AUTOMATIC AMHARIC TEXT CATEGORIZATION

BY
Yohannes Afework Worku

A thesis submitted to the School of Graduate Studies of Addis Ababa University
in partial fulfillment of the requirements for the Degree of Master of Science in Computer Science

March 2007
Addis Ababa

ADDIS ABABA UNIVERSITY
SCHOOL OF GRADUATE STUDIES
FACULTY OF INFORMATICS
DEPARTMENT OF COMPUTER SCIENCE

AUTOMATIC AMHARIC TEXT CATIGORIZATION

BY
YOHANNES AFEWORK WORKU

Name and Signature of the Board of members of the examiners' board

1. Dr. Mulugeta Libsie (Advisor) _____
2. _____ _____
3. _____ _____

ACKNOWLEDGEMENT

First I would like to thank my family for their understanding and their bearing of the additional burden due to my engagement in this study.

I am also deeply grateful for my advisor Dr Mulugeta Libsie: for making it possible that I do my thesis work on data mining as well as for his encouragements and constructive suggestions.

I would like to express my appreciation to the Ethiopian News Agency (ENA) for letting me use their news data. Especial thanks goes to the IT Department of ENA and particularly to Ato Woundimagen Mekuria, for the untiring support they provided me during data acquisition and data cleaning.

Finally, I like to thank my employer, the General Education Quality Assurance and Examination Agency, for understanding the extra time and effort required by this study.

Yohannes Afework Worku

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Rapid developments in Information and Communication Technology are making available huge amount of data and information. Much of these data is in electronics forms (like the more than billion documents in the World Wide Web). Usually these data do not have a standard structure like that of the relational database. Much of the data are unstructured or semi-structured and can generally be considered as a text database.

Text databases are showing accelerated growth throughout the world. As the result, there is an active field of study in text mining to facilitate the extraction of useful and relevant information from text databases.

The text data in local languages is also increasing fast, requiring text-processing tools for text documents to be available in local languages. This is true for Amharic also, as can be surmised from the recent boom of online newspapers, magazines, data in electronics storage, etc.

To facilitate the retrieval of useful and relevant information from Amharic documents, a number of researches on automatic processing of Amharic text have recently been conducted. This research work in Automatic Amharic Text Categorization is an effort to contribute in this direction.

Automatic classification of text data requires that documents are represented by feature words. Representing a document by relevant feature words is an important pre-processing step for automatic classification; it often determines the efficiency and accuracy of the classification. Standard pre-processing tools and methods are therefore very important for automatic classification.

Because of the lack of standard in the Amharic writing system and unavailability of Amharic text processing tools, the focus of the research was on developing a document-pre-processing scheme which facilitates for an efficient automatic classification of Amharic documents.

To this end much attention was given to the processing of the source data by developing and enhancing the following tools. The tools are specific to the source data – Amharic news documents from ENA.

- A tool to correct word spelling variations. Focusing on spelling variation due to pronunciation differences.
- Enhancement to the suffix and prefix removal tool developed in a previous study, so that it can perform semantic analysis before stripping-off affixes from words.
- A tool to correct word variations due to gender marker suffixes.
- A tool to correct word variations due to number marker suffixes.
- A tool to merge compound words (when they may result in semantic loss if separated) written as separate words.

The use of these tools (which enabled 10 to 30 % feature reduction) in addition to other tools and data reduction methods helped to analyze the huge source data (69,684 news items after data cleaning) and measure classifier performances.

Because of the high dimensionality of the source data, classifier algorithms that are suitable for high-dimensional data, Decision Tree and Support Vector Machine (SVM) classifiers were selected for the

research experiment. The open source Weka package is used for the automatic classification of the preprocessed data. Out of the many classifier algorithms available in Weka, the Logic Model Tree (LMT) and the Library of SVM (LibSVM) classifiers were used for performance testing.

Both LMT and LibSVM classifier showed good classification accuracy correctly classifying 79.72% and 81.15% of the test instance into the 15 news categories, respectively. However, the computational cost of the automatic classification was very high - taking several hours in high capacity computers (Computers with 512 MB RAM and 3.7 GHz speed).

The classification performance measures indicate the need for additional works in developing tools and methods for mining Amharic data.

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND

Technological advances are making available huge amount of digital data and information. The rapid growth in stored data has generated a need for new techniques and automated tools that assist in transforming the vast amounts of data into useful information and knowledge [1].

Specifically, text databases are growing very fast due to the increasing amount of information available in electronic forms, such as electronic publications, e-mail, and the World Wide Web. Text databases use words to describe objects. The word descriptions are usually not simple keywords but rather long sentences or paragraphs such as product specifications, summary reports, notes and other documents. Text databases may be highly unstructured (such as some web pages), some text databases may be semi-structured (such as e-mail messages), while others are relatively well structured (such as library databases).

Most text data are semi-structured at best, which make the traditional information retrieval methods inadequate for efficient information use. Typically, only a small fraction of the many available documents will be relevant for a given user. Users therefore need text-mining tools to extract the information they need from such documents [1].

Researchers are looking for better and better ways of using the fast growing huge amount of text data. Text mining is an active research area and people are looking for tools and techniques to automatically analyze, categorize, summarize, and discover trends in text data [1].

Text mining enables to extract useful or interesting patterns from a large set of documents. There are a number of important tasks in text mining such as clustering, categorization, and summarization [2]. This research focuses on text categorization, which deals with the problem of assigning predefined labels or categories to a text document.

Text categorization is a process involving the assignment of predefined categories to free (unlabeled) text data. With the rapid growth of online data and information, text categorization has become one of the key techniques for processing and organizing text documents. Text categorization techniques are used to classify new stories, to find interesting information on the World Wide Web and to guide a user's search through hypertext. Automatic text classification is a supervised learning problem involving inducing knowledge to classifiers from examples [3].

A growing number of classifiers have been applied for automatic classification of text documents including:

- **Support Vector Machine (SVM):** a universal learner, which can be used to classify both linear and nonlinear data. It transforms the original data in a higher dimension, from where it can find a hyperplane for separation of the data using essential training instances called **support vectors.**

- **Decision tree (DTree):** used to select informative words based on an information gain criterion, and predict categories of each document according to the occurrence of the word combination in the document.
- **Neural network (NNet):** uses a perception approach (a separate neural network per category to learn a non-linear mapping from input words) or a three-layered neural network approach (a network containing two hidden layers between the input and output layers).
- **Naive Bayes:** uses joint probabilities of words and categories to estimate the probabilities of categories given a document [1].

Classifiers in general and specially those used for the experiment in this research will be discussed in detail in the next chapter where tools and techniques of classification are detailed.

Text data classification[1] is a two-step process. The first step is a learning step where a model is constructed (a classifier is built) by analyzing a training dataset. Each document in the training dataset is randomly selected from the collection and has a class label indicating to which category it belongs.

In the second step the model is used for classification. First the prediction accuracy of the model (or classifier) is estimated by selecting a test set of class-labeled samples that are independent from the training dataset. If the training set is used to measure the accuracy of the classifier, the estimate would likely be optimistic, because the classifier tends to **overfit** the data (i.e., during learning, it may incorporate some particular anomalies of the training data

[1] Categorization and classification are used interchangeably in this thesis.

that are not present in the general dataset overall). If the estimation accuracy of the classifier is considered acceptable, the classifier can be used to classify future text documents for which the class label is not known [1].

The first step of classification involves telling to the model the class of each document in the training set and is therefore known as **supervised learning**. Supervised learning is the main factor differentiating classification from **clustering**. Clustering is unsupervised learning where the class label of each training sample is not known and the number or set of classes to be learned may not be known in advance [1].

The construction and use of a model to assess the class of an unlabeled sample or to assess the value or value range of an attribute that a given sample is likely to have is generally known as **prediction**. The use of prediction to predict class labels can be defined as classification. Some of the applications of classification and prediction include credit approval, medical diagnosis, and selective marketing.

1.2 STATEMENT OF THE PROBLEM

Rapid developments in information and communication technology (ICT) are impacting the way people create, process, store, access and transfer data and information. The worldwide growth in ICT is having a major effect in the information access and usage habit of people in developing countries. Internet usage, for example, has reached 1,076,203,987 as of December 2006 showing 198.1% worldwide growth in a year, with the highest growth of 625.8% for Africa [4].

The recent growth of ICT infrastructure in Ethiopia is resulting in an exponential increase of digital information being produced [5]. The continual increase of locally available data will increase the demand for tools that simplify the extraction of relevant information.

Developments and application of text processing tools and methods have so far been focused on English and to some extent on European and East Asian languages [5]. Much work has to be done to develop processing techniques and tools for the languages of developing countries like Amharic.

Amharic is one of the major African languages. It is the working language of the Federal Government of Ethiopia and widely spoken throughout the country. Amharic is an Afro-Asiatic language belonging to the Southwest Semitic group with its own unique alphabet [6]. Huge and increasing volumes of electronic data are available in Amharic, which is observed on the growing online newspapers, websites, and digital storages in the language.

It may not be long before users are faced with large volumes of Amharic text in the web and other media. In such scenarios, users will find it very difficult and frustrating to make efficient use of Amharic data unless they are aided by data processing tools for activities like searching, categorization and summarization of these documents.

The task of developing tools for easy retrieval of relevant information is specially challenging for Amharic because there are only few recent and un-coordinated efforts of automation and language processing works at hand: Amharic Text Classification [7, 8], Information Retrieval [9, 10, 11], Amharic Word Parser [12], Amharic Stemmer

(Quoting [12]), and others. Some of these research works will be discussed in later chapters where the Amharic writing system is discussed.

This research focuses on the problem of automatic categorization of Amharic texts. Since automatic text classification involves preparation of the source data before feeding it to a classifier, the work on automatic Amharic text classification involves:

i. Developing tools or using existing tools for
   - cleaning the source data and selecting text representative words (attributes)
   - attribute relevance analysis – to identify important attributes of documents for the classification task
   - data transformation such as weighting – to be applied to the level required by the classification task and the size of the preprocessed dataset

ii. Developing a prototype or applying off-the-shelf tools for automatic classification of the preprocessed Amharic text

This thesis work therefore focuses on the above two steps of classification and is an effort to contribute to the development of processing tools for Amharic text and finding an optimal Amharic text classifier.

1.3 OBJECTIVES OF THE STUDY

1.3.1 General Objective

The major objective of this thesis is to develop and adopt processing tools for Amharic text classification and evaluate the performance of selected classifiers for Amharic text classification tasks.

1.3.2 Specific Objectives

To realize the aforementioned general objective, the study aims to carryout the following tasks.

- Review literature on the techniques of automatic classification of text data
- Study the Amharic writing system and its computer representations
- Study existing tools for Amharic data processing
- Develop and adopt tools of processing Amharic documents for classification purpose
- Apply selected classifiers on a processed Amharic corpus
- Evaluate the performance of the applied classifiers
- Compare the performances of the applied classifiers

1.4 METHODOLOGY

The following methods were employed to achieve the above stated objectives.

1.4.1 Literature Review

Extensive study of available literature has been carried out on

- The methods of effective information access techniques in general and automatic document classification in particular
- Classifier algorithms and their application for classification
- The basic features of the Amharic script and the existing computer representation of Amharic characters
- Developed tools and techniques for Amharic data and information processing in general and Amharic document classification in particular.

1.4.2 Data Processing

The data source for the study is a nearly seven year's collection of Amharic news articles from the Ethiopian News Agency (ENA), which has more than 100,000 news items written using the Amharic Software Visual Geez Version 2.0 (VG2).

The ENA news items were stored in SQL Server. The source data were then cleaned using manual inspection and automatic methods. Automated data cleaning and other pre-processing tasks were carried out using the Visual Basic Programming language.

In pre-processing the news documents, it is assumed that a document is fully represented by the words in it. Furthermore, it is

assumed that the order of the words does not matter (document representation will be discussed in detail in Chapter 2).

The data pre-processing carried out in this research involved developing and adopting tools for

- Data cleaning which involves removal of repeated news items, manual classification of unclassified news items, removal of entry errors, etc.
- Identifying and removing stop words and word affixes
- Correcting for commonly missing letters in VG2 which sometimes occur during data conversion
- Normalizing the different letters of the Amharic script that have the same sound
- Correcting major spelling variations in words focusing in transliteration problems
- Analyzing compound words to correct for inconsistent usage of the compound words (the use of compound words sometimes as a single-word and sometimes as two or more words) as well as to give consideration for the semantics of the compound words
- Selection of relevant attributes (features) of documents

Moreover the processed Amharic documents were collected in their pre-defined categories and the whole data changed to **Arff** (**attribute reference file format**) file format, which is suitable for the **Weka** open source application package used for the automatic classification.

1.4.3 Experiment

Automatic classification involves identification of keywords called *feature words or attributes* for representing documents. Huge text datasets can have millions of words representing the document collection in the dataset. However, not all words in the dataset are useful for automatic classification. Classification systems use **stop list** to avoid stop words from document attributes. **Stop words** - like *a, the, of,* and *for* - are considered irrelevant for classification [1]. Moreover, some attributes may be irrelevant for a given classification task. For example, if the task is to classify news items to the major news categories, attributes such as the news creation date are likely to be irrelevant.

Automatic classifications often employ different methods of attribute selection. Observation of the preprocessed data for this research indicates that content-bearing words usually occur more than once in a document. Moreover, it is observed that words that are found in only one document in a category are usually non content-bearing words or anomalies created due to entry errors. In this research a word is selected as a document representative if it is not in the stop list, if it occurs more than once in a given document, and if it appears in more than one document in a category. All words of every category satisfying the above criteria are collected in the experiment dataset.

After pre-processing, the experimental dataset is rearranged to the format suitable for the Weka package that is used for the automatic classification. Weka requires the input data to be rearranged in an **attribute weight matrix** before it is converted to **Arff** file format.

The preprocessed dataset is rearranged in an attribute weight matrix by

- considering the whole dataset as a relation with word attributes
- considering the attributes in the dataset as fields (column) of the relation
- taking each document as a separate record or instance (row) of the relation
- using the weight of attributes in a document as the value of the fields for the instance the document represents
- considering class labels as nominal attributes in the dataset

After the arrangement, the experimental dataset is converted to **Arff** file format, which is suitable for applying classifier algorithms provided by Weka.

The Weka package provides several classifiers for automatic classification of the preprocessed dataset. The package also has a tool for comparing the performances of the different classifiers used in the classification.

1.5 EXPECTED CONTRIBUTION

This study is expected to provide

- new tools for processing Amharic text data
- enhancement to some of the existing Amharic text processing tools
- empirical evidence of the effectiveness of selected classifier algorithms for automatic classification of Amharic documents

Although it is conducted on Amharic text corpus, the results of this study could easily be applied for any document collection in local languages that use the Ethiopic script (Like Gurage, Harari, Tigrie, and Tigrniya)

Generally apart from being an academic exercise to identify efficient classifier algorithm for Amharic text corpus, the study is expected to contribute to the active research area in the development of applications that manipulate documents written in the Ethiopic script.

1.6 ORGANIZATION OF THE THESIS

The thesis is organized in five chapters. The first chapter gives the overview of the research, with the research problem statement, objectives and methodology.

In chapter two detail descriptions of major tools and techniques for automatic text categorization is given.

Chapter three discusses the Amharic writing system with emphasis on the representation and processing of Amharic texts in electronic format.

Chapter four presents the processing tools used and the experiment carried out for classification of Amharic news documents.

Finally the research findings and recommendations are presented in chapter five.

CHAPTER TWO

TEXT CLASSIFICAION

2.1 INTRODUCTION

Human capabilities of both generating and collecting data have been increasing rapidly. Factors that contribute to the accelerated growth include the computerization of businesses, scientific and government transactions as well as advances in data collection tools ranging from scanned text and image platforms to satellite remote sensing systems. In addition, popular use of the World Wide Web as a global information system has made access to huge amount of data and information possible [1].

The rapid growth in stored and transient data led to a great deal of interest in developing useful and efficient tools and software to assist users in finding relevant information. Text classification, which is a powerful technique for automating assignment of documents to categories, has been proved to be useful in helping organize and search text information on text data sources [13].

2.2 TEXT CLASSIFICATION METHODS

Depending on the context of their application different approaches to text classification - ranging from **manual** category assignments to **rule-based** approaches, and to **automatic** classification - are employed.

Manual classification involves the use of trained professionals for classification. The method is usually used in category structures that are very general, consistent across individuals, and relatively static. Examples of such classification approaches include Dewey Decimal or Library of Congress Classification systems, Medical Subject Headings (MeSH), or Yahoo's topic hierarchy [13]. Manual classification has limited applicability because it is very time-consuming and costly.

Rule-based approaches (expert systems), similar to CONSTRUE developed by the Carnegie Group, use manually developed domain specific or application specific rules. They make binary decisions about category membership and are typically difficult to modify [13]. Moreover such procedures are prone to biases and errors, and are extremely time-consuming and costly [1].

Automatic classification is another approach, which uses inductive learning (machine learning) techniques to automatically construct classifiers using labeled training data. The first step in automatic text classification is identification of the attributes of a document, i.e., selection of words (features) of a document that can adequately represent the document [13].

Text classification poses many challenges for indicative learning methods since there can be millions of word features. The resulting classifiers, however, have many advantages because:

- they are easy to construct and update
- they depend only on information that is easy for people to provide, information like examples of items that are in or out of categories

- they can be customized to specific categories of interest to individuals
- they allow users to smoothly trade-off precision and recall depending on their task [13]

Because of their ease of use and impressive results a growing number of statistical classification and machine learning techniques have been applied to text categorization including Support Vector Machines (SVM) [3], Decision tree [14, 15], Neural Network [16, 17], and Naive Bayes [14, 15].

The text categorization approach used in this thesis is automatic classification supported by limited Natural Language Processing (NLP) techniques. Automatic text classification is therefore discussed in detail in the remaining sections of this chapter.

2.3 USES OF AUTOMATIC CLASSIFICATION

Many information organization and management tasks make use of text categorization which is the assignment of natural language texts to one or more predefined categories based on their content [13].

Text categorization has widely been used for automatically assigning subject categories to documents to support

- **text routing:** for instance organizing documents by automatic filing of news stories under the sections of Sport, Health, Politics, etc.
- **text filtering:** like automatically blocking news stories that are not appropriate for children or classifying documents as relevant and irrelevant, etc.

Automatic classification can be used for information management tasks that are more dynamic and personal like

- **real-time sorting of e-mails**, i.e., using an e-mail filter to discard junk mail and further classify the useful mail into folder hierarchies corresponding to categories that are relevant to the user
- **topic identification** to support topic specific processing operations: Like using text categorization to automatically classify web pages under popular hierarchical catalogues.

2.4 STEPS IN AUTOMATIC CLASSIFICATION

Automatic text classification is a two-step process involving preparing the text data for classification (pre-processing) and feeding the processed data to a classifier.

2.4.1 Preparing the Data for Classification

Preprocessing is important to improve the accuracy, efficiency, and scalability of the classification process. It is the first step in the preparation of documents to present them in a format suitable for classification.

In general the following three pre-processing steps are applied: **Data cleaning**[2], **relevance analysis,** and **data transformation and reduction** [1]. Details of each step are given in the following sections.

[2] Data, text data, and text documents are used interchangeably in this thesis

2.4.1.1 Data Cleaning

Real world data are usually incomplete, noisy and inconsistent. Data cleaning is an attempt to fill *the missing values*, smooth out *noise* and correct inconsistencies [1].

In the context of text classification, missing values of major significance are missing class labels. If the class label of a document (in the training set) is missing the whole document has to be scanned and the appropriate class value assigned manually. In this thesis the source data is scanned for news items with missing category labels and when such news items are found, the missing class labels are manually assigned (whenever it is applicable).

When manual assignment is not feasible (especially for attributes other than the class value) a popular data cleaning strategy dealing with missing values is using the most probable value to fill in the missing value. The most probable value of an attribute may be predicted from the other non-missing attribute values with inference-based tools using a Bayesian formalism, or decision tree induction [1].

Noise is a random error or variance in a measured value. For text classification errors in assigning class labels (misclassification) need to be identified (for example, by inspection of misclassified documents during experimentation) and corrected before training a classifier for deployment [1].

2.4.1.2 Relevance Analysis

Many of the attributes of text data may be *redundant*. There may also be *irrelevant* attributes. Relevance analysis helps to reduce these redundant and irrelevant attributes from the dataset.

a) Document Representation

For the purpose of automatic classification, a document can be considered as a collection of key words. The key words are often called features or attributes of the document. Not all words in a document are considered as features.

The desire to have algorithms that can improve classification efficiency while maintaining accuracy is driving the attention for dimension (feature) reduction techniques. Dimension reduction techniques can generally be classified into **feature extraction** and **feature selection** approaches.

The traditional feature extraction algorithms reduce the dimension of data by linear algebra transformation (such as Principal Component Analysis (PCA)). On the other hand, feature selection algorithms reduce the dimension of data by select features from the original words of a document. Though the feature extraction algorithms have proved to be very effective for dimension reduction, the high dimension of datasets in the text documents often fails many feature extraction algorithms due to their high computational cost. Thus feature selection algorithms are more popular for real life text data dimension reduction problems [18].

During pre-processing one has to select a subset of the words in a document that can adequately represent the whole document. Feature selection is often employed for finding representative subset words for a document.

Feature selection often involves using a **stop list** to avoid words that are useless for classification (**stop words**). Feature

selection may also involve **stemming**, i.e., grouping words that share the same word stem [1].

b) Feature Selection

In general correlation analysis can be used to identify whether any two attributes are statistically related so as to remove one of the two if they are related [1].

Specifically for text classification the text document is analyzed to find efficient representative features for subsequent processing. Redundant attributes can be identified and removed as a result of document analysis. Since word features are used to represent documents, variations in word forms are the major sources of redundancy.

Stemming: Natural language texts are characterized by variations in word forms. The most common ways of creating word variant are suffixing and prefixing. In general, word variants may be caused by factors including grammar requirements, national or local usage, transliteration, abbreviation, and spelling errors. Stemming might be used to normalize word variants by removing affixes through identification of word-stems from full words [19].

In this thesis, tools for removal of common prefixes and suffixes, tools for correction variations due to transliteration, tools for correcting common spelling variation, and tools for normalizing different forms of words are adapted and developed - since there is no available Amharic stemmer.

Stop Word Removal: In case of irrelevant attributes in the dataset, attribute subset selection can be used to find a reduced

set of attributes while keeping the original data class distribution as much as possible [1].

After a document is processed and its features identified, different techniques are used to select the features that adequately represent the document for the purpose of text classification.

Removal of *stop words* is one method of feature selection. Stop words are sometimes defined as function words. Function words have important role in grammar but carry little meaning, and therefore do not contribute much to categorization [20]. In addition of the function words like "a", "the", "of", "and", "is", and "not", domain specific (news specific) stop words are removed from the dataset in this research.

c) Document Modeling

Different ways can be used to model a document in order to facilitate categorization. Considering the most popular approach – the vector space model - a set of d documents and a set of t terms[3], one can model each document as a vector v in t dimensional space R^t, which is why this model is called the **vector-space model.**

Let the **term frequency** be the number of occurrences of term t in the document d, which is $\text{freq}(d,t)$.

The (weighted) **term-frequency matrix** $TF(d,t)$ measures the association of a term t with respect to the given document d. It is generally defined as 0 if the document does not contain the

[3] terms, features, key words are used interchangeably

term, nonzero otherwise. There are many ways to define the term-weighting for nonzero entries in such a vector. For example the Cornell SMART system uses the following formula to compute the normalized term frequency [1].

$$TF(d,t) = \begin{cases} 0 & \text{if } freq(d,t) = 0 \\ 1 + \log(1 + \log(freq(d,t))) & \text{otherwise} \end{cases} \quad \ldots (2.1)$$

Thus relevance analysis can be used to detect attributes that do not contribute to the classification rather than slowing and possibly misleading the learning step.

2.4.1.3 Data Transformation and Reduction

The text data may be transformed by normalization, which involves scaling all values for a given attribute so that they fall within a small specified range such as 0 to 1.

In the vector-space model, in addition to the frequency measure, the importance of a term t is measured by a scaling factor called **inverse document frequency** (IDF). If a term t occurs in many documents, its importance will be scaled down due to its reduced discriminative power. According to the Cornell SMART system, $IDF(t)$ is defined by the following relation [1]

$$IDF(t) = \log \frac{1 + |d|}{|d_t|} \quad \ldots (2.2)$$

where d is the document collection (No, of documents in the collection), and

d_t is the set of documents containing term t

In a complete vector-space model, TF and IDF are combined together, forming the $TF-IDF$ measure

$$TF-IDF(d,t)=TF(d,t)\times IDF(t) \qquad \ldots (2.3)$$

$TF-IDF$ is used to calculate the normalized value of terms (features) of a document in a vector-space model.

2.4.2 Document Classification

Classification is a data analysis task where a model of classifier is built to predict categorical labels such as "Health" or "Education" or "Sport" for news documents. As described in Chapter 1, document classification involves two steps - learning and prediction.

2.4.2.1 The Training Process

In the learning step (training phase) a classifier is built describing a predetermined set of data classes or concepts. Here a classifier algorithm builds the classifier by learning from a **training set** made up of instances (documents) and their associated class labels.

An instance X of a training set is represented by an n-dimensional feature vector $X=(x_1,x_2,...,x_n)$, depicting n measurements made on an instance from n attributes, respectively, $A_1,A_2,...,A_n$ [1].

Each instance, X, is assumed to belong to a predefined class as determined by another dataset attribute called the **class label attribute**. The class label attribute is discrete-valued and unordered. It is categorical because each value serves as a category or class. The **training set** is selected from the dataset under analysis and is made up of individual training instances.

2.4.2.2 The Prediction Process

The learning step can also be viewed as the learning of a mapping or function $y = f(x)$, that can predict the associated class label y of a given instance X.

In order to avoid *overfitting* (i.e. to avoid incorporating particular characteristics of the training data that do not represent the whole dataset) a **test set** is used for the prediction. The test set is made up of test instances that are randomly selected from the general dataset which does not include the training set.

The accuracy of a classifier on a given test set is the percentage of test set instances that are correctly classified by the classifier. The accuracy is calculated by comparing the associated class label of each test instance with the learned classifier's class prediction for that instance.

2.5 CLASSIFIER ALGORITHMS

2.5.1 Introduction to Classification Modeling

In general, predictive modeling is used to predict the unknown value of a variable of interest given known values of other variables. Examples include providing a diagnosis for a medical patient on the basis of a set of test results, estimating the probability that customers will buy product A given a list of other products they have purchased, or predicting the class label of a new news item from the set of feature words making the news[1].

In classification the goal is to learn a mapping from a vector of measurements X to a categorical variable (class label) Y. The variable to be predicted is called class variable and it takes the

values in the set $\{y_1, y_2, ..., y_m\}$. The observed or measured variables $\{x_1, x_2, ..., x_n\}$ are referred as features or attributes. Therefore X is referred as an n-dimensional vector (i.e. taken to be comprised of n variables), where each component can be real-valued, ordinal, categorical, and so forth [21].

In general, for classification the training data consists of *pairs* of measurements, each consisting of a vector $x(i)$ with corresponding class values $y(i)$, $1 \le i \le n$. Thus the goal of classification is to estimate (from the training data) a mapping or a function $y = f(x, \theta)$ that can predict a value y, given an input vector of measured values x and a set of estimated parameters θ for the model f.

where f is the functional form of the model structure

θ_s are the unknown parameters within f whose values have to be determined by minimizing a suitable score function on the data, and the process of searching for the best θ values is the basics for the classifier algorithm [21].

2.5.1.1 Framework of Classification Modeling

Classification problems can be generalized into two different but related views that help explain the logic in the building of the different classifier models: Decision boundary (discriminative) viewpoint and probabilistic viewpoint [21].

Discriminative Classification and Decision boundaries: In this framework a classification model $f(x, \theta)$ takes as input the measurements in the vector X and produces as an output a class label from the set $\{y_1, y_2, ..., y_m\}$ [21].

For example, consider the nature of the mapping function for a simple problem with just two real-valued input variables x_1 and x_2. The mapping produces a piecewise constant surface over the (x_1, x_2) plane in regions where the plane takes class values like, y_1. The union of all such regions where y_1 is predicted is the *decision region* for class y_1.

Knowing where the decision regions are located in the (x_1, x_2) plane is equivalent to knowing where the *decision boundaries* are between regions. Thus the problem of learning a classification function f can be thought of as being equivalent to as learning the decision boundaries between the classes.

Probabilistic Models for Classification: In this framework it is assumed that objects belonging to class k have measurement vectors distributed according to some distribution or density function $P(x|y_k, \theta_k)$, where the θ_k are unknown parameters governing the characteristics of class y_k [21].

For example, consider the two classes, males and females. Let $P(y_k)$, $k = 1, 2$ represent the probability that at conception a person receives the appropriate chromosomes to develop as male or female. The $p(y_k)$ are thus the class *prior* probabilities predicting the probabilities that individual i belongs to class y_k, if there is no other information, i.e., the $P(y_k)$ represent the probabilities of class membership *before* observing the vector X.

Once the $P(x|y_k, \theta_k)$ distribution has been estimated (from the test set), Bayes' theorem can be applied to yield the *posterior probabilities*

$$P(y_k|x) = \frac{P(x|y_k,\theta)P(y_k)}{\sum_{l=1}^{m} p(x|y_l,\theta_l)p(y_l)} \qquad \ldots(2.4)$$

where $1 \le k \le m$

The posterior probabilities $p(y_k|x,\theta_k)$ implicitly carve up the input space X into m decision regions with corresponding decision boundaries. For example, for two classes $(m=2)$ the decision boundaries will be located along the contours where $p(y_1|x,\theta_1) = p(y_2|x,\theta_2)$.

2.5.1.2 Approaches to Building Classifiers

Three fundamental approaches can be considered for building real classifiers.

a) **The discriminative approach:** In this approach the attempt is to model the decision boundary directly, i.e., a direct mapping from input X to one of m class labels $y_1, y_2, \ldots, y_m$. Unlike probabilistic approaches, here no direct attempt is made to model either the class conditional or posterior class probabilities.

   Examples of this approach include neural network, support vector machines, and decision trees when the tree provides only the predicted class at each leaf.

b) **The regression approach:** Here the posterior class probabilities $p(y_k|x)$ are modeled explicitly, and for prediction the maximum of these probabilities is chosen.

   Examples of the regression approach include nearest neighbor methods and decision trees if the tree provides the predicted

class and also the posterior class probability distribution at each leaf.

c) **The class-conditional approach:** The approach explicitly models the class-conditional distribution $p(x|y_k,\theta_k)$ and inverts them along with estimates of $p(y_k)$ via Bayes' rule (Equation 2.4) to arrive at $p(y_k|x)$ for each class y_k.

Examples of the class-conditional approach include the Bayesian classifiers like the naïve Bayes classifier. [21]

Looking at classification problems from these three approaches it can be noted that both the discriminative and regression approaches focus on the differences between the classes, whereas the class-conditional approach focuses on the distribution of the inputs X for the classes.

All the methods are, however, related. Both class-conditional and regression models ultimately produce posterior class probabilities. However, the class-conditional method uses Bayes' rule to calculate the posterior class probabilities, whereas the regression approach is not constrained to do so. Similarly, both the regression and class-conditional methods contain decision boundaries to map inputs X to one of m classes. But unlike the discriminative classifiers, the regression and class-conditional classifiers have to map inputs to classes within the constraints of probabilistic framework [21].

2.5.2 Choosing Classifier Algorithms

The decision of which classifier to use in this research was based on the discussion of section 2.5.1.

The choice of a classifier type depends on the nature of the problem. For applications like medical diagnosis it might be important that a classifier produces posterior class probabilities rather than just class labels. However, for datasets with high dimensions, the discriminative classifier may work better, since it may be very costly to accurately estimate functions $p(x|y_k,\theta_k)$ for high dimensional data.

In general classifiers that use class-conditional methods require fitting most parameters (leading to complex modeling), those that use the regression methods require fewer fitting while those using the discriminative approach fewest of all [21].

Since the experimental dataset of this research has many features (more than 3,000 unique attributes) a classifier that uses the discriminative approach (support vector) and a classifier that uses the regression approach (decision tree) are used for the automatic classification of Amharic news items.

2.5.3 Classifier Selection Criteria

Two major factors that affect an algorithm's performance in a classification task are: the algorithm's inference model and the data processing choices. Successful classification therefore relies on the right model and the right features [20].

The following criteria from [1] maybe used in selecting a model for classification.

- **Accuracy**: The accuracy of a classifier refers to the ability of a given classifier to correctly predict the class label of new or previously unseen data.

- **Speed**: Refers to the computational cost involved in generating and using the given classifier.
- **Robustness**: Is the ability of a classifier to make correct predictions given noisy data or data with missing values.
- **Scalability**: Refers to the ability to construct the classifier efficiently given large amounts of data.
- **Interpretability**: Refers to the level of understanding and insight that is provided by the classifier.

2.5.4 Classifiers in Practice

This section presents a detailed discussion of two classifiers used in the research experiment. One classifier that uses the discriminative approach – namely support vector machines, and a classifier that uses the regression approach - decision tree induction.

The classification methods discussed in this section are analyzed according to the above five criteria and are based on the discussion in [1].

2.5.4.1 Support Vector Machines

Support vector machine (SVM) is a method which can be used for classification of both linear and non linear data. SVM uses a nonlinear mapping to transform the original training data into a higher dimension. Within this new dimension, it searches for the linear optimal separating hyperplane (a decision boundary separating the instances of one class from another). Data from two classes can always be separated by a hyperplane, with an appropriate nonlinear mapping to a sufficiently high dimension. The

SVM finds this hyperplane using essential instances from the training set called **support vectors** [1].

Vladimir Vapnik and colleagues presented the first paper on support vector machines in 1992 as referenced by [1]. Although the training time for SVMs can be slow, they are highly accurate, owing to their ability to model complex nonlinear decision boundaries. Their use of support vectors for identifying decision boundaries makes them much less prone to overfitting than the other methods. Moreover, since they usually are subsets of the training instances the support vectors provide a compact description of the learned model.

SVMs can be used for prediction as well as classification. They have been applied for handwritten digit recognition, object recognition, speaker identification and more other areas [1].

The following two classification problems provide insight on how SVM works: the case when the data are linearly separable and the case when the data are linearly inseparable.

The case when the data are linearly separable

Considering the simplest case of a two-class problem where the classes are linearly separable,

Let the dataset D be given as $(x_1, y_1),(x_2, y_2),...,(x_{|d|}, y_{|d|})$

where x_i is the set of training instances with associated class labels, y_i.

Each y_i can take one of two values either +1 or –1, corresponding to the two classes: class-1 and class-2 respectively.

i.e. $y_i \in \{+1,-1\}$

Consider an example based on two input attributes A_1 *and* A_2 as shown in Figure 2.1

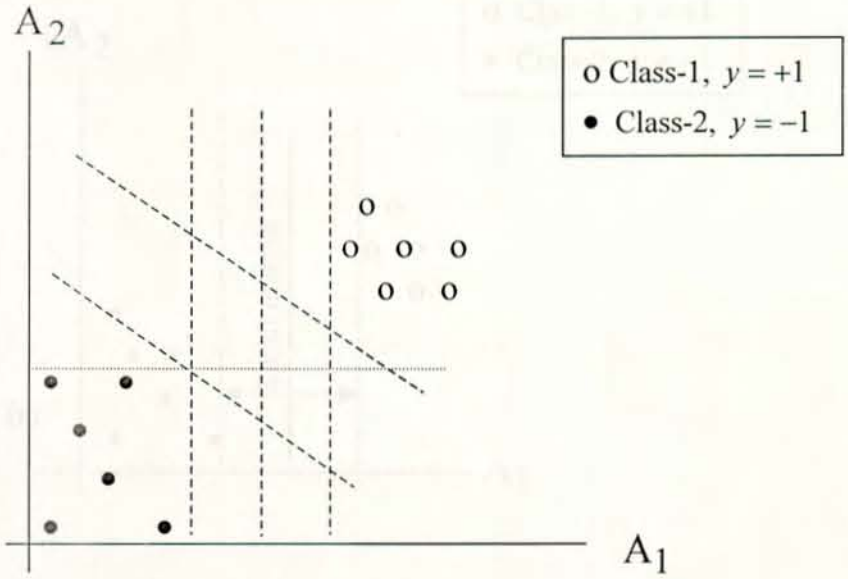

Figure 2.1 Linearly separable data with an infinite number of possible separating hyperplanes

From the figure it can be seen that the 2-D data are linearly separable because a straight line can be drawn to separate all instances of class-1 from all instances of class-2. There are an infinite number of separating lines that could be drawn. The problem is to find the best line that will have the minimum classification error on previously unseen instances. Note that for data with three attributes (3-D data) the problem would be finding the best separating *plane*. Therefore, in general for n-dimensions the problem would be to find the best *hyperplane.*

An SVM approaches this problem by searching for the maximum marginal hyperplane.

Consider Figure 2.2, which shows two possible separating hyperplanes and their associated margins.

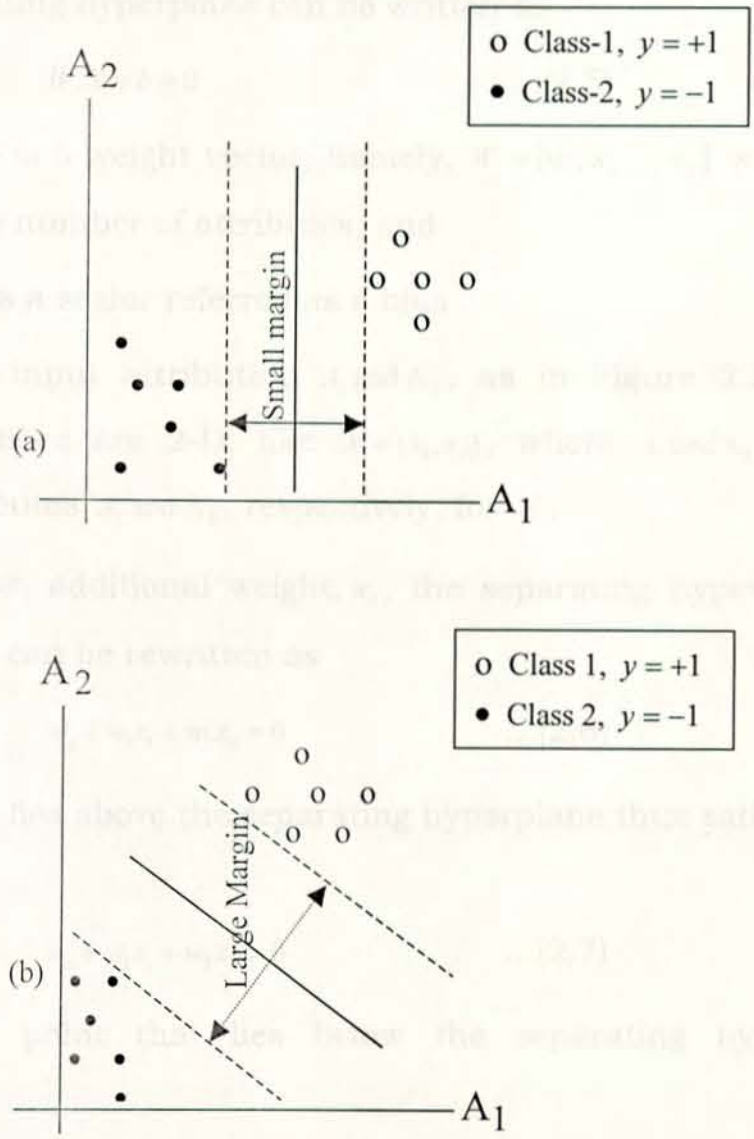

Figure 2.2 Two possible hyper planes and their associated margins

Looking at Figure 2.2 reveals that both hyperplanes can correctly classify all the given data instances. The hyperplane with the large margin is however expected to be more accurate at classifying future data instances than the hyperplane with the smaller margin. This is

why, during the learning phase, the SVM searches for a hyperplane with the largest margin - the maximum marginal hyperplane (MMH).

A separating hyperplane can be written as

$$W.X + b = 0 \qquad \ldots(2.5)$$

Where W is a weight vector, namely, $W = \{w_1, w_2, ..., w_n\}$ where n is the number of attributes, and

b is a scalar referred as a bias

Consider two input attributes, A_1 and A_2, as in Figure 2.2(b). The training instances are 2-D, like $X = (x_1, x_2)$, where x_1 *and* x_2 are the values of attributes A_1 and A_2, respectively, for X.

Taking b as an additional weight, w_o, the separating hyperplane in Equation (2.5) can be rewritten as

$$w_o + w_1 x_1 + w_2 x_2 = 0 \qquad \ldots(2.6)$$

Any point that lies above the separating hyperplane thus satisfies the equation

$$w_o + w_1 x_1 + w_2 x_2 > 0 \qquad \ldots(2.7)$$

Similarly any point that lies below the separating hyperplane satisfies

$$w_o + w_1 x_1 + w_2 x_2 < 0 \qquad \ldots(2.8)$$

The weights can be adjusted so that the hyperplanes defining the two sides of the margin can be written as

$$H_1 = w_o + w_1 x_1 + w_2 x_2 \geq 1 \quad \text{for } y_i = +1 \qquad \ldots(2.9)$$

$$H_2 = w_o + w_1 x_1 + w_2 x_2 \leq -1 \quad \text{for } y_i = -1 \qquad \ldots(2.10)$$

This means any instance that falls on or above H_1 belongs to class-1 and any instance that falls on or below H_2 belongs to class-2

Combining Equation (2.9) and Equation (2.10) one can write

$$y_i(w_o + w_1x_1 + w_2x_2) \geq 1, \ \forall_i \qquad \ldots(2.11)$$

Any training instances that fall on hyperplanes H_1 **or** $\mathbf{H_2}$ satisfy equation (2.11) and are called **support vectors**. They are equally close to the separating MMH [1].

Using a Lagrangian formulation and solving for the solution using Karush-Kuhn-Tucker (KKT) condition, Equation (2.11) can be rewritten as a constrained convex quadratic optimization problem.

Solving the constrained convex quadratic problem is required to find the support vectors and MMH and thus train the support vector machine. Such trained SVM, are called *linear* SVMs, since the MMH is a linear class.

Thus the MMH can be written as a decision boundary, based on the Lagrangian formulation

$$d(x^T) = \sum_{i=1}^{\ell} y_i \alpha_i x_i x^T + bo \qquad \ldots(2.12)$$

Where y_i is the class label of support vector x_i

x^T is test instance

b_o are numeric parameters determined automatically by the SVM algorithm

α_i are Lagrangian multipliers and

ℓ is the number of support vectors.

Using the test instances x^T in equation (2.12) is how classification is done by SVMs. If the sign of the result is positive, then x^T falls on or above the MMH, and SVM predicts that x^T belongs to class-1. If the sign is negative, then x^T falls on or below the MMH and the prediction is for class-2 [1].

The compact prediction model of SVM comes from the fact that the learned classifier is characterized by the number of support vectors rather than the dimensionality of the data. Hence SVMs tend to be less prone to overfitting than some other methods. An SVM with a small number of support vectors can have good generalization, even for a high dimensional data.

The case when the data are linearly inseparable

When the data classes are not linearly separable the approach used for linear SVM can be extended to create *nonlinear* SVMs for the classification of linearly inseparable data. Such SVMs are capable of finding nonlinear decision boundaries (i.e. non linear hypersurfaces) in input space.

Nonlinear SVM extends the approach for linear SVM using two main steps

a) Transforming the original input data into higher dimensional space using a nonlinear mapping and then

b) Searching for a linear separating hyperplane in the new space. Thus getting a quadratic optimization problem that can be solved using the linear SVM formulation

The maximal marginal hyperplane found in the new space corresponds to a nonlinear separating hypersurface in the original space.

Considering the following example of transformation of an input data into a higher dimensional space, a 3-D input vector $X=(x_1,x_2,x_3)$ is mapped to a 6-D space Z, using mappings

$$\Phi_1(X)=x_1,\ \Phi_2(X)=x_2,\ \Phi_3(X)=x_3,\ \Phi_4(X)=(x_1)^2,$$

$$\Phi_5(X)=x_1x_2,\ \Phi_6(X)=x_1x_3$$

The decision hyperplane in the new space is linear and given as

$d(Z)=WZ+b$, Where Z are vectors

Solving the above equation involves choosing a nonlinear mapping to a higher dimensional space and a subsequent costly calculation for the classification of test instant x^T (refer to Equation 2.12). However there is a way of avoiding both.

When searching for linear SVM in the new higher dimensional space, the training instances appear only in the form of dot products [1]

$\Phi(X_i).\Phi(X_j)$, where $\Phi(X)$ is the nonlinear mapping function applied to transform the training instances.

Moreover, applying a *kernel function* $k(X_i,X_j)$ is found to be equivalent to computing the dot product on the transformed data instances, i.e.

$$k(X_i,X_j)=\Phi(X_i).\Phi(X_j) \qquad \ldots(2.13)$$

Equation (2.13) shows how both nonlinear mapping and calculation on transformed data can be avoided. Afterwards the maximal separating hyperplane can be found in a process similar to linear SVM, though the non-linear SVM involves placing a user-specified upper bound, C, on the Lagrange multipliers α_i. This upper bound is best determined experimentally.

Some of the kernel functions that can be used to replace the dot product (See Equation 2-13) include.

Polynomial kernel of degree h: $k(X_i, X_j) = (X_i . X_j + 1)^h$ ...(2.14)

Gaussian radial function kernel: $k(X_i, X_j) = e^{-\|X_i - X_j\|^2 / 2\delta^2}$ ...(2.15)

Sigmoid kernel: $k(X_i, X_j) = \tanh(kX_i . X_j - \delta)$ ...(2.16)

2.5.4.2 Decision Tree Induction

Classification using decision tree induction involves the learning of decision trees from class labeled training instances. A decision tree is a tree like structure, where branches grow out of internal nodes and the tree ends with leaf nodes. Here the internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node holds a class label. The topmost node is the root node [1].

Given an instance X for which the associated class label is unknown, the attribute values of the instance are tested against the decision tree. Testing begins from the root, decisions are made at the branches and class prediction is arrived at a leaf node.

Decision tree classifiers can handle high dimensional data. Their tree-like representation of acquired knowledge is instructive and generally not difficult for human understanding. The learning and classification steps of decision tree induction are simple and fast. In general decision tree classifiers have good accuracy [1].

Classifications by decision tree induction algorithms have many applications including areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology [1].

The pioneering decision tree algorithms were developed in the early 1980s; ID3 and its successor C4.5 (Quinlan) and CART (Breiman et al., as referenced by [1]) laid the cornerstone for subsequent works on decision tree induction.

Most algorithms for decision tree induction adopt a greedy (non-backtracking) approach in which decision trees are constructed by a recursive portioning of the training set into smaller and smaller subsets.

Decision tree algorithms differ from one another in how they select attributes in creating a tree and in the mechanism they use for pruning, as discussed in the following paragraphs [1].

a) Attribute Selection:

Attribute selection measure helps to find the best partition (for a given attribute selection measure) by selecting the criteria that best separates a given data partition, D, of class-labeled training data instances into individual classes.

Attribute selection method provides a ranking for each attribute describing the given training instances. The attribute having the best score for the measure is chosen as the splitting attribute for the given instances. Different attribute selection measures are used in decision tree induction including *information gain, gain ratio,* and *Gini index.*

b) Tree Pruning

It is not possible to completely avoid noise and outliers of the training data from the branches when a decision tree is built. The resulting problem of overfitting data can be addressed by tree pruning methods. Pruned trees are usually faster and better at correctly classifying previously unseen instances, than unpruned trees.

Prepruning and *postpruning* are the two common approaches of tree pruning.

In **prepruning** a tree is pruned by halting the further partition of the subset of training instances at a given node. Halting makes the node a leaf. The class label of the leaf could be the most frequent class among the subset instances.

In **postpruning** a sub-tree at a given node is pruned by removing its branches and replacing it with a leaf. The class label of the leaf would be the most frequent among the sub-trees being replaced.

c) Decision Tree Induction Scalability

Decision tree algorithms like ID3, C4.5 and CART have restriction that the training instances should reside in memory and are inefficient for datasets with very large training instances that do not fit to memory. Thus more scalable approaches, those that do not require swapping of the training data in and out of memory, are required.

More recent decision tree algorithms that are scalable include SLIQ and SPRINT. Both algorithms propose presorting techniques on disk-resident datasets that are too large to fit in memory, and also define the use of new data structures to facilitate the tree construction.

Other methods like RainForest and BOAT (Bootstrapped Optimistic Algorithm for Tree Construction) further enhance the scalability of decision tree induction.

2.5.4.3 Other Classifiers

Other algorithms that have been frequently used for classification include: Bayesian classifiers, Neural Network Learners and Lazy Learners. This section gives brief description of the classification techniques used by these algorithms [1].

a) Bayesian Classification

Bayesian classifiers are statistical classifiers, which predict class membership probabilities like the probability that a given instance belongs to a particular class.

Bayesian classification is based on Bayes' theorem and its classifier models include *naïve Bayesian classifier* and *Bayesian belief networks.*

Naive Bayesian classifiers assume class-conditional independence (i.e. they assume that the effect of an attribute value on a given class is independent of the values of other attributes). They are found to be comparable in performance with decision tree and selected neural network classifiers.

Bayesian belief networks, unlike naïve Bayesian classifiers, allow class conditional independencies to be defined between subsets of variables. Bayesian classifiers have shown high accuracy and speed when applied to large datasets.

b) Neural Network Learning

A neural network can be considered as a set of connected input-output units in which each connection has a weight associated with it. In the training phase neural network learns by adjusting the weights, which takes long training times.

It is usually difficult for humans to interpret the symbolic meaning behind the learned weights and the hidden units of neural networks resulting in poor interpretability.

Neural networks, however, have high tolerance for noisy data and they have been useful in pattern predictions. They have been applied for practical problems including handwritten character recognition, and training a computer to pronounce English text.

c) Lazy Learners

Classifiers which use the training data to build a classification model and wait readily to classify the test data are called **eager learners.** Eager learners include SVM, decision tree induction, Bayesian classification, and neural network learners.

On the other hand a learner, which simply stores a trainee data and waits until it is given the test data to build a classification model is called a **lazy learner. K-Nearest-Neighbor Classifier** is an example of a lazy learner.

Lazy learners can be computationally expensive for classification. They offer little explanation or insight into the structure of the data. Lazy learners, however, support incremental learning. They are able to model complex decision spaces having hyperpolygonal shapes that may not be as easily describable by other learning algorithms.

2.5.5 Classifier Performance Measures

In this thesis classifiers are used for automated category assignment, i.e., for binary classification tasks. This section therefore focuses on binary classification performance measures.

As discussed in Section 2.4 the **accuracy** of a classifier should be estimated from the classifier's performance on a test data and is the percentage of test set instances that are correctly classified by the classifier.

In general the category assignment of a binary classifier can be evaluated using a *confusion matrix* which is a tool for analyzing how well a classifier recognizes instances of different classes.

Table 2.1 shows a two-class confusion matrix which shows positive instances (documents of the class of interest, i.e., yes for class y_1) versus negative instances (no for class y_2)

Table 2.1: Two Class Confusion Matrix

		Predicted Class	
		y_1	y_2
Actual Class	y_1	True positives	False negatives
	y_2	False positives	True negatives

In Table 2.1 true positives refer to positive instances correctly labeled by the classifier, true negatives are negative instances correctly labeled by the classifier, false positives refer to negative instances

incorrectly labeled, and false negatives are positive instances that were incorrectly labeled.

In addition to accuracy the experimental data may require other performance measures like **sensitivity (recall), specificity, precision, error, F-measure, and ROC Area** [1].

Sensitivity is the proportion of positive documents that are correctly identified

$$\text{i.e. Sensitivity (recall)} = \frac{t-pos}{pos} \qquad \ldots(2.17)$$

where *t- pos* is true positives and,

pos is the number of positive documents

Specificity is the proportion of negative instances that are correctly identified by the classifier

$$\text{Specificity} = \frac{t-neg}{neg} \qquad \ldots(2.18)$$

where *t-neg* is the number of true negatives and

neg is the number of negative instances

Precision is the proportion of documents labeled as yes that is actually yes instances

$$\text{Precision} = \frac{t-pos}{t-pos+f-pos} \qquad \ldots(2.19)$$

where *f-pos* is the number of false positives

Now **accuracy** can be written in terms of sensitivity and specificity as

$$\text{Accuracy} = \text{sensitivity}\left(\frac{pos}{pos+neg}\right) + specificity\left(\frac{neg}{pos+neg}\right) \quad \ldots(2.20)$$

$$\text{Or} \quad \text{Accuracy} = \frac{t-pos+t-neg}{pos+neg} \quad \ldots(2.21)$$

The **error** measure gives the proportion of documents assigned incorrectly, i.e.

$$\text{Error} = \frac{f-pos+\text{f - neg}}{\text{pos}+\text{neg}} \quad \ldots(2.22)$$

Usually a classifier exhibits a trade-off between recall and precision, i.e., when the decision thresholds in the classifier are adjusted to get a high recall, the consequence is lowering the precision and vice versa. If the recall and precision of a classifier can be tuned to have an equal value, then this value is called the *break-even point* (BEP) of the system.

The F-measure is defined in terms of recall and precision and is often used as an optimization criterion in threshold tuning for binary decisions [26]:

$$F_\beta(recall, precision) = \frac{(\beta^2+1)(precision \times \text{recall})}{\beta^2(precision)+\text{recall}} \quad \ldots(2.23)$$

where β is the parameter allowing differential weighting of precision and recall

If recall and precision are given equal weight, Equation (2.23) will be simplified to give the F_1 measure, i.e.,

$$F_1 = \frac{2(\text{recall} \times \text{precision})}{(\text{recall} + \text{precision})} \quad \ldots(2.24)$$

ROC *(Receiver Operating Characteristics)* curves show the trade-off between the true positive rate (sensitivity) and the false positive rate for a given model. The area under the ROC curve is a measure of the accuracy of the model [1].

For a given classifier, if for every true positive there is a counter false positive, the ROC curve approaches a diagonal line. The area under the diagonal is 0.5. Therefore the closer the area (under a classifier's ROC curve) is to 0.5 the less accurate is the classifier, while the area under the ROC curve of a perfect classifier is 1.

2.5.6 Classifier Accuracy Evaluation

A number of different techniques are used to reliably estimate the accuracy of classifiers using the relations defined in section 2.5.4

The techniques include holdout, random sampling, **cross-validation, and bootstrap.**

In the experimental part of this research random sampling and cross-validation techniques are used. Brief discussion on each of these two methods is presented in this section.

2.5.6.1 Holdout and Random Sampling

In the **holdout** method the given data are randomly partitioned into two independent sets: a training set (usually having 66% of the dataset) and a test set (with the remaining 33% of the dataset). The

accuracy estimate of the holdout method is pessimistic since only a portion of the training set is used to derive a classifier model [1, 24].

Random sampling repeats the holdout method a number of times and takes an overall accuracy estimate by averaging the accuracies obtained from each iteration.

2.5.6.2 Cross-validation

The cross-validation method can be generalized into two as **k-fold cross-validation** and **stratified cross-validation**:

In k-fold cross-validation the initial data are randomly partitioned into k mutually exclusive subsets (folds), $D_1, D_2, ..., D_k$, each of approximately equal size. In iteration i, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model, which will continue for all k iterations. Unlike the random sampling method here each partition is used equal number of times for training and once for testing. Classification accuracy is estimated by dividing the overall number of correct classification from the k iterations by the total number of instances (documents) in the initial data [1, 24].

In stratified cross-validation, the folds are stratified so that the class distribution of the documents in each fold is approximately the same as that in the initial data.

Generally in practice stratified 10-fold cross-validation is employed for estimating accuracy due to its relatively low bias and variance. The stratified 10-fold cross-validation is used for all experiments in this research.

CHAPTER THREE

THE AMHARIC WRITING SYSTEM

3.1 INTRODUCTION

This chapter gives a brief description of the Amharic writing system by focusing mainly on the electronic representation of Amharic characters.

Amharic is the working language of the Federal Government of Ethiopia and is spoken and written as a first or second language in many parts of the country.

Amharic, like other languages that use the Ethiopic script (Gurage, Harari, Tigre, and Tigrniya), use characters derived mainly from Geez.

The Ethiopic script was first displayed on a computer around 1986. Who was the pioneer in this endeavor is controversial but one of them was the then Ethiopian Science and Technology commission (ESTC). At the time the challenge in the computer representation of the script was developing a software package that can handle character design, keyboard layout and printer set-up.

The pioneering work by ESTC started an enthusiastic rush to develop Ethiopic software by different IT companies and teams of individuals which led to the problem of lack of standardization. At the present there are at least 35 Ethiopic software products available, each with its own character set, encoding system, typeface names and keyboard layout.

The recent development of the introduction of the Ethiopic range with the Unicode standard could help in standardizing the different incompatible software products.

3.2 AMHARIC CHARACTER REPRESENTATION

Geez has been a language of literature in Ethiopia up to recent time and is now used for the liturgy of the Ethiopian Orthodox Church. Written Geez can be traced back to at least the 4th century A.D. The first versions of the Geez script included only consonants while the characters in the later versions represent consonant-vowel (CV) phoneme pairs [5].

Amharic has borrowed most of its characters from Geez and thus the Amharic writing uses characters created by a CV fusion. Sven vowels are used in Amharic each of which comes in seven different forms (orders) reflecting the seven vowel sounds (ሀ ሁ ሂ ሃ ሄ ህ ሆ). That is each of the 33 Amharic characters has seven forms representing a consonant and a vowel at the same time which makes the Amharic script **syllabic**. The first order is the basic form and there are 33 basic forms giving 231 characters [22].

As examples, the symbolic representations of the seven forms of the Amharic characters በ (be), ገ (ge), ደ (de) are as shown in Table 3.1.

Table 3.1 Seven forms of Amharic Characters

Consonant	1st order	2nd order	3rd order	4th order	5th order	6th order	7th order	
ለ	ለ	ለ·	ለ.	ላ	ሌ	ል	ሎ	the seven forms of ለ
	ለኧ	ለኡ·	ለኢ.	ለኣ	ለኤ.	ለ	ለኦ	Consonant-vowel representation
	bä	bu	bi	ba	be	B	bo	Represented sound
ገ	ገ	ገ·	ጊ.	ጋ	ጌ	ግ	ጐ	the seven forms of ገ
	ገኧ	ገኡ·	ገኢ.	ገኣ	ገኤ.	ገ	ገኦ	consonant-vowel representation
	gä	gu	gi	ga	ge	G	go	Represented sound
ደ	ደ	ዱ.	ዲ.	ዳ	ዴ	ድ·	ዶ	the seven forms of ደ
	ድኧ	ዱኡ·	ዲኢ.	ዳኣ	ዴኤ.	ድ·	ዶኦ	consonant-vowel representation
	dä	du	di	da	de	D	do	Represented sound

The Amharic FIDEL also includes 20 symbols for labialized Velars (i.e. ቈ ጐ ኰ ጐ - each with five orders) and 24 for other labialization (Example ኳ. ሟ ሯ ... ፏ. ፗ. ፘ:). The FIDEL therefore has 275

characters (letters) to be used in the writing system. Amharic also has its own numbers (20 symbols) and its own punctuation system where the symbol ፡ (*hulet neteb*) is used to separate words, the symbol ፣ (*netela serez)* is used as a comma, the symbol ፤ (*dereb serez*) is used as a semicolon, and the symbol ። (*arat neteb*) is used as a full stop. The question and exclamation marks have recently been included in Amharic writing system [5].

It can be seen from Table 3.1 that the signs of the vowels have become an integral part of the Amharic letters so that it can be said that there are over 280 symbols in the script almost independent of each other. (The Amharic character list - The Amharic **FIDEL** - is attached as Annex 1)

3.3 PROCESSING AMHARIC DOCUMENTS

This research uses the automatic classification approach to categorize Amharic documents. The first step in automatic classification of documents is the selection of feature words that represent the documents.

As there can be millions of words in text datasets storage and processing time costs require document processing for efficient and reliable automatic classification.

Document processing is therefore an important task to get features that adequately represent a document without being redundant and irrelevant.

In this research the following characteristics of the Amharic writing system are considered during the processing of the Amharic documents of the source dataset.

3.3.1 Characteristics of the Amharic Writing

The characteristics of the Amharic writing system considered in this section are limited to those that are common to news texts.

Character Redundancy: Amharic took the whole Geez alphabet (all seven orders of the 26 symbols of Geez) without considering whether all the 26 characters have meaning in the Amharic writing system. It then added some more symbols for some other sounds that it has and that could not be represented by the symbols of the Geez alphabet. This unsystematic borrowing from Geez has resulted in redundant characters in the Amharic FIDEL [22].

Table 3.2 shows an example of the character redundancy where more than one symbol is used for a given sound.

Table 3.2 Amharic Characters with Same Sound

Consonants	Other symbols with the same sound
ሀ (hä)	ሃ ሐ ኀ and ኻ
ሰ (sä)	ሠ
አ (ä)	ኣ ዐ and ዓ
ጸ (tsä)	ፀ

Therefore, and by considering other similar characters only about 233 of the 275 characters are actually necessary to represent Amharic [5], i.e., by using only one character from a group of characters with the same sound.

Spelling variations of a word would unnecessarily increase the number of words representing a document which could reduce the efficiency and accuracy of the classifiers. Amharic document

processing for feature selection should therefore normalize word variants (spelling differences) caused by inconsistent usage of redundant characters.

During the pre-processing stage of Amharic documents for this research, the different forms of a character that have the same sound are changed to one common form.

Table 3.4 shows examples of the different word spellings caused by the redundant characters.

Table 3.3 Word Spelling Variations

The Word in English	The Word in Amharic	Spelling Variants of the Word
Work	ሥራ	ስራ
Sun	ፀሐይ	ጸሐይ፣ ፀሀይ፣ ጸሀይ
World	አለም	ዐለም፣ ዓለም፣ አለም
Power	ሀይል	ሃይል፣ ኃይል

Compound Words Usage: in the Amharic writing system, inconsistency is often observed regarding the representation of compound words. Some compound words are used as a single word in some instances (either by fusing the two words or by inserting a hyphen between them) and as two separate words at other instances. Table 3.4 shows some examples of the inconsistency in the use of compound words.

Table 3.4 **Compound Word Usage**

Compound word used as a single word	**Literal English meaning**	**Compound word used as separate words**	**Literal English meaning**
አዲስአበባ	Addis Ababa	**አዲስ አበባ**	New flower
ትምህርትቤት (**ትምህርት-ቤት**)	School	**ትምህርት ቤት**	Education house
ቀዶጥገና (**ቀዶ-ጥገና**)	Surgery	**ቀዶ ጥገና**	Cutting and fixing

Inconsistent usage of compound words could result in redundant word features by creating more words when a compound word (example **አዲስ-አበባ**) is treated as two separate words **አዲስ** and **አበባ**. It may also result in a semantic loss by confusing a document about the city Addis Ababa (**አዲስ-አበባ**) with the one talking about the floral industry.

Variations due to Pronunciations: usage of foreign language words in Amharic (transliteration) is also found to be another source of word spelling variations. Observations of the ENA's news documents shows that in most cases in the writings of words adapted from foreign languages different writers use different spellings. The cause of the difference in the Amharic spellings of these foreign language words seems to be the difference in the pronunciations of these words.

For example, the word **ሚቲዎሮሎጂ** (Meteorology) is found to have 14 different Amharic spellings in the source data. Table 3.5 shows

examples of spelling variation in the writing of foreign words in Amharic.

Table 3.5 Word Variations due to Transliteration

Foreign Word	Equivalent Words in Amharic usage
Meteorology	ሚትሪዮሎጂ፣ ሚትዎሮሎጂ፣ ሚትዮሮሎጂ፣ ሚቲዎሮሎጂ፣ ሚቲዎሮሎጅ፣ ሚቲዎሮሊጂ፣ ሜትዎሮሎጂ፣ ሜትሮዎሎጂ፣ ሜትሮዎሎጅ፣ ሜትሪዎሎጂ፣ ሜትሮሎጂ፣ ሜትርዎሎጂ፣ ሚቲዎሮሎጂ፣ ሜቴዎሮሎጂ
Million	ሚሊዮን፣ ሚልዮን፣ ሚሊየን
Television	ቴሌቪሾን፣ ቴሌቭሾን፣ ቴሌቭዝን፣ ቴሌቪዝን

Moreover there are word spelling variations that could be attributed to variations in pronunciations at different parts of the country, like for example using the two words ጠባይ and ፀባይ to mean temperament or using the three words ጢንዚዛ, ጢንዝዛ and ጥንዚዛ to mean beetle.

Other Cases of Word Variations: Difference in word affixing has also been observed to cause word spelling variations. For example difference in suffixing would result in the two writings አእምሮአዊ and አእምሯዊ to refer to human intellect while difference in prefixing would give the two writings ለአንድ and ላንድ to mean 'for one'.

3.3.2 Processing Tools

In the processing of the source dataset for this research various tools have been developed to control redundant and irrelevant words that could have been taken as document attributes due to the characteristics of the Amharic writing system discussed in section 3.3.1.

The focus and depth of pre-processing requirement depends on the purpose of the automatic classification problem [27]. This research focuses on automatic classification of Amharic documents. The special nature of the Amharic language and its writing system, the lack of standard Amharic corpus, and the unavailability of processing tools present unique pre-processing challenges.

Neither full Amharic morphological analyzer nor Amharic stemmer was available (accessible) to this research. Therefore tools for simple word suffix and prefix removal was adapted from the tool in [7] and enhanced to include semantic analysis to see the effects of the suffix and prefix removal.

Moreover various pre-processing tools were developed including those used to control

- Word spelling variations due to pronunciation differences
- For the inconsistent use of redundant Amharic characters (with some adaptation of the work in [7])
- Word variations due to gender marker suffixes
- Word variations due to number marker suffixes
- The number of words representing a document by removing stop words

- Inconsistent usage of compound words

3.4 COMPUTERIZING THE AMHARIC SCRIPT

The data source for the experiment of this research is the Amharic news database of the Ethiopian News Agency (ENA). The news articles sampled are the daily news items of ENA spanning from January 2003 to June 2006. ENA's news items in the specified period are written using the Amharic software Visual Geez Version 2.0 (VG2).

The VG2 software has to represent the over 280 Amharic characters using the English language keyboard designed to recognize only 26 letters. This means the Amharic software has to use key combinations for the characters in the Amharic FIDEL (i.e. the software has to consider most Amharic characters as a combination of two characters).

Moreover, the ASCII code does not recognize Amharic scripts and thus cannot assign numeric codes to the scripts. The Amharic software therefore has to perform the necessary mapping of Amharic characters to numerical codes. Table 3.6 shows sample representation of Amharic characters in VG2.

Table 3.6 Amharic Script Representation in VG2

Consonant	1st order	2nd order	3rd order	4th order	5th order	6th order	7th order
Amharic script ሀ	ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ
VG2 representation	h	h#	£	¦	ÿ	H	Ç
Amharic script ለ	ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ
VG2 representation	l	l#	l!	§	l@	L	lÖ
Amharic script ሐ	ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ
VG2 representation	/	/#	/!	ˆ	/@	?	‡

The full representation of the Amharic script in Visual Geez Version 2 (VG2) can be seen in Annex 2.

CHAPTER FOUR

AUTOMATIC CLASSIFICATION OF AMHARIC TEXTS

4.1 INTRODUCTION

This chapter discusses the data source for the research, the data processing carried out, and the testing procedures followed for the automatic classification of Amharic documents. Moreover, the performances of classifier models used in classifying Amharic news documents will be shown and the performances of these classifiers will also be compared.

As mentioned in Sections 1.4.2 and 3.4 the data source for this research is the news articles database of the Ethiopia News Agency (ENA). In this section the logic behind the choice of the data source, the data sampling procedures used, and the extent of the pre-processing carried out will be discussed.

The processing involved for selecting category representative words out of the news items of a category will be explained. All major data processing activities performed on the data like stemming, spelling, gender, number normalization, stop word removal, and scaling will be detailed.

The steps followed in the conversion of the preprocessed data to the **ARFF** file format, which is appropriate for use in the Weka application package, will be explained. The application of the Weka package to classify the Amharic news items will be shown.

Finally the performances of SVM and Dtree classifiers will be compared on various scenarios (different sets of categories) and the comparison result analyzed.

4.2 DESCRIPTION OF THE PRE-PROCESSING SUB-SYSTEM

The pre-processing step of the experiment involves word-level processing of the source dataset with the ultimate aim of identifying feature words that are representatives of the documents in the dataset. This step also involves the conversion of the pre-processed data to **Arff**, which is suitable for the Weka package used for the automatic classification.

The design of the pre-processing sub-system depicted in Figure 4.1 contains only the major sub-system components.

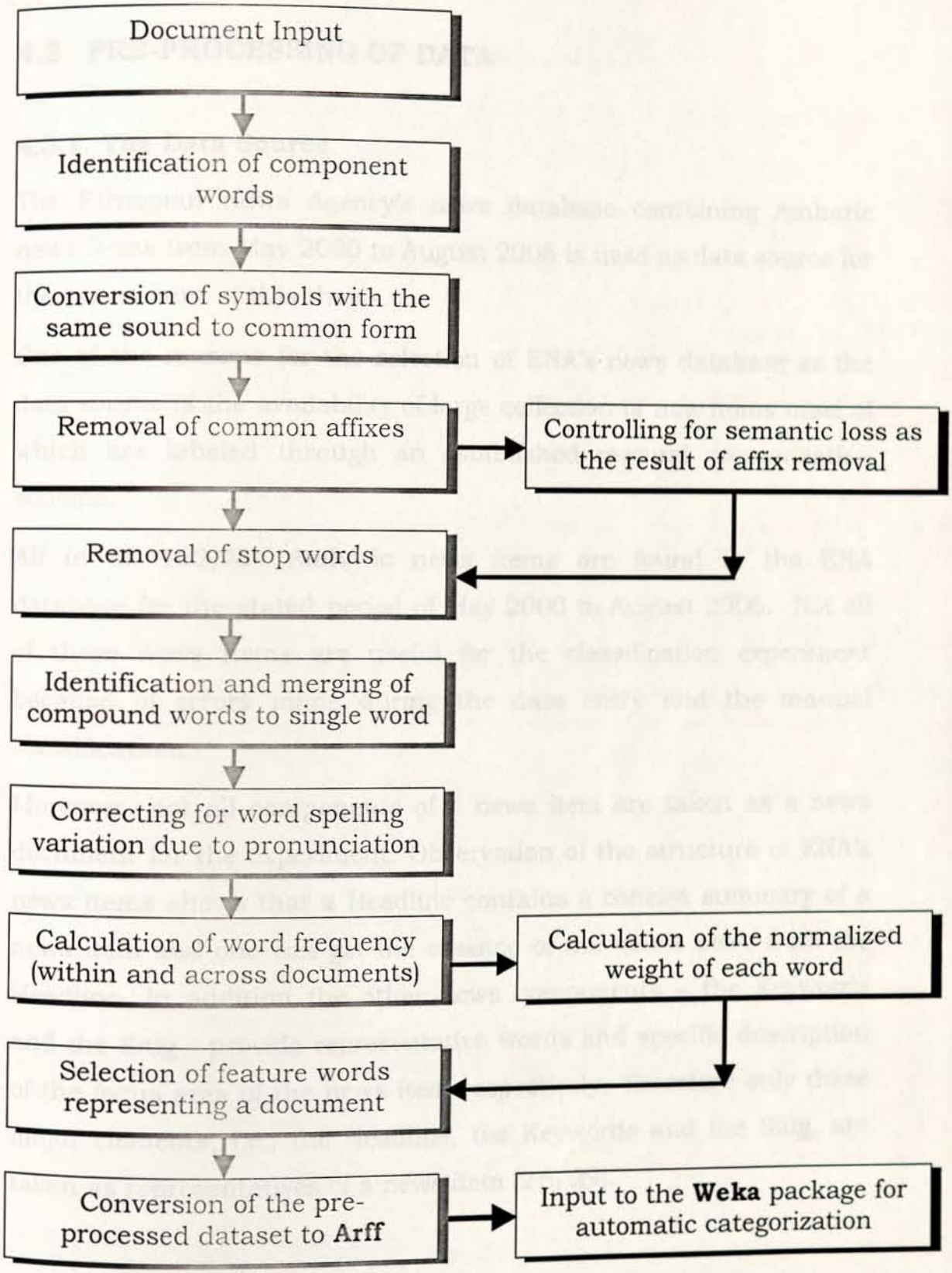

***Figure 4. 1** The general description of the data pre-processing sub-system*

4.3 PRE-PROCESSING OF DATA

4.3.1. The Data Source

The Ethiopian News Agency's news database containing Amharic news items from May 2000 to August 2006 is used as data source for the experiment of this thesis.

One of the reasons for the selection of ENA's news database as the data source is the availability of large collection of new items most of which are labeled through an established manual categorization scheme.

All in all 102,984 Amharic news items are found in the ENA database for the stated period of May 2000 to August 2006. Not all of these news items are useful for the classification experiment because of errors made during the data entry and the manual classification.

Moreover, not all components of a news item are taken as a news document for the experiment. Observation of the structure of ENA's news items shows that a Headline contains a concise summary of a news item and one can get the essence of the whole news from the Headline. In addition the other news components – the Keywords and the Slug - provide representative words and specific description of the focus area of the news item respectively. Therefore only these major elements, i.e., the Headline, the Keywords and the Slug, are taken as representatives of a news item [28, 29].

4.3.2. Problems of the News Classification at ENA

At ENA news items are manually classified into 16 categories with a number of sub-categories under each of the categories. The 16 news categories and their sub-categories are shown in Table 4.1.

Table 4.1 News categories in use at ENA

No	Category name	Category code	No of sub-categories (Amharic)
1	ሕግና ፍትህ (Law and Justice)	ሕናፍ	13
2	ጤና ጥበቃ (Health)	ጤናጥ	47
3	ጥቆማ (Events Directory)	ጥቆማ	-
4	ዓለም አቀፍ ጉዳዮች (International Relations)	ዓአጉ	21
5	ማህበራዊ (Social Affairs)	ማኅበ	26
6	ባህል ጉዳዮች (Culture)	ባሕጉ	13
7	ብሔራዊ ፖለቲካ (Politics)	ብፖለ	32
8	ግብርና ጉዳዮች (Agriculture)	ግብጉ	21
9	መከላከያና ዐጥቃ (Defense and Security)	መናጸ	-
10	ሳይንስና ቴክኖሎጂ (Science and Technology)	ሣንቴ	9
11	ስፖርት (Sport)	ስፖት	39
12	ትምህርት (Education)	ትምህ	40
13	ኢኮኖሚ (Economy)	ኢኮኖ	113
14	አደጋዎች (Accidents)	አደዎ	11
15	የአየር ዐባይ (Weather)	የአዐ	6
16	ሌሎች የፊርጅ ዓይነቶች (Other Classes)	ሌፊዓ	4

The current news classification scheme at ENA has been in use since May 2000. The system, however, hasn't brought the service delivery to the level desired by ENA [23]. Because of the observed non-uniform application of the classification system and its impact on efficient service provision, ENA has conducted two in-house studies [23] to identify the problem areas of the system and to suggest solutions.

The major classification problems identified by the study conducted in 2006 are:

a) Problems with the news categories

- *Lack of clear distinction between some news categories*

  For example confusion is often observed in differentiating the **Agriculture** and the **Economy** classes. Though to a lesser extent, the confusion is also seen in differentiating between the news items of the **Social** and the **Health** classes, the **Social** and the **Culture** classes, and the **Politics** and the **Law and Justices** classes.

- *Consideration of the events directory as a news category*

  The existing classification system considers the **Events Directory** as one news category. In fact the Events directory is not a news category. It is kept for the purpose of reminding (alerting) reporters about coming news events. This misclassification has been causing problems in subsequent activities on the news database like searching for a particular news item.

b) Classification errors

- There is no strict process to validate the categories given by reporters to the news items they are entering into the system. As a result, classification errors were found in news items that are not difficult to classify. In this regard the **Social Affairs** category has been found to be the hiding place of many news items belonging to other categories.
- News items have been entered with class codes different from the 16 recognized in the ENA classification scheme (see Table 4.1). During data pre-processing for this research it was found that the Amharic news database contains a total of 38 news categories. Table 4.2 shows the class labels and data entry records (including records with no data) of the 22 extra categories created due to classification errors.

Table 4.2 Errors Entered as Main News Category in ENA's Amharic News database

No	Category code	No. of records (including empty records)
1	ዓመባ	305
2	ዓለማ	1,114
3	ፓለ	961
4	ማኅደ	5,749
5	ማህ	1,944
6	ማቆጥ	2,720

No	Category code	No. of records (including empty records)
12	ሊዳ	187
23	ሌላ	15
14	መኮዳ	0
15	ሚቲክ	198
16	ጠቅል	651
17	ስፓ	109

No	Category code	No. of records (including empty records)
7	ባህ	35
8	ባህል	312
9	ፍፍት·	1,315
10	ጊበዜ	566
11	ኩንስ	36

No	Category code	No. of records (including empty records)
18	ቱዝ	20
19	ጋደአ	1,853
20	ወን	82
21	ኢኮ	777
22	አየጸ	0

c) Entry errors

- A look at the representative components of news items (The Headline, the Keywords, and the Slug) shows data entry errors on these fields also. For example, 864 news items are entered without a Headline, while 504 with NULL and 256 records with the '?' character for Headline. Similar data entry errors are also observed in the other representative components. Table 4.3 shows the summary of these entry errors.

Table 4.3 Entry errors in representative components of news items

News Item Component	IS NULL	No Entry	LIKE '?'
Headline	504	864	256
Keywords	956	897	1
Slug	0	539	1263

4.3.3 The Sampling

Because of the problems of ENA's news classification system discussed in section 4.2.2 the following steps were followed during the sampling of data for the research experiment.

Condition for sampling: At ENA all news items (Amharic and English) are stored in SQL Server database. The Table object holding the news items has the following fields: *ID, NewsID, Headline, DateNewscreated, Keywords, Classification, Slug, SubCclassification, and FullStory.*

Out of these fields the data on the Headline, Keywords and Slug is considered as representative of a news item. News items were considered for the study only if they have data for the Headline, for the Keywords and for the Slug. If data is missing in one of the three news sections the news item is dropped.

Category selection: Out of the 16 categories used on ENA's classification system only 15 are considered for this study. The **Events Directory** (ጥቆማ) category is not used because the category

does not contain news items, it only shows (lists) events, that are believed to be potential news, for reporters consumption.

Clearing misclassification cases: A lot of attention was given for the identification and clearing of the misclassification errors. Since most of the classification errors are results of the lack of clear distinction between some of the categories [23], automation efforts like clustering were not found effective. Manual scanning of news items was therefore used with the help of experienced professionals from ENA, a process which took a lot of time and effort. As a result, the following classification errors were identified.

- Entering the same news item to more than one category
- Entering a news item two or more times into one category
- Entering news items to categories that have no relation to the news

After the identification of the misclassified news items necessary correction actions were taken during the sampling of news items for the experiment, i.e., duplicate news items were dropped for the sample. News items that were given a clearly wrong category label were removed from that particular category.

For example, in the Social Affairs category, 7,123 data entry errors were identified. Out of which 128 are empty records, 25 are repeated news entries, and 6,970 are entries that do not belong to the Social Affairs category (i.e. classification errors). Out of the 6,970 classification errors, 3,381 belong to the Events Directory while the remaining belonging to the Accidents, the Education, the Health, and other categories.

Similarly, a total of 1,859 records of the Economy category were identified as error (empty entry, repeated entry and misclassification errors).

Table 4.4 shows news items sampled for the study after correcting the identified classification and data entry errors. After the dropping of the Events Directory (ጥቆማ) category, the remaining 15 categories are taken in the sample data.

Table 4.4 Sampled News Items

No	Category Name	Category Code	No of News Items Sampled
1	Law and Justice	ሕናፍ	2950
2	Health	ጤናጥ	6649
3	International Relations	ዓአጉ	5459
4	Social Affairs	ማ·ነበ	21741
5	Culture	ባሕጉ	775
6	Politics	ብፖለ	6789
7	Agriculture	ግብጉ	4944
8	Defense and Security	መናጸ	186
9	Science and Technology	ሣና·ቴ	629
10	Sport	ስፖር	3341
11	Education	ት·ምህ	4944
12	Economy	ኢኮኖ	9757
13	Accidents	አደዎ	687
14	Weather	የአፀ	333

No	Category Name	Category Code	No of News Items Sampled
15	Other Classes	ሌሊ	500
		Total	**69,684**

4.3.4 Data cleaning

Since news items are represented by the Headline, the Keywords and the Slug a procedure is written in Visual Basic to take only the data in these fields from the records of the source data.

The procedure collects all records belonging to a given category in a SQL Server Table – A table for the Health class, another table for the Education class, etc.

The sampling procedure checks for the following conditions to make sure that the sampling requirements are met. The procedure checks whether

- a category code is correct (i.e., the data is of the class under consideration).
- the data in the Headline, the Keywords, and the Slug satisfy the sampling requirements (no blank data in these fields).
- records marked as duplicate or belonging to a different category are not sampled.

4.4 DOCUMENT PROCESSING

Section 4.2 detailed the data cleaning process which is one of the steps of document preparation for classification. This section discusses the other steps that are important for the efficiency and accuracy of automatic classification, i.e., relevance analysis, data transformation, and data reduction.

4.4.1 Representing Documents by Relevant Words

Since documents are considered as a collection of words for the purpose of classification, identifying words that adequately represent a document is important. A document can have many words but not all words of the document are equally important to uniquely identify the document. Thus words that are relevant representatives of documents (i.e. *feature words*) need to be identified. In this study the following methods are employed to select the feature words of documents.

Word identification: The procedure for identifying the words in a document makes use of the Amharic word separators (single space, *netela serez* ፣, *hulet neteb* ፡, *dereb serez* ፤, *arat neteb* ።, carriage return, line feed, tab, etc.). The procedure also deletes a hyphen between words to merge hyphen separated words (example **ክፍለ-ክፉማ**) into a single word (**ክፍለክፉማ**). The same is done for a dot that is between words or letters (for example **ዓ.ም** is changed to **ዓም**)

Collection of numbers is not considered as a document representative word (for example, the year 1999 is not considered a word). The decision to ignore numbers in feature selection is based on the observation that usually numbers do not contribute to uniquely identify one news document from the other. Similarly

single-letter words of Amharic (for example ና, ኑ, etc.) are not considered as words.

In the process of extraction of words from documents, the different Amharic letters that have the same sound are changed to a common form (for example, ሐ, ሓ, ኀ, and ኃ are all changed to ሀ).

Stop word removal: Words that are common in Amharic documents (like ነው, እና, and ነበር) do not help to differentiate one document from another. Such stop words are ignored during feature selection. News specific stop words like አስታውቀዋል, ይካሄዳል, and ተጠናቀቀ are not considered for feature selection either. The feature selection procedure looks up to a table containing the 612 stop words (including news specific stop words) identified in this study and drops those found in the stop words table, from consideration for feature selection.

Stemming: Varieties of a word created by language requirements of affixing could result in feature words redundancy which in turn could reduce the efficiency and accuracy of the automatic classification. The procedure for feature selection removes common prefixes and suffixes to change word variants to one common form. For example, the following four variants ልማትና, ልማትንና, ልማትንናም, ልማትንምና are changed to their base term ልማት *'lemat'* (Development). Similarly the variants ሰላምና, ሰላምንና, ሰላምንናም, ሰላምንም are changed to ሰላም *'selam'* (Peace).

The unavailability of full morphological analyzer for Amharic meant that the procedure of prefix and suffix removal could not differentiate between real affixes and letters that are parts of an Amharic word

(i.e., the procedure simply strips-off letters considered to be suffixes or prefixes from the word).

For example, a simple prefix removal procedure would not identify that the letter ከ is not a prefix in the word ከተማ *'ketema'* (City) or that the letter በ is not a prefix in the word በጀት *'bejet'* (Budget). Similarly a simple suffix removal procedure will fail to recognize that the letter ና is not a suffix in the word ደመና *'damena'* (Cloud) or that the letter ም is not a suffix in the word አለም *'alem'* (World).

The solution given to this problem of failing to identify real affixes from letters that are parts of a word is - adding some intelligence to the affix removal procedure by building an affix control database. The affix control database is a collection of words whose letters should not be considered as either suffix or prefix.

Controlling spelling variation: Observation of the Amharic news words that have been spelt in different ways seems to indicate that the cause of the spelling variations is the difference in the Amharic pronunciation of the words. Most of these words are foreign words adapted to Amharic. For example, the word **exhibition** adapted from the English language has been spelt in the following four different ways in ENA's news items: **ኤግዚቢሽን, ኤግዚብሽን, እግዚቢሽን, እግዚብሽን.**

Some Amharic words have different spellings due to the difference in their pronunciation on the different parts of the country. For example, the word *'tsebay'* which means **character** is found written as ፀባይ or as ጠባይ.

In the feature selection procedure there is a table look-up to correct for these spelling variations. The spelling correction focuses on word

spelling variations due to transliterations. Word spelling variations due to usage differences at different localities are also considered to some extent.

Identifying compound words: To some extent there is lack of uniformity in the writing of Amharic compound words. A compound word is sometimes written as a single word and sometimes as two separate words. This non-uniform way of writing could result in semantic loss during document representation - thereby reducing the accuracy of document representation.

For example, writing the compound word *'meker-bet'* as a single word ምክርቤት gives a meaning equivalent to a **council**. This word may lose its meaning (in the context of the document that contains the compound word) if it is written as two separate words *'miker'* ምክር and *'bet'* ቤት meaning **advice** and **house** respectively.

To avoid the inaccuracy in document representation that might be introduced due to the non-uniform writings of compound words, the feature selection procedure converts compound words to a single word (using a lookup table for the collection of compound words).

That is, the feature selection procedure scans back to check if the current word and its predecessor are parts of a compound word. It then merges the current word with its predecessor if the two words are parts a compound word.

4.4.2 Data transformation and scaling

After feature selection, a document is treated as collection of the representative feature words while the whole dataset is a collection of documents. Feature words representing a document are therefore

attributes of the document that are given some value to reflect the degree to which they represent a document.

In this study document attribute values are calculated using the frequency of feature words in a document. The frequency is then normalized as shown in Equation 2.1. The attribute values are scaled by the inverse document frequency factor (using the relation in Equation 2.3) to reflect how uniquely an attribute represents a document with respect to other documents of the same category. Moreover, an attribute is taken as document representative only if its frequency in the document is greater than one (before it is normalized).

For example, consider the two words ሶማሊያ (Somalia) and ሰራዊት (Army) that are found in two different documents that belong to the **Defense** news category. Both words are taken as attributes of their respective news documents because ሶማሊያ has a frequency of 4 and is found in 3 defense documents while ሰራዊት has frequency of 3 and is found in 22 defense documents. The word ሶማሊያ however has much bigger attribute value (2.03969) than the word ሰራዊት with an attribute value of (0.968088), since it is more discriminative.

After feature selection and data transformation the unique attributes of each of the 15 news categories (i.e., the collection of the unique attributes of each of the news items of a category) are identified as shown in Table 4.5

Table 4.5 News categories and their attributes

No	Category	No of attributes
1	Law and Justice	412
2	Health	779
3	International Relation	1046
4	Social Affairs	1802
5	Culture	192
6	Politics	894
7	Agriculture	916
8	Defense and Security	55
9	Science and Technology	176
10	Sport	397
11	Education	454
12	Economy	978
13	Accidents	100
14	Weather	37
15	Other classes	154

4.4.3 Data Conversion

After selection is complete and number of feature words per category are known the next step is converting the dataset to a format appropriate for automatic classification.

In this study the *Weka* application package is used to classify Amharic news documents. *Weka* is an open source data mining software developed at the University of WAKATO in New Zealand. The whole package is written in Java, so it can be run on any platform. The package offers three different interfaces.

- *A command line interface*
- *An Explorer GUI interface*: which allows different types of data preparation, and modeling algorithms on a dataset

- *An experimenter GUI interface*: which allow running different algorithms in batch and comparing the results [24, 25]

Weka expects the source data for classification to be in **Arff** (or in CSV) format. Since the data processing for this research is done in Visual Basic with SQL Server at the backend, the attributes of the news documents are in SQL table object. A sample of the processed data is shown in Table 4.6.

Table 4.6 Sample Experiment Data from the Defense Category

NO	DfWord	Frequency	DocNo	DocFrq	Weight
96	አሜሪካ	4	79	2	2.25182377795
90	ሶማሊያ	4	75	3	2.03969037574
49	ወታደራዊ	3	42	2	2.18590975624
158	ሀይለኞች	3	119	3	1.97998579448
64	ኮማንዶ	3	54	4	1.83388041846
168	መከላከያ	3	125	22	0.96808833446

Therefore the data in the SQL table have to be changed to **Arff** format. In **Arff** format the whole dataset is considered as a relation, in the same sense as the relational database. Each document in the dataset is an instance (row) of the relation (i.e., a record with values for the attributes) while all attributes of the dataset are fields (columns) of the relation. Moreover, the **Arff** format considers the news categories as attributes (nominal attributes whose values are the category labels).

Conversion of the processed data to **Arff** therefore requires conversion of the SQL table data to a type shown in Table 4.7

Table 4.7 Sample Arff file in tabular format

No	አሚሪካ Numeric	ሶማሊያ Numeric	መታ·ደራዊ Numeric	ሀይሎች Numeric	ኮሚሽን Numeric	መከላከያ Numeric	Class Nominal
1	2.251824	0	0	0	0	0	Defense
2	0	2.039690	0	0	0	0	Defense
3	0	0	2.185910	0	0	0	Defense
4	0	0	0	1.979986	0	0	Defense
5	0	0	0	0	1.833880	0	Defense
6	0	0	0	0	0	0.968088	Defense

4.5 TESTING CLASSIFIER ALGORITHMS

Once the dataset is converted to the **Arff** format, the Explorer GUI of *Weka* is used to test the performances of the selected classifiers. The *Weka* explorer has various options to preprocess the input data before the algorithm is applied for classification. Some of the pre-processing options that are relevant to this research experiment are summarized in section 4.4.1 below.

4.5.1 Preprocessing Options in Weka

- **Working with attributes:** After choosing the Explorer GUI, when the input data is opened the Explorer shows the attribute list. The list shows the attributes of the opened **Arff** file, gives the detail of a selected attribute (for example, it gives the count of missing, distinct, and unique values), more importantly it gives the option of removing attributes from the subsequent classification.

- **Working with Filters:** The input data can be transformed in various ways using the filter functionality in the Explorer. Numerous filtering options are available including:

  - *Attribute filters* which can be used to apply different methods of attribute extraction like χ^2, *Gain-ratio, and Information-gain.*

    Or attribute data transformation like changing numeric value to binary, string to nominal, and also normalizing attribute values.

  - *Instance (record) filters* which can be applied to resample instances or to normalize instances or to change non-sparse instances to sparse and vice -versa.

Some of the popular feature extraction methods like χ^2, Gain-ratio, and Information-gain as well as some of the instance filters have been tried in the experimenting stage of this research. These methods, available in Weka, could reduce the high processing and time cost of the classifier algorithms, without negatively affecting the classification accuracy. However, time constraints did not allow detail exploration of these methods (for reducing the computational cost of the automatic classification) for this research.

4.5.2 The Experiment

Because of the high cost of processing time and processing power required by most of the classifiers, and because the experiment has to be repeated ten to hundred times (with parameter tuning for SVM classifier) to get the best out of the classifiers, the experiment was done stage by stage. The testing started with the data of five

categories, then the number of categories is increased to ten and finally testing all fifteen categories.

The experiment was performed using the command line interface and the Explorer GUI of the Weka package. It was started with the five categories of Weather, Defense, Science, Accidents, and Culture. The five categories have a total of 429 attributes including one attribute for the category label (i.e., they have 28 attributes in common between the three of them) and a total of 2600 instances (see Table 4.8).

Table 4.8 The five categories with their attributes

Category	Weather	Defense	Science	Accidents	Culture
Weather	37	2	15	9	5
Defense	2	55	13	10	22
Science	15	13	176	22	36
Accidents	9	10	22	100	13
Culture	5	22	36	13	192

The whole experiment dataset has two types of attributes: numeric attributes for the weight of the feature words and nominal attributes for the class labels.

For the reasons discussed in Chapter 2, the classifiers used in the experiment are SVM and Decision Tree classifiers. The Naïve Bayes

classifier is also included in the experiment as a baseline for performance evaluation and comparison.

4.5.2.1 Classification Using Decision Tree Classifiers

Weka supports the following decision tree induction classifiers: ADTree, BFTree, DecisionStamp, Id3, J48, LMT, MSP, NBTree, RandomForest, RandomTree, REPTree, and Simplecast. However, not all of these algorithms could be used in the experiment. ADTree was not tried because the algorithm cannot handle non-binary class (the experimental data has nominal classes), Id3 cannot handle numeric attributes. MSP cannot handle nominal class.

Testing for five Categories:

Out of all the Decision Tree classifiers that can work with the experiment data, Logic Model Tree (LMT) classifier showed the best performance. For the five categories, the LMT correctly classified 93.45% of the 2,610 instances.

- **Summary statistics**

Correctly classified instances	2,439	93.45%
Incorrectly classified instances	171	6.55%

The testing on the dataset is done using 10-fold stratified cross-validation. **Weka** provides a number of options for measuring the performance of a classifier, out of which the summary statistics, detailed accuracy by class, and confusion matrix are shown in Table 4.9 for the LMT classifier.

Table 4.9: Five categories LMT Confusion Matrix

Weather	Defense	Science	Accidents	Culture	
321	1	2	4	5	Weather
0	163	6	5	12	Defense
3	7	574	11	34	Science
1	5	8	655	18	Accidents
7	9	15	18	726	Culture

Looking vertically at the value of a class in the confusion matrix, one can see the instances of a category as assigned by a classifier. For example, looking the above confusion matrix shows that the LMT classifier has correctly classified 321 instances of the Weather class (True Positives, TP) while 11 instances are False Positives (FP). Each row of the confusion matrix shows the actual number of instances in a category. For example, the confusion matrix shows that the actual number of instances in the Weather news category is 333 (321 True Positive instances and 12 False Negative instances).

Moreover, the confusion matrix can be taken as a data source for measures used for calculating the detail accuracy of each class, shown in Table 4.10.

Table 4.10 Five categories LMT detailed accuracy by class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.964	0.005	0.967	0.964	0.965	0.980	Weather
0.876	0.009	0.881	0.876	0.879	0.963	Defense
0.913	0.016	0.949	0.913	0.930	0.971	Science
0.953	0.020	0.945	0.953	0.949	0.972	Accidents
0.937	0.038	0.913	0.937	0.925	0.953	Culture

The selection of the LMT tree classifier as a best performer is based on only the accuracy (i.e., the number of news items classified correctly) of the classifier. The selection does not consider classifier's performance with respect to time and memory requirements. In fact, the LMT classifier is very slow. In a PC with 256 MB memory and over 3 GHz speed it takes several hours to build a model for the 10 folds (i.e., for the test option of 10-fold cross-validation of the data of the five categories shown in Table 4.9 and Table 4.10).

Out of the decision tree algorithms tested, the J48 classifier was the fastest requiring only 14.07 seconds to build the model out of the five categories data. The correctly classified instances by the classifier are, however, 74.11%.

Observation of the detailed category statistics shows that categories that have relatively larger number of instances per attribute have been classified more accurately than the others.

From Tables 4.4 and 4.5, it can be seen that the Attribute-to-Instance ratio of the five categories tested is as follows: Weather (0.111, i.e., $\frac{37}{333}$), Defense (0.296, i.e., $\frac{55}{186}$), Science (0.280, i.e., $\frac{176}{629}$),

Accidents (0.146, i.e., $\frac{100}{687}$), and Culture (0.248, i.e. $\frac{192}{775}$). It can be seen that Weather has largest instances per category, followed by the Accidents class, then Culture which is followed by the Science class, with the Defense class having the smallest instances per category.

Looking at the True Positive Rate (TP) in Table 4.10, i.e., the instances correctly classified out of all instances of a category, it can be seen that the Weather class has the highest TP-Rate of 96.4% followed by the Accidents class with 95.3%, then Culture (93.7%), then Science (91.3%) and Defense (87.6%).

However, looking only at the TP-Rate for measuring the performance of a classifier (or the class prediction of a category) could be misleading. Other measures like precision, the F-measure (to optimize for either precision or recall), and ROC-Area (to see the trade-off between TF-Rate and FP-Rate) are also important measures of performance.

Considering precision, recall, F-measure, and ROC-Area and observing the detail class statistics in Table 4.10, it can be seen that the LMT Tree classifier consistently gives better performance in the categorization of the Weather class, followed by that of the Accidents class, then Culture, Science, and least performance for the Defense class.

Testing for ten Categories:

- **Summary statistics**

For the ten categories the LMT correctly classified 89.98% of the 6,368 instances while 10.02% were classified incorrectly.

Correctly classified instances	5,730	89.98%
Incorrectly classified instances	638	10.02%

- **Confusion Matrix**

The confusion matrix for testing the ten categories is given in Table 4.11

Table 4.11 Ten category LMT Confusion Matrix

weather	defense	Science	accident	Culture	law	Others	Sport	Education	Agriculture	
88	1	2	0	0	5	3	1	4	6	weather
0	42	0	1	3	3	2	4	2	4	defense
2	0	167	3	9	6	3	3	8	7	science
0	0	3	195	7	6	2	3	4	7	accident
2	2	4	4	216	7	2	4	5	10	culture
5	3	7	9	11	891	5	12	15	16	law
0	0	2	6	4	10	113	9	9	12	others
6	4	6	9	10	11	5	1016	17	19	sport
3	1	6	8	11	14	6	20	1535	28	education
11	9	20	19	20	23	17	20	26	1467	Agriculture

Looking vertically in the confusion matrix, one can see the True Positive and the False Negative values of each category. These values are used to calculate the detail classification accuracy as shown in Table 4.12.

Table 4.12 Ten categories LMT detailed accuracy by class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.800	0.005	0.752	0.800	0.775	0.885	Weather
0.689	0.003	0.677	0.689	0.683	0.918	Defense
0.803	0.008	0.770	0.803	0.786	0.940	Science
0.859	0.010	0.768	0.859	0.811	0.945	Accidents
0.844	0.012	0.742	0.844	0.790	0.938	Culture
0.915	0.016	0.913	0.915	0.914	0.967	Law
0.685	0.007	0.715	0.685	0.700	0.919	Others
0.921	0.014	0.930	0.921	0.926	0.971	Sport
0.941	0.019	0.945	0.941	0.943	0.971	Education
0.899	0.023	0.931	0.899	0.915	0.961	Agriculture

Looking at the TP-Rate, the Precision, the F-measure, and the ROC-Area indicates that categories with relatively larger instances per attribute (Education, Sport, and Law) and categories with relatively large instances (Agriculture) are classified more accurately than the others.

Testing for fifteen Categories:

- **Summary statistics**

The LMT classifier showed an average accuracy of 79.72% for a dataset of 22,999 instances and fifteen categories.

Correctly classified instances	18,423	79.72%
Incorrectly classified instances	4,576	20.28%

Table 4.13 Fifteen categories LMT Detailed accuracy by class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.600	0.002	0.611	0.809	0.697	0.847	Weather
0.410	0.001	0.431	0.656	0.521	0.759	Defense
0.572	0.006	0.472	0.768	0.586	0.790	Science
0.604	0.006	0.481	0.781	0.596	0.794	Accidents
0.559	0.008	0.433	0.774	0.556	0.774	Culture
0.775	0.010	0.773	0.798	0.785	0.887	Law
0.467	0.004	0.445	0.659	0.532	0.763	Others
0.772	0.012	0.769	0.800	0.784	0.886	Sport
0.831	0.017	0.793	0.874	0.832	0.908	Education
0.744	0.018	0.760	0.780	0.770	0.876	Agriculture
0.799	0.018	0.775	0.869	0.820	0.900	Health
0.800	0.018	0.776	0.810	0.793	0.887	Politics
0.757	0.018	0.732	0.780	0.756	0.869	Inter Rel
0.804	0.019	0.810	0.870	0.839	0.910	Economy
0.849	0.024	0.900	0.911	0.905	0.940	Social

Observation of the fifteen categories dataset detail statistics in Table 4.13 reveals (from the values of the TP-Rate, the Precision, the F-measure, and the ROC-Area) that for categories with relatively larger instances in the dataset as well as relatively large instances per attribute (Social, Education, Economy, Politics, and Health) the LMT classifier shows the best performance. Least accuracy of the classifier is seen for categories with relatively small instances in the dataset (i.e. Defense, Other classes, and Culture).

- **Confusion Matrix**

The accuracy measures are calculated on the basis of the data shown in the confusion matrix for the 15 categories in Table 4.14.

Table 4.14 Fifteen categories LMT Confusion Matrix

weather	defense	science	accidents	culture	Law	Others	Sport	education	agri	Health	Politics	Inter Rel	Economy	Social	
66	0	2	3	3	1	1	4	4	6	3	3	2	5	7	weather
0	25	2	2	2	1	2	2	2	2	3	3	3	5	7	defense
2	0	119	3	5	2	2	3	10	9	9	6	6	13	19	science
0	1	4	137	4	5	3	6	6	7	10	8	6	12	18	accidents
2	2	4	5	143	5	2	6	11	14	12	10	6	15	19	culture
2	1	5	9	11	755	2	5	11	12	23	31	33	35	39	law
0	0	4	3	5	5	77	2	5	7	7	8	11	13	18	others
3	1	7	6	6	10	2	851	12	17	34	32	23	42	57	sport
2	1	8	8	9	13	5	20	1356	28	35	34	24	38	51	education
3	2	13	11	14	16	6	15	28	1214	48	55	62	70	75	agri
2	1	15	14	25	24	9	32	35	38	1753	37	42	60	107	Health
3	4	10	15	27	37	13	18	29	38	56	1791	57	63	79	Politics
4	6	11	13	19	22	13	32	40	39	47	58	1365	59	74	Inter Rel
7	6	18	24	24	28	14	35	53	51	87	89	84	2590	110	Economy
12	8	30	32	33	53	22	76	107	116	135	142	140	176	6093	Social

4.5.2.2 Classification using SVM

The Weka version 3.5.5 used for the experiment comes with the following SVM classifiers: GaussianProcesses (RBF kernel) and SMO (poly kernel). Moreover, Weka supports an add-on SVM function library of SVM.

The performances of only the Library of SVM (LibSVM) and the Sequential Minimal Optimization (SMO) classifiers were tested on the dataset because the GaussianProcesses function cannot handle nominal classes. The SVM classifiers require parameter tuning to find values at which they perform best. They also require relatively long time for model building from the dataset.

After testing the dataset with both the SOM and the LibSVM classifiers, the LibSVM classifier was found to be better with the following performance measures.

Testing for five Categories:

- **Summary statistics**

Correctly classified instances	2,485	95.21%
Incorrectly classified instances	125	4.79%

The LibSvm classifier was tested on the same dataset of five classes used to test the decision tree classifier. The experiment was performed using 10-fold stratified cross-validation. The overall accuracy of the LibSVM classifier was found to be 95.21%, i.e.,

correctly classifying 2,485 documents and misclassifying 125 out of the total of 2,610 documents.

Table 4.15 Five category LibSVM detailed accuracy by class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.970	0.003	0.982	0.970	0.976	0.987	Weather
0.887	0.007	0.912	0.887	0.899	0.970	Defense
0.949	0.013	0.958	0.949	0.954	0.979	Science
0.961	0.017	0.954	0.961	0.957	0.977	Accidents
0.955	0.025	0.943	0.955	0.949	0.969	Culture

Observation of the detail accuracy for each category in table 4.15 reveals that the SVM classifier also performs better for categories that have larger instances per attribute. Therefore, a relatively better classifier performance is seen on the Weather, Accidents, and Culture classes, followed by the Science and Defense classes.

- **Confusion Matrix**

Table 4.16 Five categories LibSVM Confusion Matrix

Weather	Defense	Science	Accidents	Culture	
323	1	2	2	5	Weather
0	165	5	7	9	Defense
2	4	597	9	17	Science
1	4	8	660	14	accidents
3	7	11	14	740	Culture

The confusion matrix shown in Table 4.16 confirms the detail class statistical values used to determine the class level accuracy of the classifier.

With regard to the performance measures of time and memory requirements, the LibSVM classifier was found to be faster than the LMT Tree classifier. However the LibSVM requires several test runs and parameter tuning to arrive at the best prediction values.

Testing for ten Categories:

For the ten categories, the LibSVM classifier shows an average accuracy of 91.36% for the dataset of 6,368 instances with 8.64% classification error.

- **Summary statistics**

Correctly classified instances	5,818	91.36%
Incorrectly classified instances	550	8.64%

Table 4.17 Ten category LibSVM detailed accuracy by class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.882	0.003	0.829	0.882	0.855	0.926	Weather
0.705	0.003	0.717	0.705	0.711	0.926	Defense
0.808	0.008	0.774	0.808	0.791	0.942	Science
0.877	0.007	0.829	0.877	0.852	0.958	Accidents
0.859	0.012	0.756	0.859	0.805	0.942	Culture
0.919	0.011	0.938	0.919	0.928	0.974	Law
0.715	0.007	0.733	0.715	0.724	0.926	Others
0.937	0.011	0.945	0.937	0.941	0.977	Sport
0.950	0.018	0.949	0.950	0.950	0.974	education
0.915	0.023	0.933	0.915	0.924	0.964	Agriculture

For the ten categories dataset, observation of the values of the TP-Rate, the Precision, the F-measure, and the ROC-Area in Table 4.17shows that LibSVM classifier also has better accuracy for categories with relatively larger instances in the dataset (Education, Sport, Law, and Agriculture) and for categories with relatively large instances per attribute (Weather).

Table 4.18 Ten categories LibSVM Confusion Matrix

Weather	defense	Science	accidents	culture	law	others	Sport	Education	agriculture	
97	0	1	0	1	1	2	1	3	4	Weather
0	43	0	1	5	1	2	3	2	4	Defense
1	0	168	3	11	4	3	2	8	8	Science
1	1	3	199	5	3	2	2	5	6	Accidents
1	2	5	3	220	3	3	4	6	9	Culture
3	3	8	9	9	895	5	11	13	18	Law
0	1	3	3	5	5	118	6	10	14	Others
4	3	7	6	5	9	6	1033	11	19	Sport
2	1	7	5	10	12	7	12	1551	25	Education
8	6	15	11	20	21	13	19	25	1494	Agriculture

Confirmation of the values used in the class detail accuracy, is given by the confusion matrix for the ten categories in Table 4.18. For example, the True Positive Rate of the Education category is calculated from the data in the confusion matrix (Table 4.18) as follows:

$$TP\ \text{Rate} = \frac{1551}{(2+1+7+5+10+12+7+12+1551+25)} = 0.950$$

Testing for fifteen Categories:

- **Summary statistics**

The LibSVM classifier showed an average accuracy of 81.15% for a dataset of 22,999 instances and fifteen categories while 18.85% of the instances were incorrectly classified.

Correctly classified instances	18,664	81.15%
Incorrectly classified instances	4,335	18.85%

Table 4.19 Fifteen categories LibSVM detailed accuracy by class

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0.645	0.002	0.664	0.645	0.654	0.826	Weather
0.443	0.001	0.509	0.443	0.474	0.736	Defense
0.692	0.005	0.535	0.692	0.604	0.799	Science
0.744	0.006	0.537	0.744	0.624	0.809	Accidents
0.746	0.008	0.508	0.746	0.605	0.798	Culture
0.781	0.009	0.799	0.781	0.790	0.891	Law
0.509	0.004	0.488	0.509	0.499	0.747	Others
0.772	0.010	0.791	0.772	0.782	0.886	Sport
0.809	0.016	0.790	0.809	0.800	0.892	Education
0.760	0.017	0.774	0.760	0.767	0.875	Agriculture
0.804	0.017	0.793	0.804	0.798	0.890	Health
0.798	0.017	0.790	0.798	0.794	0.888	Politics
0.787	0.017	0.757	0.787	0.772	0.877	Inter Rel
0.807	0.018	0.823	0.807	0.815	0.898	Economy
0.869	0.023	0.906	0.869	0.887	0.932	Social

Observation of the detail statistics from the values of the TP-Rate, the Precision, the F-measure, and the ROC-Area reveals that for 15 categories (Table 4.19) dataset, the LibSVM classifier (like the LMT

tree classifier) shows best performance for categories with relatively larger instances in the dataset as well as relatively large instances per attribute (Social, Education, Economy, and Health). The classifier's accuracy is least for categories with relatively small instances in the dataset (i.e., Defense, Other Classes, and Weather).

- **Confusion Matrix**

The accuracy measures are calculated on the basis of the data in Table 4.20, for the confusion matrix of the 15 categories.

Table 4.20 Fifteen category LibSVM Confusion Matrix

weather	defense	science	accidents	Culture	Law	Others	Sport	education	agriculture	Health	Politics	Inter Rel	Economy	Social	
71	0	2	2	3	1	1	3	4	5	3	3	2	4	6	weather
0	27	2	2	2	1	2	2	2	2	3	3	2	5	6	defense
1	0	144	2	4	2	1	1	8	6	5	4	5	9	16	science
0	0	2	169	4	3	1	5	3	3	4	4	5	11	13	accidents
1	1	3	2	191	2	1	3	5	7	6	8	4	8	14	culture
2	1	5	6	9	761	1	5	12	13	22	31	34	35	37	law
0	0	3	3	5	5	84	3	4	6	7	8	8	12	17	others
2	0	6	5	8	7	2	852	22	18	34	29	23	41	54	sport
1	1	8	9	12	10	5	22	1321	36	34	35	24	54	60	education
2	1	11	8	13	14	4	12	27	1241	47	53	61	67	71	agriculture
2	0	17	19	23	24	9	37	39	45	1763	43	39	53	81	Health
2	4	9	13	25	34	11	17	30	37	55	1788	63	65	87	Politics
5	5	9	13	14	18	13	26	34	31	43	51	1418	54	68	Inter Rel
7	5	21	29	28	30	16	36	59	56	73	78	65	2599	118	Economy
11	8	27	33	35	40	21	53	102	98	125	126	119	142	6235	Social

4.5.2.3 Comparison of the DTree and the SVM classifiers

Section 4.5.2.2 shows the test on the classification accuracy of the decision tree (LMT) and SVM (LibSVM) classifiers. The testing is done separately for 5 categories data, then for 10 categories, and then for 15 categories.

It can be seen in the experimentally generated data that 10-fold cross-validation is used to test the accuracy of the classifiers for 5-category data. For 10-category and 15-category data however, the holdout method is used (with 66% of the data used for training a classifier and 33% for performance testing) because of the high performance cost of both the DTree and SVM classifiers.

Table 4.21 shows the summary of the average accuracy of LMT and LibSVM classifiers for the experiment with 5, 10, and 15-category datasets.

Table 4.21 Average accuracy of LMT and LibSVM classifiers

Classifier	Average Accuracy for 5-category data	Average Accuracy for 10-category data	Average Accuracy for 15-category data
LibSVM	95.21%	91.36%	81.15%
LMT	93.45%	89.98%	79.72%

Table 4.21 shows that LibSVM seems to give better classification accuracy in all the 5, 10, and 15-category dataset experiments. The Experimenter of the Weka package is used to test if there is a statistically significant performance difference between the two classifiers. The Package provides the option of comparison of

different classifiers. The Experimenter GUI or the command prompt can be used for the comparison.

The two classifiers used in the experiment: The LMT and the LibSVM classifiers are compared for performance while the Naïve Bayes classifier is used as a baseline.

The testing was done for the 5-categories dataset, i.e., for an experiment type of 10-fold stratified cross-validation. The selected tester is the corrected paired T-Test (correcting for dependencies in estimates that may be biased to give a result showing significant difference). The comparison field chosen is percent correct (percent of instances classified correctly).

The result of the comparison is analyzed using Weka showing the following results.

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 3
Confidence: 0.05 (two tailed)

Dataset                  (3) bayes | (1) funct (2) trees
-----------------------------------------------------------
AmhaicNews      (100)   73.33 |   95.44 v   93.21 v
-----------------------------------------------------------
                         (v/ /*) |  (1/0/0)  (1/0/0)
```

Key:

(1) functions.LibSVM '-S 0 -K 2 -D 3 -G 0.1 -R 0.3 -N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1' 14172

(2) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943

(3) bayes.NaiveBayes '' 5995231201785697655

The symbol v indicates that a specific result is statistically better than the baseline scheme. Therefore, the comparison result shows that both LibSVM and LMT classifiers have shown significantly better classification accuracy (at the significance level of 0.05) than the baseline classifier.

Comparison between the LibSVM and the LMT classifiers

Tester: weka.experiment.PairedCorrectedTTester
Analysing: Percent_correct
Datasets: 1
Resultsets: 2
Confidence: 0.05 (two tailed)

Dataset	(1) trees	(2) functions
AmharicNews (100)	93.22	94.45
	(v/ /*)	(0/0/1)

Key:

(1) trees.LMT '-I -1 -M 15 -W 0.0' -1113212459618104943

(2) functions.LibSVM '-S 0 -K 2 -D 3 -G 0.3 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.0010 -P 0.1' 14172

The classification accuracy comparison test between the LibSVM and LMT classifiers sows that there is no statistically significant difference between the two classifiers.

Tests for the datasets of 10 and 15 categories also showed that there is no statistically significant performance (accuracy) difference between the LibSVM and LMT classifiers.

CHAPTER FIVE

CONCLUSIONS AND RECOMANDATIONS

5.1 CONCLUSIONS

Rapid developments in Information and Communication Technology are making available huge amount of data and information. Much of these data are in electronics forms (like electronic publications, e-mail, the Internet, etc.). These data do not follow a standard format like the relational database and are available in various forms from various sources. Much of the data is unstructured or semi-structured and can generally be considered as a text database.

Text databases are showing accelerated growth in all corners of the world. As a result there is an active field of study in text mining to facilitate the extraction of useful data and information from text databases.

The text data in local languages is also increasing fast requiring text processing tools for text documents to be available in local languages. This is true for Amharic also, as can be surmised from the recent boom of online newspapers, magazines, data in electronics storage, etc.

A number of researches on automatic processing of Amharic text have recently been conducted. This research work in Automatic Amharic Text Categorization is an effort to contribute in this direction.

In this section the direction followed in this research, the main contributions of the research and the conclusions reached are summarized.

As discussed in Chapters 3 and 4, documents are represented by feature words. Because of the problems of the Amharic writing system and unavailability of Amharic text processing tools, the focus of the research was on developing a document processing scheme which facilitates efficient automatic classification of Amharic documents.

To this end, much attention was given on the processing of the source data by developing and enhancing the following tools. The tools are specific to the source data – Amharic news documents from ENA.

- A tool to correct word spelling variations, focusing on spelling variation due to transliterations and due to usage variation at different localities.
- Enhancement to the suffix/prefix removal tool developed in [7] so that it can differentiate letters in the affix list when they are really parts of a word or an affix to it.
- A tool to correct word variations due to gender marker suffixes.

- A tool to correct word variations due to number marker suffixes.
- A tool to merge compound words (when they may result in semantic loss if separated) written as separate words.

The use of these tools (which enabled 10 to 30% feature reduction) in addition to other tools and data reduction methods helped to analyze the huge source data (69,684 news items after data cleaning) and measure classifier performances. However, the experiment was time intensive, taking several hours using high capacity computers (Computers with 512 MB RAM and 3.7 GHz speed).

From this work on automatic classification of Amharic texts, it can be concluded that

- The Amharic writing system presents unique data pre-processing challenges
- The data pre-processing for the automatic classification should be aided by comprehensive Natural Language Processing to be effective (i.e., to develop effective data pre-processing tools and use the tools)
- Both LMT and LibSVM classifiers have high accuracy but with high computational cost
- Both LMT and LibSVM classifiers showed better accuracy for categories with relatively large number of documents (instances) and for categories with relatively large number of instances per attribute
- There is no significant performance (accuracy) difference between LMT and LibSVM classifiers

- Both LMT and LibSVM have significantly better accuracy than the Naïve Bayes classifier

5.2 RECOMMANDATIONS

The results of this research indicate the need for Amharic text processing tools for efficient automatic classification. Removal of redundant and irrelevant attributes of documents has a determining effect on the use of text classifiers - in real situations and on large datasets it could decide whether or not a classifier is useful.

Although text classifiers are available off-the-shelf, they may not show good performance unless aided by an efficient pre-processing of the source data.

Therefore much more has to be done to ensure automatic processing of Amharic texts for all situations. The following are some of the areas identified in this research for future work.

1. Highly customized tool for correcting spelling variations was developed in this research. Full-fledged Amharic spelling checker, which addresses the various causes of spelling variations need to be developed.

2. A tool has been developed to merge separated compound words so as to prevent the semantic loss that could happen as a result of the separation. Further investigation is needed to see the importance of merging separated compound words. Especially with the existing non-uniform usage of the compound words (sometimes as a single word sometimes as separate words).

3. In this research a number of tools have been developed and enhanced to reduce word variation due to the various language requirements affixing. These separate and incomplete efforts should be replaced by the availability of a complete Amharic stemmer.

4. The SVM and Decision Tree classifiers used in this research have shown good accuracy. They both have, however, high performance cost (processing power and time cost). Therefore, there is a need to look for other classifiers with less processing cost and better accuracy.

5. Huge data has been available for this research work. Many hours of work has been spent on pre-processing the data. The work on the data is by no means complete, mainly because of the size of the data. Further work by researchers on these data could lead to the development of the much needed Amharic corpus.

6. Experience from this research shows that there is very little cooperation among researchers working to automate the information processing and retrieval tasks - from Amharic data sources. Better cooperation among researchers as well as coordination of the various individual efforts could result in more efficient and standard tools and methods for mining Amharic data sources. Moreover, standard tools developed for Amharic could be used for other languages that use the Ethiopic script.

BIBLIOGRAPHY

[1] J. Han and M. Kamber. *Data Mining: Concepts and techniques* (2nd ed.). Morgan Kaufmann Publishers, 2006.

[2] R. M. Duwairi. A Distance-based Classifier for Arabic Text Categorization. In *Proc. The 2005 International Conference on Data Mining, DMIN 2005*, