mining and knowledge discovery handbook.* pp. 667-685. 2010. Springer.

44. G. Tsoumakas and I. Vlahavas, "Random k-labelsets: An ensemble method for multi label classification." *Machine Learning: ECML 2007.* pp. 406-417. 2007. Springer.

45. "Mulan: A Java Library for Multi-Label Learning." . 2014.

46. H. Qu, S. Zhang, H. Liu et al., "A multi-label classification algorithm based on label-specific features," *Wuhan University Journal of Natural Sciences,* vol. 16, no. 6. pp.520-524, 2011.

47. X. Kong, B. Cao, and P. S. Yu, "Multi-label classification by mining label and instance correlations from heterogeneous information networks." *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* , pp. 614-622. 2013. ACM.

48. B. Chen, Y. Ding, and D. J. Wild, "Assessing drug target association using semantic linked data," *PLoS computational biology,* vol. 8, no. 7. pp.e1002574, 2012.

49. X. Kong, P. S. Yu, Y. Ding et al., "Meta path-based collective classification in heterogeneous information networks." *Proceedings of the 21st ACM international conference on Information and knowledge management* , pp. 1567-1571. 2012. ACM.

50. X. Kong, X. Shi, and S. Y. Philip, "Multi-Label Collective Classification." *SDM* vol. 11, pp. 618-629. 2011. SIAM.

51. R. Nicolas, A. Sancho-Asensio, E. Golobardes et al., "Multi-label classification based on analog reasoning," *Expert Systems with Applications,* vol. 40, no. 15. pp.5924-5931, 2013.

52. L. Enrique Sucar, C. Bielza, E. F. Morales et al., "Multi-label classification with Bayesian network-based chain classifiers," *Pattern Recognition Letters*, 2013.

53. R. Cerri, R. C. Barros, and A. C. De Carvalho, "Hierarchical multi-label classification using local neural networks," *Journal of Computer and System Sciences,* vol. 80, no. 1. pp.39-56, 2014.

54. "HMC Software and Datasets." *http://dtai.cs.kuleuven.be/clus/hmcdatasets/* . 2008.

55. D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research,* vol. 3. pp.993-1022, 2003.

56. S. Arlot and A. Celisse, "A survey of cross-validation procedures for model selection," *Statistics surveys,* vol. 4. pp.40-79, 2010.

57. A. Santos, A. Canuto, and A. F. Neto, "A comparative analysis of classification methods to multi-label tasks in different application domains," *Int.J.Comput.Inform.Syst.Indust.Manag.Appl,* vol. 3. pp.218-227, 2011.

58. M. Kubat and S. Matwin, "Addressing the curse of imbalanced training sets: one-sided selection." *ICML* vol. 97, pp. 179-186. 1997.

59. A. Singhal, "Modern information retrieval: A brief overview," *IEEE Data Eng.Bull.,* vol. 24, no. 4. pp.35-43, 2001.

60. Y. Yang, "An evaluation of statistical approaches to text categorization," *Information retrieval,* vol. 1, no. 1-2. pp.69-90, 1999.

61. V. Gjorgjioski, D. Kocev, and S. D++eroski, "COMPARISON OF DISTANCES FOR MULTI-LABEL CLASSIFICATION WITH PCTs,"

62. K. Brinker, J. Frnkranz, and E. Hllermeier, "A unified model for multi label classification and ranking." *Proceedings of the 2006 conference on ECAI 2006: 17th European Conference on Artificial Intelligence August 29--September 1, 2006, Riva del Garda, Italy* , pp. 489-493. 2006. IOS Press.

63. M. L. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency." *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining* , pp. 999-1008. 2010. ACM.

64. G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," *International Journal of Data Warehousing and Mining (IJDWM),* vol. 3, no. 3. pp.1-13, 2007.

65. D. Damm, C. Fremerey, V. Thomas et al., "A digital library framework for heterogeneous music collections: from document acquisition to cross-modal interaction," *International Journal on Digital Libraries,* vol. 12, no. 2-3. pp.53-71, 2012.

66. "Internet Archieve." *http://archive.org/index.php* . 2014.

67. "Google Books." *http://books.google.com* . 2014.

68. "Open Library." *http://openlibrary.org* . 2014.

69. "New York Public Library." *http://www.nypl.org* . 2014.

70. "Wiley Online Library." *http://onlinelibrary.wiley.com/* . 2014.

71. D. D. Lewis, "Reuters-21578 text categorization test collection." . 2014.

72. "The 20 Newsgroups data set." *http://qwone.com/~jason/20Newsgroups/* . 2014.

73. W. B. Frakes and R. Baeza-Yates, "Information retrieval: data structures and algorithms," Prentice Hall; 1 edition (June 22, 1992).

74. M. F. Porter, "An algorithm for suffix stripping," *Program: electronic library and information systems,* vol. 14, no. 3. pp.130-137, 1980.

