EXAMINING LEARNING ALGORITHMS FOR TEXT CLASSIFICATION IN DIGITAL LIBRARIES

By

Ismail Fahmi

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENTS FOR THE DEGREE OF
MASTER OF ARTS
AT
UNIVERSITY OF GRONINGEN
GRONINGEN, THE NETHERLAND
JANUARY 2004

UNIVERSITY OF GRONINGEN

DEPARTMENT OF

ALFA-INFORMATICA

The undersigned hereby certify that they have read and recommend to the Faculty of Letteren for acceptance a thesis entitled "**Examining Learning Algorithms for Text Classification In Digital Libraries**" by **Ismail Fahmi** in partial fulfillment of the requirements for the degree of **Master of Arts**.

Dated: <u> January 2004 </u>

Supervisor:

<u>John Nerbonne</u>

Readers:

<u>Henny Klein</u>

<u>—</u>

UNIVERSITY OF GRONINGEN

Date: **January 2004**

Author: **Ismail Fahmi**

Title: **Examining Learning Algorithms for Text Classification In Digital Libraries**

Department: **Alfa-Informatica**

Degree: **M.A.**   Convocation: —   Year: **2004**

To Agnes

Table of Contents

Abstract

Information presentation in a digital library plays important role especially in improving the usability of collections and helping users to get started with the collection. One approach is to provide an overview through large topical category hierarchies associated with the documents of a collection. But with the growth in the amount of information, this manual classification becomes a new problem for users. The navigation through the hierarchy can be a time-consuming and frustrating process.

In this master thesis, we examine the performance of machine learning algorithms for automatic text classification. We examine three learning algorithms namely ID3, Instance-Based Learning, and Naive Bayes to classify documents according to their category hierarchies. We focused on the effectiveness measurement such as recall, precision, the F_1-measure, error, and the learning curve in learning a manually classified metadata collection from the Indonesian Digital Library Network (IndonesiaDLN), and we compare the results with an examination of the Reuters-21578 dataset. We summarize the algorithm that is most suitable for the digital library collection and the performance of the algorithms on these datasets.

Acknowledgements

I would like to thank Prof. dr. ir. John Nerbonne, my supervisor, for his many suggestions and constant support during this research. Without his proofreading, this thesis will never been possible to read understandably. I am also grateful to Dr. Henny Klein for reading this thesis and giving insightful comments.

The H.1311 is a corridor where I always confuse with its direction after I stepped out from the lift during my first two months in the Harmonie building. Here, the people in Alfa-Informatica work everyday in harmony. I would like to thank them all for the dance, song, and lunch together.

This work is also dedicated to all people in the Indonesia Digital Library Network, that have shared their intellectual resources to public, using the Ganesha Digital Library software. Without them, this work will never exist.

Alhamdulillah, this thesis can be finished. A very special thank I would like to dedicated to Agnes, my very patient and strong wife, for caring my two children, Lala and Malik, and constantly supporting me when we are in a distance. I know this is a very difficult period in your life. Finally, my greatest gratitude goes to my parents for their love and care.

Groningen, The Netherland   Ismail Fahmi
January 2004

Chapter 1

Introduction

In recent years, many research projects, workshops, and conferences have focused on digital libraries as a relatively new field [1]. Digital libraries have been seen as a means of making distributed information readily and easily available. One of the efforts is the Open Archive Initiative (OAI) that provides universal mechanisms to integrate metadata from distributed digital libraries. Using the OAI protocol, one can harvest metadata from other digital libraries and store it into one's own database, or provide access to one's metadata for other digital libraries [2]. A collection of millions of metadata entries that represents million of documents is then ready for users. The question arises of how retrieval of such information can be facilitated [3].

When search engines became available on the Internet, many users felt that they only needed to enter the appropriate terms into a search engine, and the desired information would be retrieved. But inexperienced searchers discover difficulties when they are provided with millions of hits. It leads to the recognition that subject classifications may become more important than ever. Organizing information into subjects, or *taxonomies*, provides users with a significant improvement in retrieval [4].

One of the digital library systems, examined in this thesis, is the Ganesha Digital Library (GDL) software which is used by the Indonesian Digital Library Network (IndonesiaDLN) [5]. This system implements the OAI protocol for metadata gathering and dissemination over more than 45 digital libraries in Indonesia, provides metadata searching, and classifies the metadata into categories for browsing.

Unlike other on-line directories, such as Yahoo! which uses trained information professionals to organize and categorize collections, the digital libraries in this network are

maintained by people with many backgrounds [6]. This raises another problem, that digital libraries will contain incorrect or inconsistent classification, so in turn browsing through category hierarchies can be frustrating and time consuming.

Apparently, the manual classification of the IndonesiaDLN metadata is noisy [1]. Realizing the shortcoming of the manual classification method, we ask **how good the performance of the machine learning algorithms to perform automatic text classification on the digital libraries metadata can be**.

In order to bring us closer to this goal, in this first chapter we outline how machine learning techniques have been used for information retrieval. Then, we describe how machine learning algorithms approach to the text classification. In the rest of this chapter we describe briefly the motivations, organization, and contributions of this thesis.

1.1   Machine Learning for Information Retrieval

Machine learning was applied in information retrieval long before the recent advances of the Web. It has been used in the following areas: information extraction, relevance feedback, information filtering, text classification, and text clustering. In this section, we summarize the research in these areas [7], while emphasizing the text classification area.

Information extraction techniques identify useful information from text documents automatically. The most widely studied sub-field is named-entity extraction. For example, we can extract persons, locations, organizations, dates, times, number expressions, currency amounts, email addresses, and Web addresses. The machine learning algorithms for this extraction include neural networks, decision trees, Hidden Markov Models, and Maximum Entropy models.

Relevance feedback reformulates search queries based a user's evaluation of previously retrieved documents. The main assumption is that the relevance of documents to a particular query is represented by a set of similar keywords [8]. Probabilistic techniques have been used in relevance feedback by estimating the probability of the relevance of a given document to a user.

Information filtering and recommendation techniques are similar to the relevance feedback. The difference is that while relevance feedback allows user to reformulate their search

---

[1]A discussion in an IndonesiaDLN's mailing list in 2002 about users' experiences in browsing articles through the IndonesiaDLN's category trees has figured out this problem. Users got difficulties in finding articles because the placement of the articles in the category trees confused them.

queries, information filtering techniques learn a user's interest based on his or her evaluation of system suggestions. One example of this technique combined with the recommendation and the text classification system has been developed for book recommending systems [9]. The technique used the Naive Bayes algorithms.

Text classification and *text clustering* techniques have been studied extensively in traditional information retrieval literature. Text classification is a supervised learning that classifies textual documents according to their predefined categories, while text clustering is unsupervised learning that groups documents into categories dynamically according to their similarities. Machine Learning techniques have been applied extensively to text classification and text clustering. Five text classification methods that have been widely examined are Support Vector Machines (SVM), k-Nearest Neighbor (kNN), Naive Bayes, Neural Network (NNet) [10], and Rocchio [11]. It has been shown that SVM achieved the best performance on the Reuters-21578 data set [10]. For text clustering, the Estimation Maximization (EM) algorithm has became the basis for many unsupervised clustering algorithms [12].

One aims in machine learning to construct algorithms for predicting a certain label of unseen examples after learning a limited number of training examples. In the absence of any additional assumptions, this task cannot be solved since unseen situations might have an arbitrary label. The assumptions about the nature of the label are subsumed in the term inductive bias [13]. Different classifiers may have different inductive biases. On one hand, an inductive bias enables the classifier to classify unseen instances, and on the other hand it may be the cause of classification errors when it is inconsistent with the labels [14].

To attain the goals of this thesis, three machine learning techniques will be applied, namely **Naive Bayes, kNN,** and **ID3**. Their importance and a more detailed description of their techniques is presented in Chapter 2.

1.2   Text Classification

The automated text classification can be defined as assigning pre-defined category labels to new documents based on the likelihood suggested by a training set of labelled documents [10]. The machine learning paradigm has became one of the main approaches in this area. It generates a classifier from the training set based on the characteristics of the documents already classified. Then it uses the classifier to classify the new documents [11]. Using this

approach, digital libraries that don't have information professionals can use other digital libraries' classification on the same knowledge domain as the training set, to classify their documents.

The question arises of how a document be assigned to the most likely category, with the highest accuracy and the lowest computational cost [11]. Beside the limitation of the techniques to classify the documents correctly, the correctness and consistency of the classifications provided by the information professionals for the training set also influence the accuracy. The last issue leads to the question of how well the current classification has been built at the IndonesiaDLN. We will compare the classification of the IndonesiaDLN metadata with another well known data set, the Reuters-21578.

1.3 Motivation

The two most important motivating factors for the examination of the learning algorithms for text classification in digital libraries are: identifying the most appropriate learning algorithms for text classification in the digital libraries; and improving usability and services of the digital libraries.

1.3.1 Improving The Usability and Services of the Digital Libraries

The rapid growth of information provided on the Internet has increased popularity of search engines. Having an information need, one will go to Google or Yahoo! to find links to the related web pages or one will type queries into Citeseer for scientific articles. Unfortunately, most of the digital library database entries are not indexed by the search engines. While it is easy to harvest the digital libraries databases using a particular protocol, such as the OAI protocol, it is very likely that the search engines have not implemented the protocol to crawl the digital libraries databases. In order for the databases to be searched directly by the search engines (e.g. by Google), they should have an application that serves database entries as web pages or exports to XML files on a web server [15].

The ability of the digital libraries to retrieve information according to user's queries with high accuracy and efficiency is of vital importance. A million records of metadata can be harvested, stored, and then integrated from many digital libraries. The final question is how usable the collections are for the users. Whether or not users from various backgrounds can find valuable information somewhere in the collection is very important.

Machine learning techniques can help to improve the digital library services such as ad hoc retrieval, routing and filtering, and browsing. Although machine learning techniques have been researched extensively for long time in these fields there is still room to improve various aspects of the learning algorithms and to apply them in many new ways.

1.3.2 Identifying the Most Appropriate Learning Algorithm

It is clear that no learning algorithm is optimal for all tasks [16]. One should find a situation where the algorithms will perform well. After examining the accuracy of the algorithms over the digital library metadata collections, we can identify the most appropriate learning algorithm to be applied. This motivation leads to the main goal of this thesis.

1.4 Contributions

In this thesis, we provide the following contributions:

- Identify the most appropriate learning algorithm for automatic text classification of the IndonesiaDLN metadata collection.

- Compare the automatic text classification results for the IndonesiaDLN metadata collection data set with the Reuters-21578 data set.

1.5 Evaluation Framework and Thesis Organization

In this section, we describe the evaluation framework and the organization of this thesis.

1.5.1 Evaluation Framework

Figure 1.1 shows the evaluation framework used to examine the learning algorithms in this thesis. The framework consists of three major stages: document preprocessing, classification and testing, and analysis. The sub-stages are shown beneath each major stage.

1.5.2 Thesis Organization

This thesis consists of four chapters. Chapter 1 (this chapter) presents the problems, their importance, and our research questions.
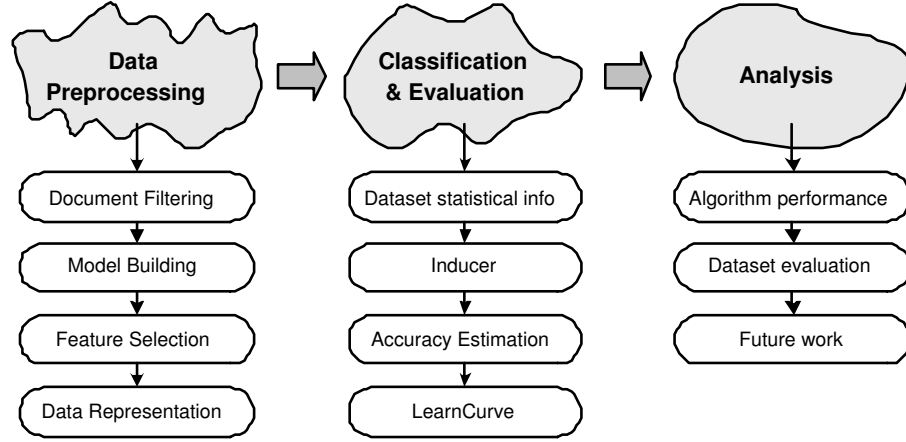
Figure 1.1: Evaluation framework used in this thesis.

Chapter 2 describes definitions and methodology of the research. First, we review some existing works on the automatic text classification and give a brief introduction to the learning algorithms. Next, we describe the characteristics and processing of the data sets, the classifiers and the system environment, and a method to measure the accuracy.

Chapter 3 describes the result of the experiments and the evaluation. We compare the performance of the learning algorithms using two data sets, and identify one that is most appropriate for the GDL application. We also discuss how the learning algorithm can be implemented in the IndonesiaDLN.

Chapter 4 summarizes what we have learned and the possibilities for future work.

Chapter 2

Definitions and Methodology

We begin this chapter by defining the terms used throughout this thesis. We briefly describe text classification, the learning algorithms that are used in this thesis, their importance, and the selection of the learning software. We also describe the selection of the datasets, their properties, and how to represent the datasets for the learning process. We conclude this chapter with the accuracy measures used in this thesis.

2.1   Definitions

The type of learning task in this thesis is **supervised learning**. Given a pre-defined classification (labelling) of some document collection, the learning program is to generate a **classifier**. We can use the classifier to classify novel **instances** with the goal of predicting labels correctly. The quality of the learning program for a specific task can be evaluated from its accuracy, performance, and other properties. Table 2.1 describes the learning problem of the automatic text classification in the IndonesiaDLN.

The automatic text classification learning problem:

Task: classifying the metadata into predefined categories.
Performance measure: F_1-measure and percent of errors of the predicted labels.
Training experience: pre-defined classification of a metadata set by information professionals.

Table 2.1: The automatic text classification learning problem. The problem uses discrete-valued target functions.

A **metadata instance** in a digital library is a simple description about a digital resource. For example, a metadata instance describes title, author, abstract, date, list of files, etc of an electronic paper stored in the digital library.

An **instance** or example is a fixed list of **feature** values. An instance in this task describes a metadata instance as the basic entity we are dealing with.

A **feature** or **attribute** describes characteristics of an instance. A feature in this task is a **word** occurred in the metadata. The value type of the feature is **continuous** and represents the frequency of occurrence of the word in the metadata.

A **label** is a special feature that describes a category or classification of a metadata. For example, acq, alum, heat, etc are some labels used by the Reuters-21578 dataset. A **dataset** is a set of labelled instances. An **unlabelled instance** is an instance without label.

A **classifier** is a function that maps an unlabelled instance to a label. We shall examine concrete classifier functions below. For example, the ID3 algorithm stores the function in a decision tree that maps an unlabelled instance to a category by following a path from the root to a leaf an return a category at the leaf.

An **inducer** or **a learning algorithm** generates a classifier after learning a dataset, and uses the classifier to classify the unlabelled instances [17]. Three inducers that will be evaluated in this thesis are ID3, Instance-Based learning, and Naive Bayes.

2.2 Text Classification in the IndonesiaDLN

The metadata and its digital resources represented in the IndonesiaDLN is classified using a hierarchical category model. This model has been extensively used by many web portals such as Yahoo! and the Google Web Directory. This representation is useful for inexperienced users to get an overview of documents under a certain category.

Figure 2.1 shows a category hierarchy that lists names of digital libraries and provides links to their metadata collection. The metadata is generated and contributed by the digital library servers in the IndonesiaDLN. From this category, users select sub-categories from the institutions listed, and then select articles to read.

Figure 2.2 (a) shows the category hierarchy model of the IndonesiaDLN. Every institution adds their own categories under the Institution Categories. This could produce long hierarchy trees making browsing a time-consuming and frustrating process. The browsing

Categories (By Publisher code)

ACPTUNSYIAH / Universitas Syah Kuala (49)
GDLHUB / GaneshaDL Central Hub (2)
IJPTUNCEN / Universitas Cendrawasih (549)
JBKMRGGREY / KMRG ITB (110)
JBPKINSTY / The Indonesian Institute of Science and Society (6)
JBPTIAIN / IAIN SUNAN GUNUNG DJATI (2)
JBPTIPBMMA / Magister Manajemen Agribisnis - IPB (134)
JBPTITBPP / ITB Central Library (1560)
JBPTITBTI / Departemen Teknik Industri ITB (5)
JBPTUPI / Universitas Pendidikan Indonesia (2)
JIPTIAIN / IAIN Sunan Ampel Surabaya (5)
JIPTITNPP / ITN Malang Central Library (5)
JIPTSTIKI / Perpustakaan STIKI Malang (3)
JIPTUMM / Universitas Muhammadiyah Malang (184 1)
JIPTUNAIR / Universitas Airlangga Surabaya (4)
JIPTUNMERPP / UNMER Central Library (69)
JKLPNDPDII / Pusat Dokumentasi Dan Informasi Ilmiah LIPI (2)
JKPKBPPK / Badan Litbang Kesehatan (1609)
JKPKELNUSA / PT ELNUSA Tbk (1356)
JKPKFORLINK / Dana Mitra Lingkungan - FORLINK (51)
JKPKJPLH / JARINGAN PERPUSTAKAAN LINGKUNGAN HIDUP (119)
JKPKKLH / Kementerian Lingkungan Hidup (1)

JKPKLEMHANNAS / Lembaga Ketahanan Nasional RI (19)
JKPNPNRI / Perpustakaan Nasional Repu blik Indonesia (1)
JKPTBINUS / Bina Nusantara University (107)
JKPTIAINPP / IAIN SYARIF HIDAYATULLAH (10)
JKPTPERBANAS / STIE PERBANAS (7)
JKPTUAIPP / Perpustakaan Pusat UAI (1)
JKPTUTPP / Perpustakaan Pusat Universitas Terbuka (57)
JKPTYARSI / Universitas Yarsi (175)
JKUNUAJ / Atma Jaya Catholic University (4)
JTPTIAIN / IAIN Wali Songo Semarang (22)
KBPTUNTAN / Universitas Tanjungpura (12)
KSPTIAIN / IAIN Antasari Banjarmasin (54)
LAPTIAIN / IAIN Lampung (146)
RIPTIAIN / IAI N Sulthan Syarif Qasim (8)
SAPTUNSRAT / Universitas Sam Ratulangi (314)
SBPTIAIN / IAIN Padang (16)
SGPTUNHALU / Universitas Haluoleo (4)
SNPTIAIN / IAIN Auludin Makasar (42)
SSPTIAIN / IAIN Raden Fatah Palembang (38)
SUPTIAIN / IAIN Sumatera Utara (31)
TESTINS TITUTION / Institution Name Corp (4)
YOPTIAIN / IAIN Sunan Kalijaga (1)

Figure 2.1: The category hierarchy contains the list of the digital libraries in the IndonesiaDLN. The number represents the total amount of metadata submitted to the IndonesiaDLN.

facility is intended to show the richness or poorness of the collections shared by the institutions. It is expected that the institutions will be motivated to share more knowledge through the IndonesiaDLN, because their collections are disseminated and presented to all other digital libraries in the network [6].

Figure 2.2 (b) shows an example of the documents classification presented by JBP-TITBPP (ITB Central Library) digital library server. The classification of the 25 articles is based on their document types, department names, study programs, and publication years [18].

2.3 The Learning Algorithms

In this section, we introduce three inducers that represent the three families of learning algorithms: **decision tree, instance-based learning**, and **Bayesian**. We describe the
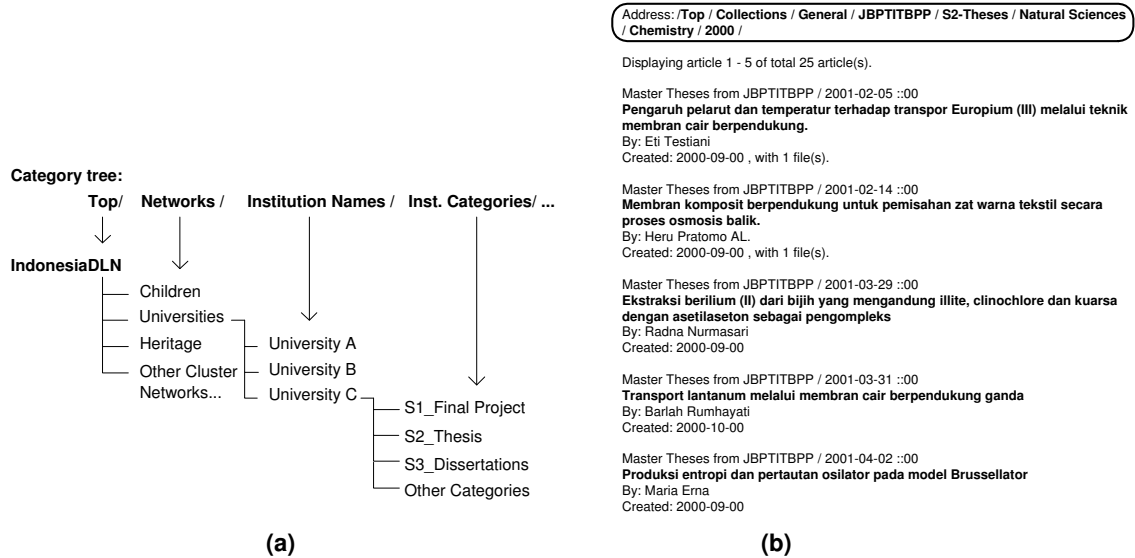
Figure 2.2: (a) The category hierarchy model in the IndonesiaDLN. (b) Example of the documents classification in the IndonesiaDLN.

importance of the algorithms and their basic ideas. Further discussion about these algorithms is provided in chapter 3. Anyone who wants more description about these learning algorithms can refer to [12].

2.3.1 Decision Tree

Decision tree learning is one of the most popular and practical methods currently used for inductive bias. This method approximates classification to the discrete-valued label that is robust to noisy data. Several methods have been proposed for constructing decision trees, including ID3, C4.5, and CART. Among these learning algorithms, we chose ID3 because it corresponds approximately to the basic decision tree learning algorithm, while other learning algorithms has been extended from this basic algorithm to achieve better performance.

The basic idea of the ID3 algorithm is to construct an inducer in the form of a decision tree, from the top node to the leaf nodes. Given the partial decision at one node, the decision tree user then follows the next branch until he reaches the leaf node or the final classification. ID3 uses a statistical property, called *information gain*, to select the attribute or word that is most useful for classifying examples at each step while growing the tree. We

define information gain formally as follow:

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} Entropy(S_v) \qquad (2.3.1)$$

$$Entropy(S) \equiv \sum_{i=1}^{c} -p_i log_2 p_i \qquad (2.3.2)$$

As seen in the equation 2.3.1, the information gain $Gain(S, A)$ of an attribute A relative to a collection of examples S, where $Values(A)$ is the set of all possible values for attribute A and S_v is the subset of S for which attribute A has value v, is simply the reduction in entropy caused by partitioning the examples according to this attribute. The next equation defines entropy of S, if the classification label take on c different values, where p_i is the proportion of S belonging to class i. Because entropy specifies the minimum number of bits of information needed to encode the classification of an arbitrary member of S, the logarithm in this equation is base 2.

Figure 2.3 shows an illustrative example of a training set taken from the Reuters-21578 for a text classification task. The category classes are economic subject categories, including *acq* (acquisition), *alum* (aluminum), *bop* (balance of payments), *carcass* (animal body), *cocoa*, *coffee*, *copper*, *cotton*, *cpi* (consumer price index), *crude*, and *earn*. The sample has attributes *ct* (cents), *shr* (shares), *net* (advantage, excess of revenues), *qtr* (quarter), *rev* (revenues), *trade*, *note*, *oil*, *pct* (percent), and *compani* (stemmed from *company*) with continuous values that indicate the number of occurrence of the attributes in each document instance.

The ID3 algorithm processed the training set and generated a classifier in a form of decision tree as shown by Figure 2.4. ID3 uses this decision tree to classify novel instances.

2.3.2 Instance-Based Learning

Instance-based learning algorithms are sometimes referred to as "lazy" learning methods because they delay processing until a new instance must be classified. This characteristic distinguished instance-based learning from other learning algorithms that construct an explicit classifier as a description of the target label when training examples are provided. We choose k-Nearest Neighbor (kNN) to represent this learning algorithm family because it is the most basic instance-based method and has become the most widely used approach. It is one of the top performing methods on the Reuters-21450 benchmark [10].

No	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	Class
1	0	0	0	0	0	2	0	0	5	2	acq
2	2	0	1	0	0	0	0	0	1	4	acq
3	0	0	1	0	0	1	0	0	2	4	acq
4	0	0	0	0	0	1	0	2	2	4	acq
5	0	0	0	0	0	0	1	0	7	3	acq
6	0	0	0	0	0	1	0	0	8	2	alum
7	0	0	0	0	0	2	0	0	2	1	alum
8	0	0	0	0	0	1	0	2	2	0	bop
9	0	0	0	0	0	7	0	0	4	0	bop
10	0	0	0	0	0	2	0	0	1	1	bop
11	0	0	0	0	0	0	0	0	3	0	carcass
12	0	0	0	0	0	1	0	0	1	0	cocoa
13	0	0	0	0	0	3	1	0	1	1	cocoa
14	0	0	0	0	0	1	0	0	2	0	cocoa
15	0	0	0	0	0	2	0	1	0	0	coffe
16	0	0	0	0	0	2	0	0	3	0	coffe
17	0	0	0	0	0	1	0	0	4	0	coffe
18	0	0	0	0	0	2	0	0	0	0	coffe
19	2	0	0	0	0	1	0	1	1	4	copper
20	0	0	3	0	0	1	0	0	2	1	copper
21	0	0	0	0	0	1	0	0	1	0	cotton
22	3	0	0	0	0	0	0	0	0	0	cotton
23	0	0	0	0	0	0	0	0	3	0	cotton
24	0	0	0	0	0	0	0	0	10	0	cpi
25	0	0	0	0	0	0	0	0	15	0	cpi
26	0	0	0	0	0	0	0	0	8	0	cpi
27	0	0	0	0	0	0	0	0	14	0	cpi
28	10	0	0	0	0	0	0	5	5	0	crude
29	0	0	0	0	0	0	1	8	0	2	crude
30	0	0	0	0	0	1	0	12	0	10	crude
31	2	0	0	0	0	1	0	6	0	2	crude
32	2	0	0	0	0	0	0	2	2	2	crude
33	2	0	0	0	0	0	0	4	1	1	crude
34	1	0	0	0	0	0	0	5	0	3	crude
35	2	0	0	0	0	0	0	3	0	2	crude
36	4	2	3	1	2	0	0	0	0	0	earn
37	3	4	3	1	2	0	0	0	0	0	earn
38	3	4	4	1	0	0	1	0	0	0	earn
39	6	2	2	1	1	0	1	0	0	0	earn
40	6	4	3	3	2	0	1	0	0	0	earn

Features:
f1=ct; f2=shr; f3=net; f4=qtr; f5=rev; f6=trade; f7=note; f8=oil; f9=pct; f10=compani

Figure 2.3: An example of a training set with 40 instances taken from the Reuters-21578. Each instance has 10 attributes with a target concept.

The basic idea of the kNN algorithm is to simply store the training data, delay the processing, and to classify when a new query instance is encountered. Given a test set, the algorithm retrieves a set of k *most similar* or *nearest* instances from memory and classifies the query instance based on the most common value among the k instances. The *similarity* or *nearest distance* of the neighbors is defined by the Euclidean distance.

We define the feature vector of an instance x as $\langle a_1(x), a_2(x), ... a_n(x) \rangle$. The distance between two instances x_i and x_j is defined as $d(x_i, x_j)$, where $a_r(x)$ denotes the value of the rth attribute of instance x, is formulated by equation 2.3.3. Another method to measure the distance is using cosine value of two document vectors [10].

$$d(x_i, x_j) \equiv \sqrt{\sum_{r=1}^{n}(a_r(x_i) - a_r(x_j))^2} \qquad (2.3.3)$$

2.3.3 Bayesian

The Bayesian learning methods use a probabilistic approach to generate a classifier. The basic assumption is that the probability distributions of the training data together with the test data govern the interested classification. The most practical algorithm in this learning family is the Naive Bayes algorithm. For the text classification task, Naive Bayes is among
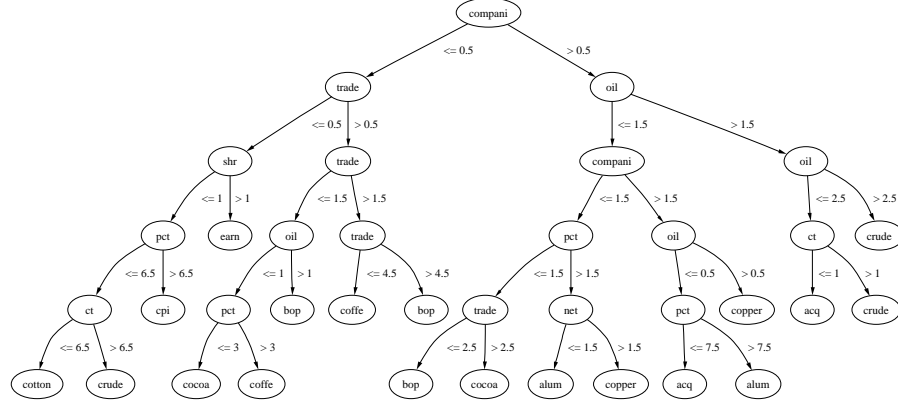
Figure 2.4: The decision tree graph of the training set shown by Figure 2.3. The decision tree was generated by the ID3 Inducer provided by the MLC++ and the graphs was constructed by the *dot* preprocessor application. Each node contains an attribute and each leaf indicates the target concept of its decision path. Which path should be followed to reach a leaf is guided by the numbers that indicate the values of the attribute.

the most effective algorithms.

The basic idea in Naive Bayes approach is to use the joint probabilities of words and categories to estimate the categories of a given document. We derive the Naive Bayes classifier from Bayes's theorem that defines the posterior probability $P(v|D)$ from the prior probability $P(v)$, together with $P(D)$ and $P(D|v)$:

$$P(v|D) = \frac{P(D|v)P(v)}{P(D)} \tag{2.3.4}$$

where $P(v|D)$ is the probability of target value v (classification) given the observed data D. In general, $P(x|y)$ denotes the probability of x given y, while $P(z)$ denotes the unconditional probability of z. The most probable target value given the observed data D, also called *a maximum a posteriori* (MAP) hypothesis, is provided by equation 2.3.5.

$$
\begin{aligned}
v_{MAP} &\equiv argmax_{v \in V} P(v|D) \\
v_{MAP} &= argmax_{v \in V} \frac{P(D|v)P(v)}{P(D)} \\
v_{MAP} &= argmax_{v \in V} P(D|v)P(v)
\end{aligned}
\tag{2.3.5}
$$

$P(D)$ is a constant independent of v so that we can eliminate it from the equation. In our case, given an instance with the attribute values $\langle a_1, a_2...a_n \rangle$, we will classify this instance

to the most probable value, v_{MAP}. We rewrite the equation 2.3.5 for this case into equation 2.3.6.

$$v_{MAP} = argmax_{v_j \in V} P(a_1, a_2...a_n|v_j)P(v_j) \tag{2.3.6}$$

The meaning of "naive" in the "Naive Bayes" is based on the simplifying assumption that the attribute values are conditionally independent given the target value. It means that given the v_j, the probability of the conjunction $a_1, a_2...a_n$ is just the product of the individual attributes' probabilities.

$$P(a_1, a_2...a_n|v_j) = \prod_i P(a_i|v_j) \tag{2.3.7}$$

where a_i denotes the attribute found in the ith position within document j. With this assumption, we can rewrite equation 2.3.6 and produce the approach used by the Naive Bayes classifier.

$$v_{NB} = argmax_{v_j \in V} P(v_j) \prod_i P(a_i|v_j) \tag{2.3.8}$$

Actually, the "naive" assumption for the text classification is incorrect. For example, attribute "library" will have greater probability if preceded by attribute "digital". Fortunately, in practice the Naive Bayes inducer performs well on many text classification problems.

2.4 The Learning Algorithm Tools

Ron Kohavi et al have implemented a library of C++ classes and tools for supervised Machine Learning, called the MLC++ [16]. Other learning algorithm collections are the one developed by Mooney [19], the Consultant [20], and the MLToolbox [21].

We choose the MLC++ for the learning algorithms implementation in this thesis for the following reasons. First, it integrates the algorithms in a C++ library with one unified coding methodology and the decomposition of components into C++ classes. This means that the MLC++ allows more control over the algorithms. Second, it provides an array of tools that give a broad picture of the performance of the algorithm when compared to other algorithms. Some built-in utilities that are useful for examining the algorithms include the Inducers, which run a given inducer on a given data file; Accuracy Estimation

in different methods; Info, which provides basic statistical information about the dataset; and LearnCurve, which generates a learning curve for a given induction algorithm and a dataset [22].

2.5   The Datasets

This section describes the datasets used in this thesis. The first dataset is the IndonesiaDLN metadata collection. This is the main dataset for the evaluation of the learning algorithms for the text classification task in the digital library. The second dataset is the Reuters-21578 which aims is to compare the results with the previous examinations.

2.5.1   IndonesiaDLN Metadata Collection

The IndonesiaDLN metadata collection consists of 7793 metadata instances, collected from about 45 digital library servers in the IndonesiaDLN network since October 2001 until July 2003. The collection represents many types of digital resources as shown by Figure 2.5 (a).
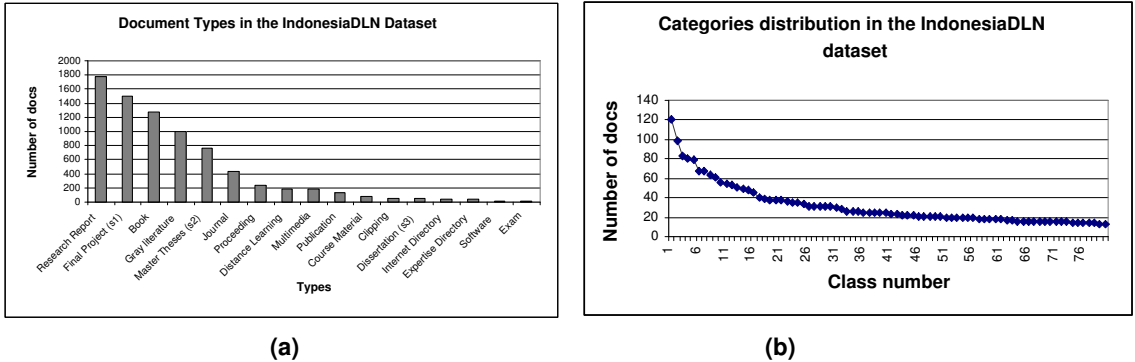
Figure 2.5: (a) The document types in the IndonesiaDLN metadata collection. The most common type, "research reports", contains 1780 metadata instances, while "Exam" only contains 7 metadata instances. (b) Category distribution in the IndonesiaDLN. The X axis is scaled to only include 80 categories from total 489 categories. The most common category, category number 1, contains 120 metadata instances. There are 309 categories that contain two or more metadata items. The rarest 180 categories contain 1 metadata each.

The metadata is categorized into a hierarchical category tree as shown by Figure 2.2. For the text classification task in this thesis, we map every category tree into a non-hierarchical category number as shown by figure 2.6. We eliminate the "year" part of the hierarchy such

as shown by Figure 2.2(b). This process results 489 distinct categories whose distribution is shown by Figure 2.5 (b).

Category Trees	Class
Terbitan_Internal/Data2_-_Dokumentasi/Dokumentasi/	1
Research_Reports/Institute_for_Research/	2
S2-Theses/Engineering/Civil_Engineering/	3
Research_Report/Law/	4
Research_Report/Education/	5
S1-Final_Project/Dept._of_Agriculture/Agribusines	6
Journal/Kedokteran/Jurnal_Kedokteran_YARSI/	7
S2-Theses/Engineering/Highway_Systems_and_Eng./	8
Referensi/Koleksi_Khusus/	9

Figure 2.6: The category trees are mapped into class numbers.

The IndonesiaDLN dataset was preprocessed to produce two dataset representations, i.e. a dataset that represents the collection from the Bandung Institut of Technology (ITB) only, and a dataset that represents the whole IndonesiaDLN collection. Our objective is to see the classification effectiveness over local collection and global collections.

For the ITB dataset, we only use categories that contain 4 or more metadata instances. This elimination produces 64 categories and 729 metadata instances (429 metadata instances for training set, and 300 metadata instances for test set). While for the whole IndonesiaDLN, we only use categories that contain 8 or more metadata instances. This produce a dataset that contains 137 categories and 3082 metadata instances (2009 metadata instances for training set, and 1073 metadata instances for test set).

2.5.2 Reuters-21578 Dataset

In order for our results to be compared with previous works, we also evaluate the algorithms using the news collection from the Reuters-21578 dataset. This corpus has become a new benchmark lately in text classification evaluation [10]. We use the ApteMod version of the Reuters-21578 dataset, that contains 21578 documents [23].

In evaluating the Reuters-21578 dataset, we only use the TOPICS category tag and ignore other category tags such as EXCHANGES, ORGS, PEOPLE, and PLACES. We only processed documents that have at least one category in their TOPICS tag. There are 120 categories that contain one or more documents, and 57 categories with more than 20 documents as shown by Figure 2.7.

The evaluation of learning methods in this thesis will be limited to the classification of the single-label documents. Elimination of the multi-label documents from the Figure 2.7
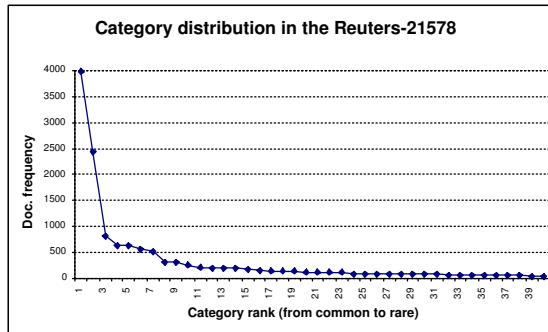
No	Train	Test	Category	No	Train	Test	Category
1	1596	696	acq	27	37	12	jobs
2	31	19	alum	28	4	4	lead
3	22	9	bop	29	11	3	lei
4	6	5	carcass	30	13	5	livestock
5	46	15	cocoa	31	7	4	lumber
6	90	22	coffee	32	6	1	meal-feed
7	31	13	copper	33	206	87	money-fx
8	15	9	cotton	34	123	28	money-supply
9	54	17	cpi	35	24	12	nat-gas
10	3	1	cpu	36	3	1	nickel
11	253	121	crude	37	13	9	orange
12	3	3	dlr	38	13	6	pet-chem
13	2840	1083	earn	39	1	2	platinum
14	4	7	fuel	40	2	3	potato
15	10	8	gas	41	37	12	reserves
16	58	15	gnp	42	19	1	retail
17	70	20	gold	43	31	9	rubber
18	41	10	grain	44	108	36	ship
19	6	4	heat	45	9	6	strategic-metal
20	15	2	housing	46	97	25	sugar
21	7	4	income	47	2	3	tea
22	5	1	instal-debt	48	17	10	tin
23	190	81	interest	49	251	75	trade
24	33	11	ipi	50	19	11	veg-oil
25	26	12	iron-steel	51	14	9	wpi
26	2	1	jet	52	8	5	zinc

(a)   **(b)**

Figure 2.7: (a) Category distribution in Reuters-21578. The X axis is scaled to include only 40 categories. The most common category, category number 1 "earn", contains 3987 documents. The most rare categories, 15 categories, contain 1 document each. (b) List of the categories with the number of documents for the training set and the test set.

(a) resulted 52 categories and 9100 documents (training set 6532 documents and test set 2568 documents) as shown by Figure 2.7 (b).

2.6   Document Preprocessing

In this section we discuss the process of transforming the metadata and news texts into a representation suitable for the learning algorithms. The process of this transformation is shown by Figure 2.8, which mainly uses the Rainbow program from the BOW library [24].
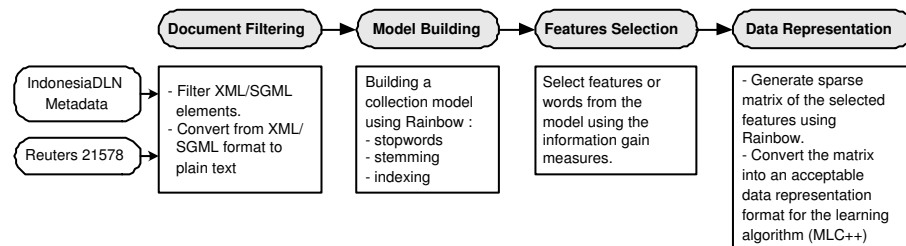

Figure 2.8: The data preprocessing steps. This process converts the original documents in XML and SGML format into a suitable format for the learning algorithms in the MLC++.

2.6.1 Document Filtering

Document filtering is a process of eliminating the irrelevant portions of the document for the text classification. For example, the label, topics, or classification element embedded in the document will be eliminated because they specify values which we wish to induce from the other information.

The IndonesiaDLN metadata is formatted in XML format using the Dublin Core (DC) metadata element set standard [25]. The DC standard consists of 15 main elements such as title, creator, subject, description, relation, and so on. The elements can be expanded into sub-elements.

Figure 2.9(a) shows an example of the IndonesiaDLN metadata. The metadata is enclosed by <DC> and </DC> tags which include several internal element tags. The classification of the metadata made by the information professional is enclosed by <HIERARCHY> and </HIERARCHY> tags. We only include the most relevant elements for text classification, i.e. title, creator, publisher, subject, and description.

The Reuters-21578 collection is distributed in 22 files. Each of the first 21 files (reut2-000.sgm through reut2-020.sgm) contain 1000 documents, while the last (reut2-021.sgm) contains 578 documents. Figure 2.9(b) shows an example of the Reuters-21578 document. The classification of the document is enclosed by <TOPICS> and </TOPICS> tags where in this example the topic is empty. We only use the TITLE and BODY elements for the text classification.

The results of the document filtering are collections of files grouped into directories based on their classification. All documents with the same class label are contained within a directory. This grouping is required by the Rainbow program to build an index of the collection.

2.6.2 Elimination of Stopwords

A stopword is defined as a very frequent word in a text corpus that does not carry discriminative meaning of the document content. Articles, prepositions, and conjunctions are natural candidates for a list of stopwords and normally filtered out as potential index terms [26]. Elimination of stopwords has an additional important benefit, namely reducing the size of features.

For documents in English, we use the SMART system's list of 524 common words, like

```
<dc>
    <title>Tasamuh, bersama Prof. Dr. Alwi Shihab</title>
    <title>
        <alternative>Tasamuh, with Prof. Dr. Alwi Shihab</alternative>
    </title>
    <creator>Alwi Shihab, Prof. Dr.</creator>
    <publisher>JBPTITBPP</publisher>
    <subject>
        <keywords>eksklusivisme, inklusivisme, pluralisme</keywords>
    </subject>
    <description>Dalam setiap agama akan selalu ada kelompok-kelompok
        pemikiran dalam memandang agamanya maupun agama mitra
        dialognya ....
    </description>
    <contributor>NARATOR : Evi Yuliani; digitized by Indro</contributor>
    <date>1997-07-20</date>
    <date>
        <modified>2000-09-26 ::00</modified>
    </date>
    <type>dlearn</type>
    <type>
        <schema>general</schema>
    </type>
    <identifier>itb-dist-salma-1997-Alwi-tasa</identifier>
    <identifier>
        <hierarchy>/Distance_Learning/Religion/Salman_Mosque/
            Ceramah_Umum/</hierarchy>
    </identifier>
    <source>Studium Generale</source>
    <language>Bahasa Indonesia</language>
    <relation>
        <count>1</count>
    </relation>
    <relation>
        <no>1</no>
        <datemodified>2003-04-15 11:08:12</datemodified>
        <haspart>itb-dist-salma-1997-Alwi-tasa-part2.RM</haspart>
        <haspath>disk1/1/itb-dist-salma-1997-Alwi-tasa-part2.RM
            </haspath>
        <hasfilename>itb-dist-salma-1997-Alwi-tasa-part2.RM
            </hasfilename>
        <hasformat>application/vnd.rn-realmedia</hasformat>
        <hassize>7726181</hassize>
        <hasuri>/download.php?
            f=/disk1/1/itb-dist-salma-1997-Alwi-tasa-part2.RM</hasuri>
    </relation>
</dc>
```

(a)

```
<REUTERS TOPICS="YES" LEWISSPLIT="TRAIN" CGISPLIT="TRAINING-SET"
OLDID="16396"
 NEWID="1076">
<DATE> 3-MAR-1987 11:00:25.62</DATE>
<TOPICS><D>acq</D></TOPICS>
<PLACES><D>italy</D><D>west-germany</D><D>ussr</D></PLACES>
<PEOPLE></PEOPLE>
<ORGS></ORGS>
<EXCHANGES></EXCHANGES>
<COMPANIES></COMPANIES>
<UNKNOWN>
&#5;&#5;&#5;RM
&#22;&#22;&#1;f0664&#31;reute
u f BC-ITALY'S-BNL-NEGOTIATI   03-03 0097</UNKNOWN>
<TEXT>&#2;
<TITLE>ITALY'S BNL NEGOTIATING PURCHASE OF GERMAN BANK</TITLE>
<DATELINE>    ROME, March 3 - </DATELINE><BODY>Italy's state-
owned &lt;Banca Nazionale Del Lavoro-BNL> said it is negotiating
to buy a West German bank as part of its foreign expansion
policy.
    BNL president Nerio Nesi told a news conference the Italian
bank was currently involved in talks but declined to name the
German institution.
    He said the takeover move could be seen as BNL's reply to
Deutsche Bank AG &lt;DBKG.F>, which entered the Italian market
in December 1986, with the purchase of BankAmerica &lt;BACN>
subsidiary &lt;Banca D'America e D'Italia>.
    Nesi said BNL had also approved a 200 mln dlr credit line
to the Soviet Union aimed at enabling Soviet companies to pay
for Italian imports. He gave no further details.
    BNL officials said the group had also decided to increase
its activities in the Soviet Union by opening a representative
office in Moscow this month through its subsidiary lt;Sogecred>,
which specialises in Italian-Soviet trade.
 REUTER
&#3;</BODY></TEXT>
```

(b)

Figure 2.9: (a) An example of metadata from the IndonesiaDLN. The metadata is formatted in XML using the Dublin Core metadata element set standard. (b) An example of the Reuters-21578 document. The file is formatted in SGML.

"the" and "of". While for documents in Indonesian Language, we use the stopwords list of 556 common words from [27], like *sebuah* 'a' and *dari* 'of, from'.

2.6.3   Word Stemming

Stemming is a process of removing word's affixes to generate word stems. For example, the word *represent* is the stem of the variants *represents, representing, representation,* and *represented* [26]. Stemming also reduces the number of features. Many research projects on text classification use stemming for their document preprocessing such as in [10] and [28].

We apply the Porter Algorithm to stem the English documents because of its simplicity and elegance [29]. In this thesis, we don't apply stemming to documents in Indonesian

language. To see the effect of stemming, we make two types of dataset, i.e. a dataset with stemming and dataset without stemming.

2.6.4 Feature Selection

Feature selection is a process of removing undiscriminative terms from documents in order to improve classification accuracy and reduce computational complexity. Several feature selection methods can be applied including *document frequency threshold, information gain, X^2-statistics, mutual information,* and *term strength.* We use *information gain* for feature selection. It measures the number of bits of information obtained for category prediction by knowing the absence or presence of a term in a document [30]. Previous work on feature selection has indicated that such information theoretic approaches are quite effective for text classification [31][30].

Let $c_1...c_m$ denote the set of possible classes in the target space. The information gain in 2.3.1 of a two-valued term t can be rewritten as:

$$G(t) = -\sum_{i=1}^{m} P(c_i)logP(c_i) + P(t)\sum_{i=1}^{m} P(c_i|t)logP(c_i|t) + P(\bar{t})\sum_{i=1}^{m} P(c_i|\bar{t})logP(c_i|\bar{t}) \quad (2.6.1)$$

where $P(c_i)$ can be estimated from the fraction of documents in the total collection that belongs to class c_i and $P(t)$ from the fraction of documents in which the term t occurs. $P(c_i|t)$ can be computed as the fraction of documents that also are members of class c_i that have at least one occurrence of term t and $P(\bar{t})$ is the fraction of documents in which term t fails to appear.

To get a list of features that have highest information gain, we use the rainbow program with `--print-word-infogain` (or `-I`) option. We generated feature sets with 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140, 150, 200, 300, 400, 500, 600, 700, 800, 900, and 1000 features. The 20 features with the highest information gain for the Reuters-21578 datasets is shown by Figure 2.10(a).

2.6.5 Data Representation

Data file format that MLC++ recognizes are based on Quinlan's C4.5 format, which share the same *data* file format with Irvine database [17]. A dataset consists of three files: *data, test,* and *names.* The *data* file and *test* file contain labelled instances, one instance per line.

The *names* file defines the schema that allows parsing the data file. Figure 2.10(b) shows the name definition file for the Reuters-21578 dataset. We can see that the attributes list is defined by the words with the highest information gain as shown by Figure 2.10(a).

Figure 2.10: The data file format used by MLC++. (a) Example of the first 19 features with the highest information gain for the Reuters-21578 dataset. (b) The ".names" file for the Reuters-21578 dataset with 100 attributes. (c) The ".data" file for the Reuters-21578 dataset with 100 attributes.

The *data* file (training set) describes dataset instances that are used to train the algorithm. The *test* file (test set) describes datasets instances that are used to test performance of the algorithm. Each instance is composed of comma-separated attribute values, a label, and (finally) a period. The attribute values are frequency of occurrence of the attributes in a document or instance. Figure 2.10(c) shows a portion of the data file for the Reuters-21578 datasets.

2.7   Effectiveness Measures

In this section, we describe effectiveness measures to evaluate the text classification methods. In term of information retrieval, effectiveness is purely a measure of ability of the system to satisfy user in term of the relevance of documents retrieved [32]. For text classification, we define it as a measure of ability of the system to classify documents into their relevant categories. We begin this discussion with the *contingency table* model, which leads to the most widely used effectiveness measures.

2.7.1   The Contingency Table

The Yahoo! Dictionary defines contingency table as a statistical table that shows the observed frequencies of data elements classified according to two variables, with the rows indicating one variable and the columns indicating the other variable [33]. We will generate two important measures of the system's effectiveness from this kind of table, which is not really a contingency table at all [32], namely *recall* and *precision*.

Consider a simple contingency table from a binary decision system as shown by Table 2.2. The system makes n binary decisions, each of which has exactly one correct answer, either Yes or No [34]. Each entry specifies the number of decisions of the specified type. For example, a is the number of times the system decided Yes, where Yes was the correct answer. And b is the number of times the system decided Yes (incorrect decision), where No was the correct answer.

Category		Expert Judgements		
		Yes	No	
Classifier	Yes	a	b	$a + b$
judgments	No	c	d	$c + d$
Total		$a + c$	$b + d$	$a + b + c + d = n$

Table 2.2: The contingency table for a set of binary decisions.

The following are two important measures of effectiveness that are derived from the contingency table:

- **Recall** is the number of categories correctly assigned divided by the total number of correct categories that should be assigned:

$$R = a/(a + c) \qquad\qquad (2.7.1)$$

- **Precision** is the number of categories correctly assigned divided by total number of categories assigned:

$$P = a/(a + b) \qquad (2.7.2)$$

The effectiveness measures could be misleading if we examined the recall and precision alone. We may sacrifice precision to obtain a high recall, and vice versa. To summarize and make a composite measures, we use the F_1-measure as an evaluation criterion, which was initially introduced by van Rijsbergen [32]:

$$F_1 = \frac{2PR}{P + R} \qquad (2.7.3)$$

In this thesis, we use M-ary decision instead of the binary decision. We can calculate the precision and recall from the confusion matrix that is similar to the contingency table, but the table is expanded for the M-ary decision. A confusion matrix is a tool aiding in the analysis of misclassification [35]. One can find that tokens belonging to a specific class are misclassified to other classes. From the confusion matrix one also can determine which other classes tend to be chosen. An example of a confusion matrix produced by MLC++ is shown by Figure 3.2.

2.7.2 Averaging Techniques

For evaluating the global effectiveness across categories, there are two ways of computing effectiveness namely *macro-averaging* and *micro-averaging* [28, 36, 34, 26]. Macro-averaging computes the effectiveness on a per-category basis, then averages over categories. Micro-averaging computes the average effectiveness over all individual decisions as a single group. Macro-averaging gives equal weight to each category, while micro-averaging gives equal weight to every document. Intuitively, we can understand that macro-averaging scores are more influenced by the classifier's effectiveness on rare categories, while micro-averaging scores tend to be dominated by the effectiveness on common categories [10].

The M-ary decision that is used in this thesis leads to the recall and precision calculation using the macro-averaging technique. They are calculated from the confusion matrix per-category basis, then are averaged over the categories. When we calculate over all instances as a group, we will get the total correct classification and total incorrect classification. Thus, the evaluation based on the last calculations will produce **Accuracy** and **Error** measurements.

2.7.3 Accuracy Estimation

Other measure commonly used in machine learning literature, such as *accuracy (A)* and *error (E)*, are not widely used in TC [36]. As shown by equation 2.7.4, the large value that their denominator has in TC makes them much more insensitive to variations in the number of correct decision $(a + d)$ than R and P.

Given the contingency table as shown in Table 2.2, accuracy and error are defined as:

$$A \quad = \quad \frac{a+d}{a+b+c+d} \tag{2.7.4}$$

$$E \quad = \quad \frac{b+c}{a+b+c+d} = 1 - A \tag{2.7.5}$$

The main effectiveness measure provided by MLC++ is the accuracy estimation (error rate). MLC++ provides a LearnCurve utility that generates a learning curve for a given induction algorithm and a dataset. Given a dataset, the x-axis represents the number of training instances and the y-axis represents the accuracy when trained on the given number of instances and tested on the unseen instances [16].

2.8 Utilities

We use the following utilities supported by MLC++ to get effectiveness measurement results:

- **Inducer**, runs the given inducer on the given data file and produce these reports: instance counts, classification counts, generalization accuracy, the overall accuracy, and the confusion matrix. Output of this utility will be processed to get the recall and precision for different feature sizes.

- **Info**, provides basic statistical information about the dataset. It reports the number of instances in the ".data", ".test", and ".all" files. It also reports the class probabilities, the number of attributes, and their types.

- **LearnCurve**, generates a learning curve for a given induction algorithm and a dataset.

Chapter 3

Results and Evaluation

In this chapter, we provide the basic statistical information about the datasets, the measurement results, and evaluation of the learning effectiveness based on recall, precision, F_1-measure, and the inducers' learning curve.

3.1   The Datasets

We use three datasets in this thesis, i.e. the collection of metadata from ITB (729 instances), the whole metadata collection from the IndonesiaDLN (3082 instances), and the Reuters-21578 document collections (9100 instances). From each dataset, we generate 24 new datasets in different number of features as listed by Figure 3.1(b).

   We use the **Info** utility of MLC++ to get the basic statistical information about the datasets. One item of information derived from the statistics is the number of duplicate or conflicting instances as shown by Figure 3.1. The duplicate instances occurs when instances with a set of features have the same feature frequency profiles and the same classes. The duplicate instances show us the similarity of the documents according to the selected features and target values. The conflicting instances occur when instances with the same feature frequency profiles are labelled with different classes.

   Example of a dataset with duplicate instances is shown by Figure 3.1(a). It shows 2 subsets of 26 instances with 10 features from the Reuters-21578 dataset. The first subset was generated without stemming and consists of the following features: *cts, shr, net, qtr, revs, trade, note, oil, pct,* and *company.* The second subset was generated with stemming and consists of the following features: *ct, shr, net, qtr, rev, trade, note, oil, pct,* and *compani.*

```
| File: reuters-10.all          | File: reuters-10.all
| Without stemming              | With stemming

0,0,0,0,0,0,0,0,2,4,acq.        0,0,0,0,0,0,0,0,2,4,acq.
0,0,0,0,0,0,0,0,0,7,acq.        0,0,0,0,0,1,2,0,0,9,acq.
0,0,0,0,0,0,0,1,0,0,acq.        0,0,0,0,0,0,0,1,0,1,acq.
0,0,0,0,0,0,0,0,2,0,acq.        0,0,0,0,0,0,0,0,2,0,acq.
0,0,0,0,0,0,0,0,2,5,acq.        0,0,0,0,0,0,0,0,2,5,acq.
0,0,0,0,0,0,0,0,3,1,acq.        0,0,0,0,0,0,0,0,3,1,acq.
0,0,0,0,0,0,0,0,0,0,acq.        0,0,0,0,0,0,0,0,0,0,acq.
0,0,0,0,0,0,0,0,0,1,acq.        0,0,0,0,0,0,0,0,0,1,acq.
0,0,0,0,0,0,0,0,2,0,acq.        0,0,0,0,0,0,0,0,2,0,acq.
0,0,0,0,0,0,0,0,1,1,acq.        0,0,0,0,0,0,0,0,1,1,acq.
0,0,0,0,0,0,0,1,5,3,acq.        0,0,0,0,0,0,0,1,5,3,acq.
0,0,0,0,0,0,0,0,0,0,acq.        0,0,0,0,0,0,0,0,0,0,acq.
0,0,0,0,0,0,0,0,0,0,acq.        0,0,0,0,0,0,0,0,0,0,acq.
0,0,0,0,0,0,0,0,1,2,acq.        0,0,0,0,0,0,0,0,1,2,acq.
0,0,0,0,0,0,0,0,1,0,acq.        0,0,0,0,0,0,0,0,1,0,acq.
0,0,0,0,0,0,0,0,2,2,acq.        0,0,0,0,0,0,0,0,2,2,acq.
0,0,0,0,0,0,0,0,0,0,acq.        0,0,0,0,0,0,0,0,0,0,acq.
0,0,0,0,0,0,0,0,10,2,acq.       0,0,0,0,0,0,0,0,10,5,acq.
0,0,0,0,0,0,0,0,2,0,acq.        0,0,0,0,0,0,1,0,2,0,acq.
0,0,1,0,0,0,0,0,1,0,acq.        0,0,1,0,0,0,0,0,1,0,acq.
0,0,1,0,0,0,0,0,4,2,acq.        0,0,1,0,0,0,1,0,4,2,acq.
0,0,0,0,0,0,0,0,1,1,acq.        0,0,0,0,0,0,0,0,1,2,acq.
0,0,1,0,0,0,0,0,0,0,acq.        0,0,1,0,0,0,0,0,0,0,acq.
0,0,1,0,0,0,0,0,2,2,acq.        0,0,1,0,0,0,0,0,2,3,acq.
0,0,0,0,0,0,0,0,2,1,acq.        0,0,0,0,0,0,0,0,2,1,acq.
0,0,0,0,0,0,0,0,1,1,acq.        0,0,0,0,0,0,2,0,1,1,acq.
...                            ...
```

(a)

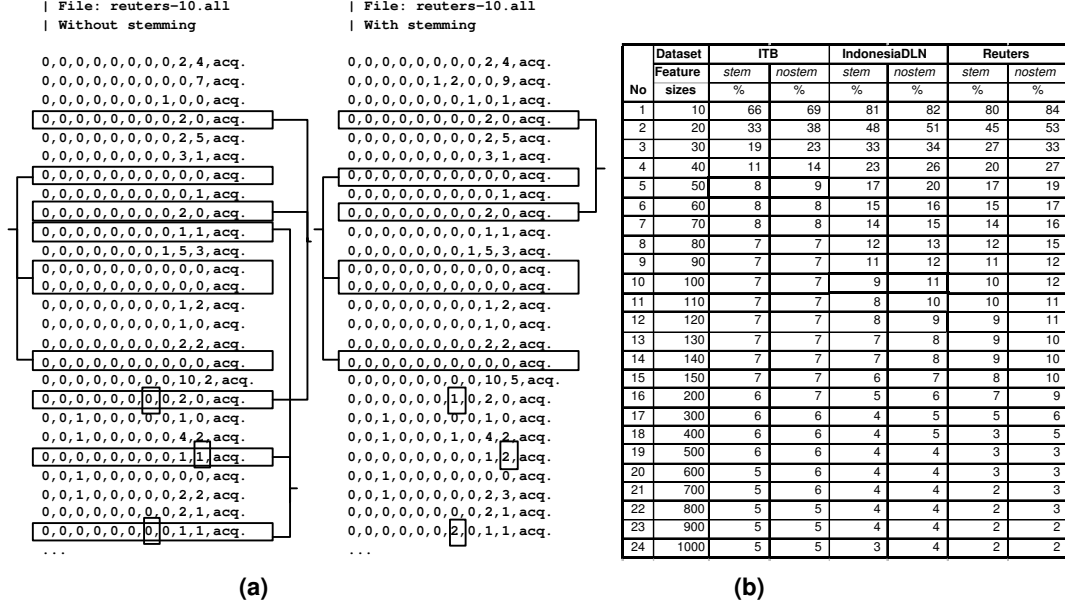| Dataset	ITB		IndonesiaDLN		Reuters	
Feature	stem	nostem	stem	nostem	stem	nostem
No	sizes	%	%	%	%	%	%
1	10	66	69	81	82	80	84
2	20	33	38	48	51	45	53
3	30	19	23	33	34	27	33
4	40	11	14	23	26	20	27
5	50	8	9	17	20	17	19
6	60	8	8	15	16	15	17
7	70	8	8	14	15	14	16
8	80	7	7	12	13	12	15
9	90	7	7	11	12	11	12
10	100	7	7	9	11	10	12
11	110	7	7	8	10	10	11
12	120	7	7	8	9	9	11
13	130	7	7	7	8	9	10
14	140	7	7	7	8	9	10
15	150	7	7	6	7	8	10
16	200	6	7	5	6	7	9
17	300	6	6	4	5	5	6
18	400	6	6	4	5	3	5
19	500	6	6	4	4	3	3
20	600	5	6	4	4	3	3
21	700	5	6	4	4	2	3
22	800	5	5	4	4	2	3
23	900	5	5	4	4	2	2
24	1000	5	5	3	4	2	2

(b)

Figure 3.1: (a) Two subset of 26 instances with 10 features from the Reuters-21578 dataset. These subsets show the effect of stemming in decreasing the number of duplicate instances. (b) Duplicate or conflicting instances in the ITB, IndonesiaDLN, and Reuters-21578 datasets, based on feature sizes. The values represent the total percentage of duplicate instances and conflicting instances. The datasets with stemming tend to have lower percentage of these instances compared with the ones without stemming.

The first subset has 10 duplicate instances, while the second subset has 6 duplicate instances. Two features that contribute to this cut are *note* and *company*. In the stemmed subset, *notes* and *noted* were stemmed into *note*, while *company* and *companies* were stemmed into *compani*. This stemming will increase the frequency of both features and as an effect it may reduce the number of duplicate instances in the stemmed dataset.

Figure 3.1(b) shows the percentage of duplicate or conflicting instances in the datasets with stemming and without stemming, based on the feature sizes. The datasets with stemming tend to have lower duplicate or conflicting instances. On all datasets with 10 features, more than 60% of the instances are duplicate or conflicting instances. We reduce the fraction of duplicate or conflicting instances to under 10% using feature sizes 50, 100, and 120 for the ITB, IndonesiaDLN, and Reuters datasets respectively.

3.2   The Learning Output

The main measurement test was carried using the **Inducer** utility to process the datasets on each classifier. We also wrote several Perl and unix shell scripts to automate the learning processes.

Figure 3.2(a) gives an illustration of IB's learning output on the ITB dataset with 10 features. It shows the number of instances in the training and test sets, number correct and incorrect instances, errors, and the confusion matrix. For further analysis, we only use the **error** scores and the **confusion matrix**. We calculate the macro-averaging for recall and precision measurements from the matrix.

The Unix `time` command is used to capture information about the elapsed time in seconds for each learning process. Figure 3.2(b) shows the `time` command output that describes the required time to execute each process. We only use the **elapsed** time values of the output for further analysis.

3.3   Results and Evaluation

We ran each learning algorithm with various-sized feature sets on each dataset. In this section, we present the test results and evaluation grouped by recall-precision, time performance, errors, and learning curve. All of the learning processes were conducted using AlphaStationXP1000 nodes provided by the RUG's High Performance Computing Center. The processors are alpha processors 21264A, with 667 MHz a 64K/64K data/instruction cache and a 4MB L2 cache.

We created 18 jobs, and each job was ran on a node. 9 jobs ran the Inducer commands for NB, IB, and ID3 on the ITB, IndonesiaDLN, and Reuters-21578 dataset collections (each collection consists of 24 new datasets in different feature sizes). The other 9 jobs ran the LearnCurve commands for the three classifiers on the three datasets (each dataset with 1000 features size).

3.3.1   Recall, Precision, and F_1-Measure

Given the confusion matrices produced by MLC++, we calculate recall, precision, and F_1-measure of the classifiers on the datasets. We got their scores by calculating the decisions on a per-category basis and then averaging over all categories. This produced macro-averaging

(a) The Inducer's output

```
begin: itb-10 ib
Classifying (% done): 10%  20%  30%  ... 80%  90%  100%  done.
Number of training instances: 729
Number of test instances: 403.  Unseen: 128,  seen 275.
Number correct: 70.  Number incorrect: 333
Generalization error: 85.938%.  Memorization error: 81.091%
Error: 82.630% +- 1.890% [78.628% - 86.016%]
Average Normalized Mean Squared Error: 59.653%
Average Normalized Mean Absolute Error: 87.192%


Displaying confusion matrix...
    (a)      (b)      (c)      (d)   ...  <-- classified as
 -------- -------- -------- --------
   14.00     1.00     0.00     0.00 ...  (a): class 1
    0.00     9.00     0.00     0.00 ...  (b): class 10
    0.00     0.00     5.00     0.00 ...  (c): class 11
    5.00     0.00     1.00     0.00 ...  (d): class 12
    ....     ....     ....     ....       ....
end: itb-10 ib
```

(b) The Unix time output for above Inducer's process:

```
0.60user 0.08system 0:08.15elapsed 8%CPU (0avgtext+0avgdata 0maxresident)k
0inputs+0outputs (1757major+535minor)pagefaults 0swaps
```

Figure 3.2: (a) Output of the **Inducer** utility for IB on the ITB dataset with 10 features size. (b) The complete output of a Unix `time` command. The elapsed time is shown by `0:08.15elapsed` (in hours:minutes.seconds).

scores that are dominated by rare categories. The tabular results provided by Table 3.1 concern the IndonesiaDLN dataset.

For overall comparison, Figure 3.3 shows the curves of the F_1-measures for various feature sizes on the ITB, IndonesiaDLN, and Reuters datasets. The system run was able to process up to 1000 features on the ITB dataset but up to 600 features on the IndonesiaDLN dataset. While on the Reuters-21578 dataset, the system run was only able to process maximally 500 features for NB, 700 features for ID3, and 800 features for IB. This difference was caused by the limitation of the application to handle large data files. The application requires a lot of memory and time to process larger files, and on some data files the process was terminated because it was out of memory.

Classifiers	NB			IB			ID3		
Features	R	P	F	R	P	F	R	P	F
10	0.09	0.05	0.06	0.1	0.08	0.09	0.1	0.08	0.09
20	0.15	0.11	0.13	0.16	0.16	0.16	0.17	0.16	0.16
30	0.16	0.13	0.14	0.19	0.18	0.18	0.2	0.2	0.2
40	0.16	0.13	0.14	0.21	0.22	0.21	0.21	0.22	0.21
50	0.17	0.14	0.15	0.24	0.27	0.25	0.24	0.24	0.24
60	0.19	0.18	0.18	0.26	0.29	0.27	0.25	0.26	0.25
70	0.19	0.18	0.18	0.27	0.29	0.28	0.25	0.25	0.25
80	0.19	0.18	0.18	0.28	0.3	0.29	0.26	0.25	0.25
90	0.19	0.18	0.18	0.3	0.33	0.31	0.28	0.27	0.27
100	0.2	0.19	0.19	0.3	0.34	0.32	0.3	0.29	0.29
110	0.2	0.19	0.19	0.29	0.34	0.31	0.3	0.29	0.29
120	0.21	0.19	0.2	0.31	0.35	0.33	0.32	0.32	0.32
130	0.21	0.19	0.2	0.31	0.34	0.32	0.32	0.32	0.32
140	0.21	0.19	0.2	0.31	0.35	0.33	0.33	0.33	0.33
150	0.21	0.19	0.2	0.3	0.35	0.32	0.33	0.33	0.33
200	0.21	0.19	0.2	0.31	0.34	0.32	0.34	0.35	0.34
300	0.25	0.23	0.24	0.34	0.38	0.36	0.36	0.37	0.36
400	0.31	0.29	0.3	0.33	0.34	0.33	0.38	0.4	0.39
500	0.32	0.3	0.31	0.33	0.39	0.36	0.37	0.38	0.37
600	0.33	0.32	0.32	0.32	0.37	0.34	0.37	0.37	0.37

Table 3.1: The macro-averaged results for the recall, precision, and F_1-measure on the IndonesiaDLN datasets. The system was only able to process the datasets with maximally 600 features because of the run time limitation.

On the average, the lowest score of the F_1-measures is on the ITB dataset. This difference is related to the number of instances provided by the ITB dataset which is relatively small (1132 instances) compared to the IndonesiaDLN and the Reuters-21378 datasets that have 3082 and 9100 instances respectively. This reason will be clearly explained by Figure 3.6 in the **Learning Curve** section.

We know from the curves that the score of the F_1-measure on all datasets are relatively low. Even if we compare the score for IB and NB on the Reuters-21578 dataset with the one reported by Yang [10], our F_1-measure score is significantly lower. This fact can be explained by two possible reasons. First, we used the macro-averaging technique to calculate the scores and this allows the rarely used categories to dominate. For example, on the IndonesiaDLN dataset with 10 features, the rare categories with zero precision and recall (totally missclassified) are about 74% of total categories. These categories force the global average to a lower level. Second, possibly because we used a different feature weighting setting and techniques. Yang use log(tf) while we use information gain, and the feature sizes used by Yang is about 1000-2000 features while we only able to process 500-1000 features. For IB, we set the number of neighbors to 10 while Yang used 45. As a consequence, this difference will influence the accuracy of the learning results.
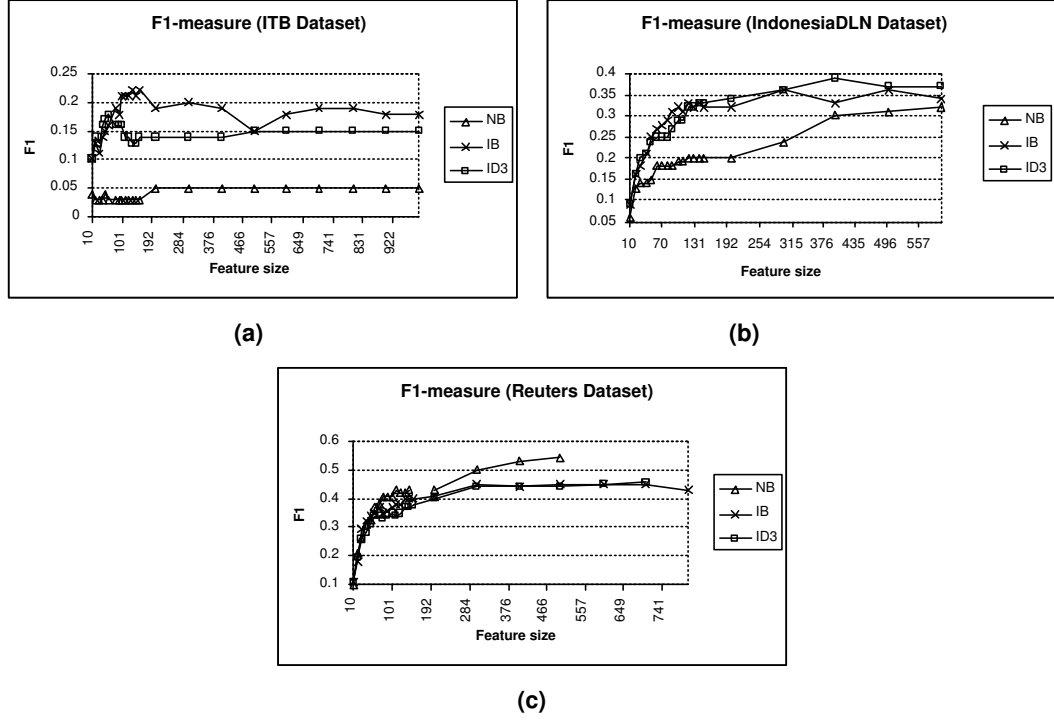
(a)

(b)

(c)

Figure 3.3: The macro-averaged F_1-measure for various feature sizes on three datasets with stemming. The X axis represents the feature size, and the Y axis represents the macro-averaged F_1-measure of a classifier on a given feature size. (a) F_1-measure on the ITB dataset (729 instances). (b) F_1-measure on the IndonesiaDLN dataset (3082 instances). (c) F_1-measure on the Reuters-21578 dataset (9100 instances), where NB was only able to process until 500 features.

NB gets the lowest position among other classifiers on both ITB dataset and IndonesiaDLN datasets, but it gets the highest position on the Reuters-21578 dataset. This result can be used to explain the quality of the IndonesiaDLN classification by its users. We know from the theory as noted by Mitchell [12] that the Bayesian inference, which NB classification is one of its worked example, depends strongly on the prior probabilities. With the probabilities about situation at hand, NB attempts to classify novel situations, and its accuracy relies heavily on the probability assigned to the prior situation. Thus, given the low scores on the ITB and IndonesiaDLN datasets and high scores on the Reuters-21578 dataset, we can conclude that prior probabilities over document classification by IndonesiaDLN users at hand is less reliable as an indicator to classify novel documents compared with the one by Reuters's professionals.

IB performs better than ID3 on the ITB dataset, and comparable on both the Indonesia DLN and the Reuters-21578 datasets. IB with the number of neighbors more than one is intended to form smoothing that can clean up the influence of some data inconsistencies and exceptions, and if this works well, we can conclude that the ITB dataset is noisier than other datasets. From Figure 3.3(a) we also know that the performance of this classifier is decreasing when the feature size is increasing.

The larger number of features doesn't always provide better input data for the classifiers. It can be seen from Figure 3.3 that the macro-averaged F_1-measure for all classifiers will increase until a certain number of features, then will decrease or become constant. For example, Figure 3.3(a) shows that IB decreases after 170 features, ID3 decreases after 60 features, and NB remains constant after 200 features. This phenomenon is referred to as "the curse of dimensionality" [12]. As an illustration of this phenomenon, we can imagine instances described by 200 features and more, but only 50 are relevant to the label. In this case, adding features increases the amount of information available, but also leads to a degraded performance. As an increasing number of parameter are estimated from a fixed amount of data, we induce a model of the noise as well as the true relationship in the training set [37].

3.3.2 Error

We calculate error by averaging all decisions in a single group with no weighting on the basis of category size to produce a result that will be dominated by common categories. This averaging is important because, as shown by Figure 2.5(b), most documents are classified into common categories whose number is less then 20% of the total categories in the IndonesiaDLN dataset.

Figure 3.4 shows the micro-averaged error of the classifiers on all datasets. Figure (a) shows the error curves on the ITB dataset, figure (b) on the IndonesiaDLN dataset, and figure (c) on the Reuters-21578 dataset. The number of features for each curve vary depending on the capability of the system run to process the data files completely.

Given the curves we can see that the "curse of the dimensionality" is clearly felt by IB. On all datasets, the error rate of IB decreased until a certain level of feature size (between 150 and 300 features), and then increased after that level. The large number of features increases the number of irrelevant attributes that dominate the distance between neighbors. As a consequence, the similarity matric used by IB will be misleading. IB and other nearest
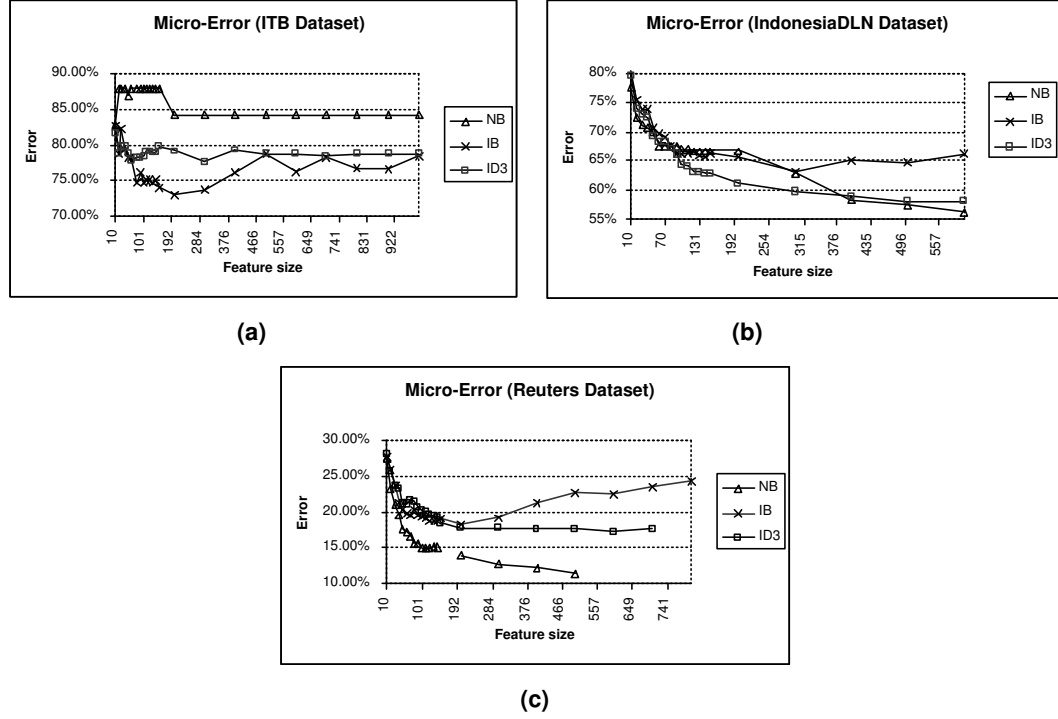
Figure 3.4: Micro-averaged error for various feature sizes on three datasets with stemming. The X axis represents the feature size, and the Y axis represents the percentage of error by a classifier on a given feature size.

neighbor classifiers are sensitive to this problem [12](p.235).

On the ITB dataset, ID3 outperforms others until we use 60 features. After that, its error increases and becomes constant. At this point, IB outperforms ID3 and has the lowest error rate at feature size 200. After that, IB's error rate increases because of the dimensionality problem.

On the IndonesiaDLN dataset, ID3 outperforms others until reaching feature size 400. At this point, NB outperforms ID3 after decreasing the error rate significantly from feature size 200 on. This phenomena could be caused by sparse location of the features. Probably the NB classifier found that features ranked after 200 by information gain contribute to the positive results that in turn will improve its performance.

On the Reuters-21578 dataset, NB outperforms others on all feature sizes. This result confirms the highest performance that is achieved by NB as shown by Figure 3.3(c). ID3 is at the second position with a constant error rate of 17.5% since the feature size 200.

3.3.3 The Effect of Stemming

Beside running tests on the datasets with stemming, we also run tests on the datasets without stemming. Given the resulting confusion matrices, we calculated the macro-averaged F_1-measure on the datasets without stemming. To see the effect of the stemming, we calculated the difference of F_1-measure scores on the datasets with stemming and without stemming, which produced improvement of F_1-measures on each datasets.

The curves of these improvements are shown by Figure 3.5. A curve rising above the X axis shows an improvement of the classifier's performance at that number of features. From this figure we know that for a dataset with certain numbers of features stemming doesn't always improve the performance of the classifier. For example on the IndonesiaDLN dataset, the stemming improves almost all of the classifiers, but at some level of feature sizes, the performance dropped. The performance of ID3 dropped on the feature size 60-80, and IB on the feature size 120+.

We calculate the cumulative improvement for each curve by summing the values above and below the X axis, and show the result in the legend box. On average, the summation shows improvement on all classifiers over all datasets, except on IB (-0.12) and ID3 (-0.01) over the ITB dataset. This results shows that stemming in general will improve the performance.

A better improvement is shown by the Reuters-21578 dataset. The performance of IB and ID3 were improved 0.4 and 0.35 points respectively, while NB also improved but only about 0.04 points. If we compare these improvements with other datasets, the classifiers gain more benefits from the stemming on the Reuters-21578 dataset although not significant. It is understood because the Reuters-21578 dataset is in English, while the ITB and IndonesiaDLN datasets are predominantly in Indonesian with very little in English.

We didn't apply stemming for Indonesian language because of the following reason. The effectiveness of stemming depends on the morphological complexity of the language, and Indonesian language is considered to be morphologically less complex than English because it does not recognize tenses, gender and plural forms. Thus, the impact of stemming on retrieval for Indonesian language is limited [27][38].
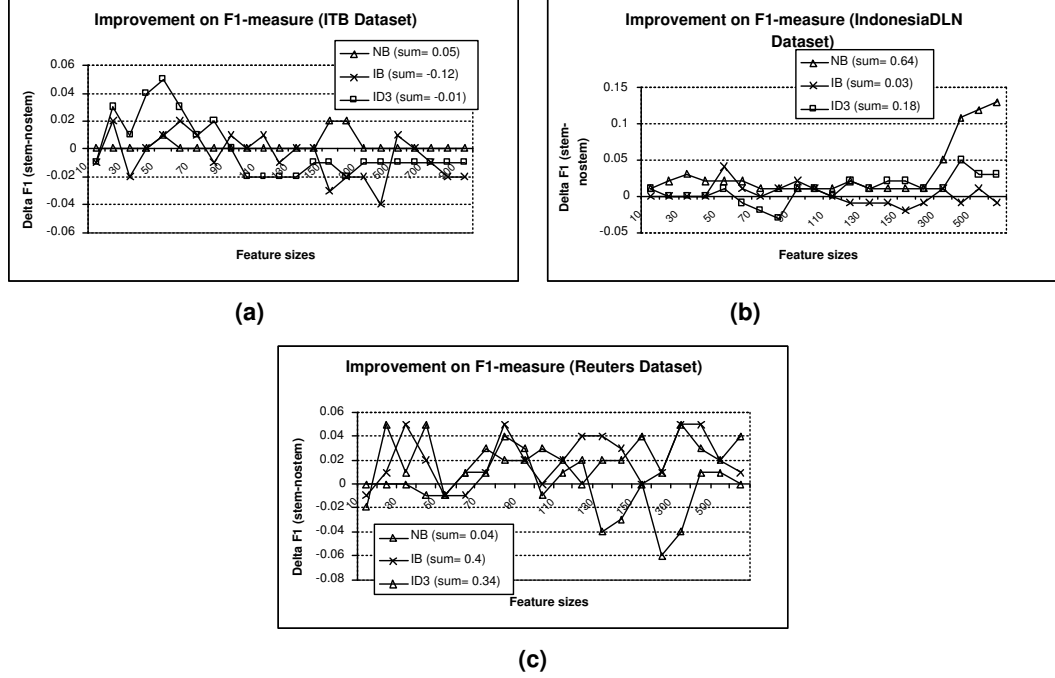
(a)

(b)

(c)

Figure 3.5: The effect of stemming on the F_1-measure over three datasets. The X axis represents the feature size, and the Y axis represents the difference between the F_1-measure scores of a classifier on a datasets with stemming and without stemming on a given feature size. The values in the legend box show the total sum of the differences. A rising curve above the X axis shows an improvement.

3.3.4 Learning Curve

We ran the LearnCurve utility for each classifier on each dataset. From the total of 9 Learn-Curve runs, we were only able to get 7 results, because two runs for NB on the IndonesiaDLN and the Reuters-21578 datasets were terminated due to the maximum execution time limit (72 hours or 3 days). For evaluation, we generate curves that are fitted to the data with two motivations. First, to see the performance of each classifier on the different datasets, and second to see the performance of different classifiers on the same dataset.

Figure 3.6 compares the performance of IB and ID3 over the three datasets. From this figure we know that on different datasets, the classifiers will perform differently, and all classifiers achieve better performance in accordance with the increasing of the number of instances. On the same feature size and number of instances, both classifiers have better performance on the Reuters-21578 dataset, while on the ITB and the IndonesiaDLN datasets
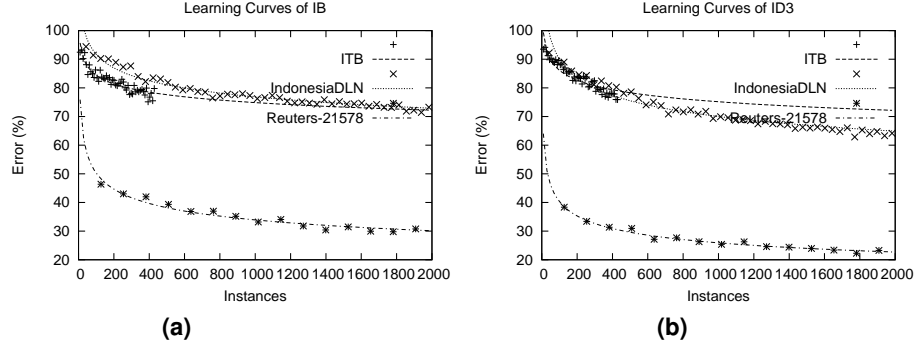
Figure 3.6: Learning curves fit to the data, using $f(x) = a * x^b$ function, of IB and ID3 on three datasets with 1000 features. The X axis represents the number of training instances and the Y axis represents the accuracy when trained on a given number of instances and tested on the unseen instances. The number of instances for this comparison is limited to 2000.

the performances are far lower.

The hierarchical classification model used by the ITB and the IndonesiaDLN which is mapped into a flat classification model for our data representation also could lead to this low result. Similar documents that was classified under similar sub-category but under different root categories could lead to the emergence of conflicting instances as shown by Figure 3.1. For example, a *research report* about "environment" and a *thesis* about the same topic will be classified into different categories by the domain expert.

Since at the beginning of learning until 429 instances, both IB and ID3 perform better on the ITB dataset than on the IndonesiaDLN dataset. IB achieves error rate 79% vs 85% on both datasets respectively, and ID3 achieves 76% vs 80%. These figures explain the results shown by Figure 3.3, where the classifiers achieved lower F_1-measure score on the ITB dataset. Intuitively, the classification on the ITB dataset should be more consistent compared with on the IndonesiaDLN dataset. It should be more consistent because it was done by selected staff at the ITB library while the IndonesiaDLN's classification was done by many people from many organizations that join the digital library network. There is no guarantee that they used the same classification standard and had the same background for the classification technique. Figure 3.6 confirms this assumption, that given the same number of instances and feature size, the classifiers will achieve better performance on the ITB dataset.

Given Figure 3.7, we will evaluate how the different classifiers will perform on the same
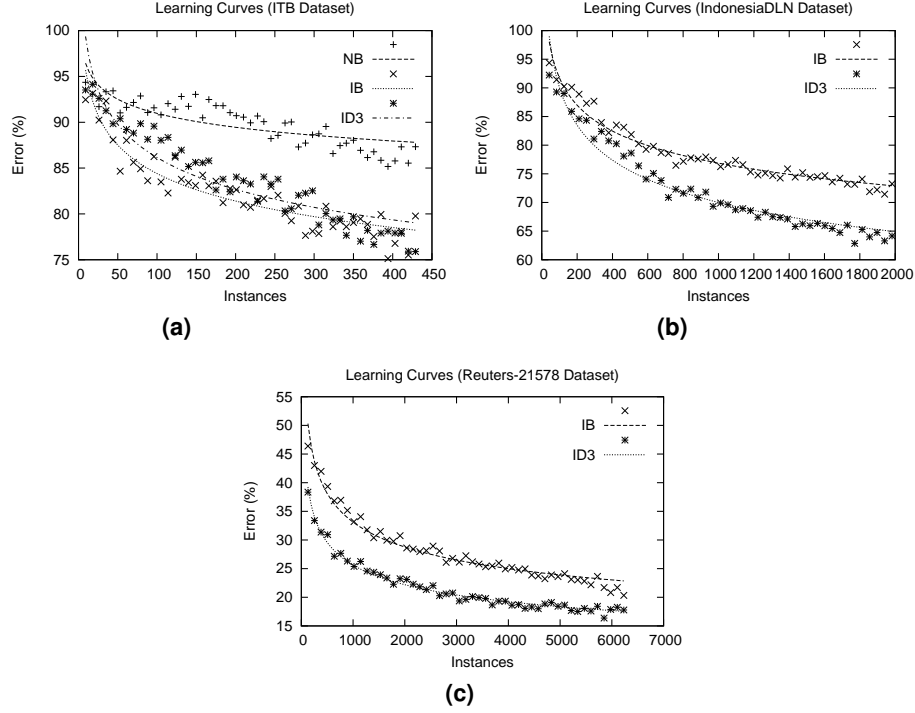
Figure 3.7: Learning curves fit to the data, using $f(x) = a * x^b$ function, of (a) three classifiers on the ITB dataset, (b) IB and ID3 on the IndonesiaDLN dataset, and (c) IB and ID3 on the Reuters-21578 datasets. All datasets use 1000 features. The X axis represents the number of training instances and the Y axis represents the accuracy when trained on a given number of instances and tested on the unseen instances.

dataset. Figure 3.7(a) shows the learning curves of three classifiers on the ITB dataset. We know from this figure that NB significantly underperforms other classifiers. On this dataset, IB slightly outperforms ID3, but on the IndonesiaDLN (b) and on the Reuters datasets (c), ID3 outperforms IB. If we see the inclination of the IB and ID3 curves on the ITB dataset and extrapolate the curves on more instances number, ID3 will show better performance.

Figure 3.7(a) also shows that at the beginning, IB learns faster than other classifiers, but after 300 instances, its error rate become higher than ID3. Figures (b-c) show that ID3 learns faster than IB, and it appears to be the best classifier for this measurements. These results confirmed Moulinier's report [39] where ID3 outperforms IB and NB.
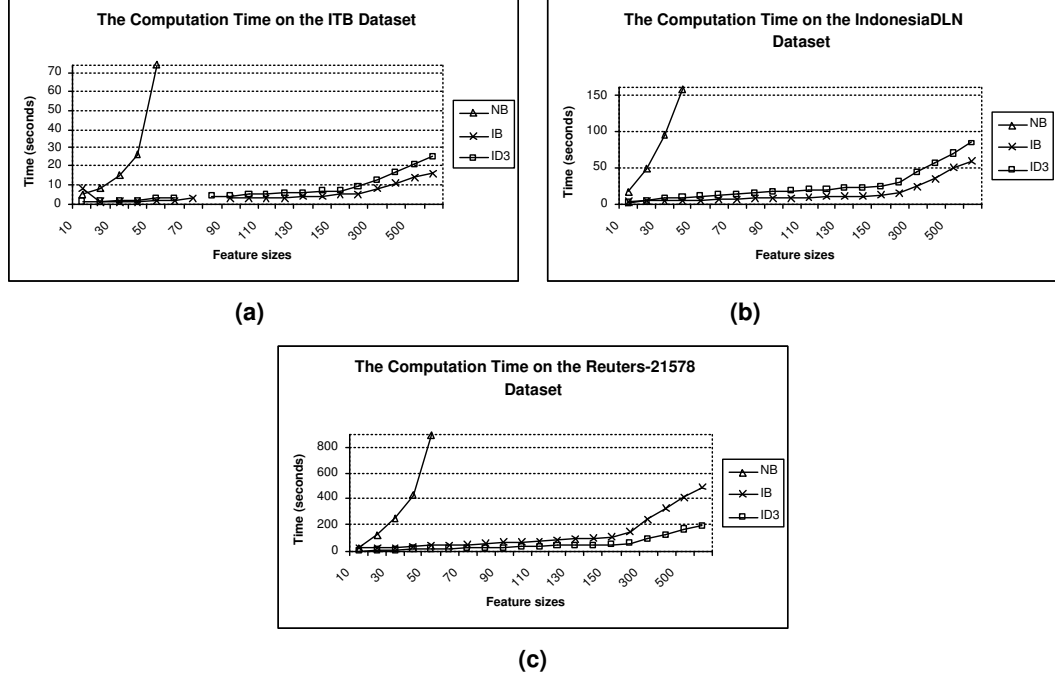
Figure 3.8: The computation time curves of (a) NB, (b) IB, and (c) ID3 on all datasets. The X axis represents the feature sizes, and the Y axis represents the computation time (seconds) by a classifier to learn a dataset on a given feature size.

3.3.5 The Computation Time

We compare the computation time of NB, IB, and ID3 where the computation time data was collected during the precision-recall measurement as shown by Figure 3.3 previously. The datasets varied on their number of instances and feature sizes.

Figure 3.8 shows the computation time curves of the classifiers on the ITB, Indonesi-aDLN, and Reuters-21578 datasets. Given these comparisons, we know that among the classifiers, NB is the worse in term of the computation time required to learn the datasets. It is because according to the equation 2.3.8, the number of distinct $P(a_i|v_j)$ terms that must be estimated by NB from the training data equals to the number of distinct feature values time the number of distinct target values. When new instance is learned, NB also must update its estimation, and that in turn will increase its computation time.

On the ITB and IndonesiaDLN datasets, IB achieved lower computation time compared to ID3. Given a new training data, IB doesn't give the processing result or the explicit

description of target function. It stores the training examples and postpones the generalization, until a new instance is given to be classified. This reduces the computation time significantly during the training. While on the Reuters-21578 dataset, which has more instances for training and classification, ID3 requires lower computation time than IB. On this dataset, IB must pay more computation time to classify more instances than on other datasets.

3.4 Summary

In these experiments, we have measured the performance of NB, IB, and ID3 classifiers on the ITB, IndonesiaDLN, and Reuters-21578 datasets. The following points summarize the results demonstrated by the experiments:

- Duplicate or conflicting instances are easily found on the datasets with small feature sizes. On the datasets with 10 features, more than 60% of the instances are these instances. We can reduce the duplicate or conflicting instances to less then 10% by using between 50 and 150 features.

- A larger feature size doesn't guarantee a better performance, because it could lead to the "curse of dimensionality". This phenomena was clearly felt by IB. Almost on all datasets, its error rate decreased until a certain level of feature size (between 150 and 300 features), and then increased after that level. It is because the large feature size also will increase the number of irrelevant attributes that dominate the distance between neighbors (see Figure 3.4).

- Stemming doesn't always improve the performance of the classifiers, but in general it does. To get the best performance, we should adjust the feature size to a value that will lead to the best performance, and this value could be different for different classifiers (see Figure 3.5).

- The performances of the classifiers on the ITB and the IndonesiaDLN datasets are not satisfactory compared to the one on the Reuters-21578 dataset with the same feature sizes and number of instances (see Figure 3.3 and 3.6). The most possible reason is that the mixing of topics (e.g. *agriculture* and *engineering*) and information types (e.g. *research report* and *journal*) in making up the category hierarchies may lead to the inconsistency in classification.

- In general, ID3 is the winner for this experiment, followed by IB on the second place and then NB (see Figure 3.7).

- NB suffers the highest computation time among other classifiers, especially when the feature size is increased as shown by Figure 3.8. The figure also shows the "laziness" nature of IB, where it requires less computation time on training phase and more computation on the classification phase. On the ITB and IndonesiaDLN datasets, whose test instances are small, IB achieved the shortest computation time, but on the Reuters-21578 with more test instances, it required higher computation time than ID3.

Chapter 4

Conclusions and Future Work

In this thesis, we examined learning algorithms for automatic text classification in DLs. We measured the performance of NB, IB, and ID3 algorithms on the ITB, and the IndonesiaDLN collections and compared the results with the Reuters-21578 collection. This chapter summarizes the results and discusses possible future work on this field.

4.1 Conclusions

The goal of this thesis was to measure the performance of three machine learning algorithms to perform automatic text classification on DL, especially on the ITB and the IndonesiaDLN datasets. Intuitively, the classification on the ITB dataset should be better compared with on the IndonesiaDLN dataset. It is because it was done by selected staffs at the ITB library while the IndonesiaDLN's classification was done by many people from many organizations that join the network. There is no guarantee that they used the same classification standard and had the same background on the classification technique. We want to know how good the performance on the ITB dataset is and the possibility of using its classifiers to classify novel instances in the IndonesiaDLN. By comparing the results with performance on the Reuters-21578 dataset, we want to know whether it is suitable to apply ML techniques for the automatic text classification on the IndonesiaDLN metadata.

The experiment apparently confirmed the assumption, that given the same number of instances and feature size, the performance of IB and ID3 on the ITB dataset is better than on the IndonesiaDLN dataset (figure 3.6). Given 1000 features and 429 instances, IB achieved error rate 79% vs 85% on both datasets respectively, and ID3 achieved 76%

vs 80%. Compared with the result on the Reuters-21578 dataset, performance on both datasets is significantly lower. IB achieved error rate 40% and ID3 achieved 31,5% on the last dataset.

The poor results on the ITB and the IndonesiDLN datasets are possibly caused by the following reasons. First, the classification in both datasets is mainly based on the mixing of information sources, types, and domains which make up the category tree. This would cause the ambiguity in categorizing the documents. Thus, the inconsistency in categorizing documents by many users is very possible. Second, we transformed the hierarchical classification model used by the IndonesiaDLN into a flat classification model for our data representation. It seems that the mapping imperfectly represents the actual classification. Third, we didn't apply stemming for Indonesian, while most of documents in this dataset are in this language. The stemming for English is proven to improve performance on the Reuters dataset (figure 3.5).

Although the performance is not very good, we found several interesting results from this experiment. First, ID3 appears to be the best classifier on the ITB and IndonesiaDLN dataset, followed by IB, and then NB. Second, on large feature sizes, IB faces the curse of dimensionality. The features will increase the number of irrelevant attributes that dominate the distance between neighbors. Third, stemming does not always improve the performance on some conditions, but in general it will give better dataset input to the classifiers that could improve the performance.

As long as the classification on training material is done by many users with different background on classification technique, the automatic text classification in DLs, especially in the IndonesiaDLN, will never achieve high performance. The DLs need to improve their classification quality, such as using a particular classification standard, before we can apply automated text classification using ML techniques. Otherwise, other techniques to improve DL usability should be evaluated that is not based on the existing classification, for example using document clustering [40] or concept mapping [41]. The results achieved could be useful not only for the IndonesiaDLN and the Ganesha Digital Library software, but for other text classification studies as well. One can compare the results with other text classification reports on the Reuters dataset.

4.2 Future Work

After examining three learning algorithms for text classification in the Indonesia Digital Library Network, there is more work that needs to be done.

- Investigate a better way to represent the hierarchical classification used by the IndonesiaDLN. Domain experts build the classification based on the resource types (e.g. research reports, thesis, journal, etc.) that are followed by the subjects of the resources. Applying automatic classification on the basis of the individual resource type should improve the performance of the classifiers. On the same resource type, the category hierarchies should be more consistent.

- Improve the classification method of the IndonesiaDLN by using a common classification standard among its domain experts. Without a good classification at hand, it is difficult to achieve a good performance of the automatic classification on this digital libraries.

Bibliography

[1] Edward A Fox and O. Sornil. Digital libraries. In R. B-Yates and B. R-Neto, editors, *Modern Information Retrieval*, page 415, New York, 1999. Addison Wesley.

[2] OAI. The Open Archives Initiative, 2003. Retrieved September 4, 2003 from the WWW: http://www.openarchives.org.

[3] Mehran Sahami. *Using Machine Learning to Improve Information Access*. Dissertation, Stanford University, Stanford, 1998.

[4] S.E Larson D.T. Hawkins and B. Q. Caton. Information science abstracts: Tracking the literature of information science. part 2: A new taxonomy for information science. *JASIST*, 54(8):771–779, 2003.

[5] Ismail Fahmi. The Indonesian Digital Library Network is born to struggle with the digital divide. *International Information and Library Review*, 34:153–174, 2002.

[6] Ismail Fahmi. The Network of Networks (NeONs). In *The Fourth IndonesiaDLN Meeting, Institut Teknologi Surabaya, Surabaya*, 2003.

[7] Hsinchun Chen. Introduction to the JASIST Special Topic Section on Web Retrieval and Mining: A Machine Learning Perspective. *JASIST*, 54(7):621–623, 2003.

[8] Gerard Salton. *Automatic Text Processing*. Addison Wesley, 1989.

[9] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of DL-00, 5th ACM Conference on Digital Libraries*, pages 195–204, San Antonio, US, 2000. ACM Press, New York, US.

[10] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In Marti A. Hearst, Fredric Gey, and Richard Tong, editors, *Proceedings of SIGIR-99, 22nd ACM International Conference on Research and Development in Information Retrieval*, pages 42–49, Berkeley, US, 1999. ACM Press, New York, US.

[11] Alexander Bergo. Text categorization and prototypes. 2001. Retrieved September 4, 2003 from the WWW: http://www.illc.uva.nl/Publications/ResearchReports/ MoL-2001-08.text.pdf.

[12] Tom M. Mitchell. *Machine Learning*. McGraw Hill, 1997.

[13] Wikipedia. Inductive bias. Retrieved January 13, 2004 from the WWW: http://en2.wikipedia.org/wiki/Inductive_bias.

[14] Yan Liu, Yiming Yang and Jaime Carbonell. Boosting to correct inductive bias in text classification. In *CIKM'02, November 4-9, 2002, McLean, Virginia, USA*, 2002.

[15] Google. Google Search Appliance FAQ. Retrieved August 10, 2003 from the WWW: http://www.google.com/appliance/faq.html#13.

[16] R. Kohavi, G. John, R. Long, D. Manley, and K. Pfleger. MLC++: A machine learning library in C. In *Tech Report, Computer Science Dept, Stanford University*, 1994. Retrieved September 15, 2003 from the WWW: http://www.sgi.com/tech/mlc/docs/intromlc.ps.

[17] Ron Kohavi. *Wrappers for performance enhancement and oblivious decision graphs*. Dissertation, Stanford University, Stanford, 1995.

[18] IndonesiaDLN. The IndonesiaDLN central hub server, 2003. Retrieved September 4, 2003 from the WWW: http://hub.indonesiadln.org.

[19] Raymond J. Mooney. ML-code Machine Learning Archive, 1991. Retrieved January 7, 2004 from the WWW: http://www.cs.utexas.edu/ftp/pub/mooney/ml-code/.

[20] Derek Sleeman. The role of CONSULTANT in helping domain experts use machine learning, 1993. Workshop on fielded applications of ML 1993, University of Massachusetts, Amherst.

[21] D. Mitchie C.C. Taylor and D.J. Spiegalhalter. *Machine Learning, Neural, and Statistical Classification.* Paramount Publishing International, 1994.

[22] Ron Kohavi, Dan Sommerfield, and James Dougherty. Data Mining Using MLC++: A machine learning library in C++. In *Tools with Artificial Intelligence.* IEEE Computer Society Press, 1996. Retrieved August 17, 2003 from the WWW: http://www.sgi.com/Technology/mlc.

[23] David D. Lewis. Reuters-21578 text categorization test collection. 1997. AT&T Research Lab.

[24] Andrew Kachites McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. Retrieved August 10, 2003 from the WWW: http://www.cs.cmu.edu/ mccallum/bow, 1996.

[25] DCMI. The Dublin Core Metadata Initiative, 2000. Retrieved August 17, 2003 from the WWW: URL http://dublincore.org.

[26] R. B-Yates and B. R-Neto. *Modern Information Retrieval.* Addison Wesley, New York, 1999.

[27] Vinsensius Berlian Vega S N. Information retrieval for the Indonesian Language. Thesis, National University of Singapore, 2001.

[28] K. Aas and L. Eikvil. Text categorisation: A survey. 1999. Technical report, Norwegian Computing Center, June 1999.

[29] M. F. Porter. An algorithm for suffixes stripping. *Program*, 14(3):130–137, 1980.

[30] Y. Yang and J. O. Pedersen. A comparative study on feature selection in text categorization. In *Proc. 14th Int. Conf. Machine Learning*, pages 412–420, 1997.

[31] D. Heckerman M. Sahami, S. Dumais and E. Horvitz. A bayesian approach to filtering junk mail. *Proc. AAAI'98 Workshop Learning for Text Categorization*, 1998.

[32] C. J. van RIJSBERGEN B.Sc. Ph.D. M.B.C.S. *Information Retrieval (second edition)*, chapter Evaluation, pages 112–138. 1979.

[33] Yahoo! Yahoo! Dictionary. Retrieved November 4, 2003 from the WWW: http://education.yahoo.com/reference/.

[34] David D. Lewis. Evaluating text categorization. In *Proceeding of the Speech and Natural Language Workshop*, pages 312–318, San Mateo, CA, Feb 1991. Morgan Kaufmann.

[35] Marie Roch. Confusion Matrices. Retrieved November 4, 2003 from the WWW: http://www-rohan.sdsu.edu/ mroch/acoustic/confusionmatrix.html.

[36] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM Computing Surveys*, 34(1):1–47, 2002. Retrieved August 20, 2003 from the WWW: http://faure.iei.pi.cnr.it/~fabrizio/Publications/ACMCS02.pdf.

[37] David D. Lewis and Marc Ringuette. A comparison of two learning algorithms for text categorization. In *Simposium on Document Analysis and IR, ISRI, Las Vegas*, 1994.

[38] Karin Muller Fadillah Tala, Jaap Kamps and Maarten de Rijke. The impac of stemming on information retrieval in Bahasa Indonesia. In *14th Meeting of CLIN, December 19, 2003*. Retrieved January 5, 2004 from the WWW: http://cnts.uia.ac.be/clin2003/abstracts/tala.html.

[39] I. Moulinier. A framework for comparing text categorization approaches. In *AAAI Spring Symposium on Machine Learning and Information Access, Stanford University, March*, 1996.

[40] Christoper R. Palmer et all. Demonstration of hierarchical document clustering of DL retrieval results. In *ACM/IEEE Joint Conference on DLs*, page 451, 2001.

[41] B. R. Gaines and M. L. G. Shaw. Web Map: Concept Mapping on the Web. In *World Wide Web Journal*, pages 171–184, 1995.