75. K. J. Cios, R. W. Swiniarski, W. Pedrycz et al., "Feature Extraction and Selection Methods." *Data Mining*. pp. 133-233. 2007. Springer.

76. K. K. Dobbin and R. M. Simon, "Optimally splitting cases for training and testing high dimensional classifiers," *BMC medical genomics,* vol. 4, no. 1. pp.31, 2011.

77. V. I. Levenshtein, "Binary codes capable of correcting deletions, insertions and reversals." *Soviet physics doklady* vol. 10, p.707. 1966.

78. T. Fagni and F. Sebastiani, "On the selection of negative examples for hierarchical text categorization." *In Proceedings of the 3rd language*

technology conference, pp. 24–28, 2007.

79. T. Fagni and F. Sebastiani, “Selecting negative examples for hierarchical text classification: An experimental comparison.” *Journal of the American Society for Information Science and Technology*, vol. 61, no.11. pp.256–2265, 2010.

80. C. Vens, J. Struyf, L. Schietgat, S. Dzeroski, and H. Blockeel, “Decision trees for hierarchical multi-label classification.” *Machine Learning*, vol. 73, no. 2. pp.185–214, 2008.

81. “Lancaster University”. *http://www.comp.lancs.ac.uk/computing/research/stemming/ general/porter.htm* . 2014.

التأثر من الأمثلة المتعددة التصنيف

هند هزاع طلال الشريف

المستخلص

ان عملية تصنيف النصوص الكتابية هي باختصار وضع كل مثال (كتاب او وثيقة نصية) في صنف واحد او اكثر. وابسط عملية يتم فيها استخدام الكمبيوتر للتصنيف تحت مجال تعليم الالة (او ان الكمبيوتر يتعلم من الامثلة ويصبح اكثر فعالية مع الزمن) هي ان ننشي مصنف واحد يصنف الى جزئين فقط ثم نصنع من هذا المصنف عدة صور بحيث تلائم كل منها مجال محدد، ثم يتم تشغيل هذه المصنفات في وقت واحد للعمل بالتوازي.

وقد نجد ان تصنيف مكتبة رقمية بمافيها من كميات كبيرة من الكتب والوثائق هو مثال جيد يعطي دافع قوي لمجال البحث. بحيث يتم تصنيف الكتب الى اصناف متعددة ثم اصناف فرعية وهكذا. بحيث يتنج في النهاية تصنيف شجري– هيكلي

في مجال هذا البحث نجد ان هناك نقظة اخرى قيمة للبحث وهي– هل يمكن ان يتم تصنيف وثيقة واحدة الى اكثر من صنف، اذا كانت الاجابة بنعم، فاننا في هذه الحالة سنعمل في بحثنا في مجال ما يسمى تصنيف متعدد الاصناف للوثيقة الواحدة.

ان الدراسة التي نقوم بها هنا هي تبحث في اظهار مدى الفائدة العائدة من استخدام الكمبيوتر في التصنيف الالي بطريقة– تعليم الالة ، وهل هي مفيدة ام لا. فمثلا ان كنا بحاجة الى استخدام اكثر من 50% من المحتوى لتعليم الكمبيوتر، فان ذلك قد يكون مجهود كبير ولا توجد فائدة عائدة الينا. وان كنا بحاجة الى 10% فقط من المحتوى، فان هناك فائدة كبيرة لنا من استخدام الكمبيوتر.

اخيرا، نريد ان نبحث ان كان هناك علاقة بين المصنفات، بحث هل نجد ان الوثيقة التابعة لصنف معين، هي تتبع جميع الاصناف العليا له في الشجرة، ان كانت مصنفة هل هيا تعتبر مرجع لمجيع الاصناف التي تاتي تحتها في الشجرة.

سنقوم بتصميم نموذج لتصنيف الوثائق تلقائيا واختباره ومقارنته مع الانظمة الاخرى الموجودة.

التأثر من الأمثلة المتعددة التصنيف

هند هزاع طلال الشريف

بحث مقدم لنيل درجة الماجستير في العلوم (علوم الحاسبات)

اشراف

د.وديع صالح الحلبي

كلية الحاسبات ونظم المعلومات

جامعة الملك عبدالعزيز

جدة ـ المملكة العربية السعودية

شعبان ١٤٣٥هـ ـ يونيو ٢٠١٤ م (١٤٣٥/٨/٢٧)

بسم الله الرحمن الرحيم

التأثر من الأمثلة المتعددة التصنيف

هند هزاع طلال الشريف

بحث مقدم لنيل درجة الماجستير في العلوم (علوم الحاسبات)

كلية الحاسبات ونظم المعلومات

جامعة الملك عبدالعزيز

جدة ـ المملكة العربية السعودية

شعبان ١٤٣٥هـ ـ يونيو ٢٠١٤ م (١٤٣٥/٨/٢٧)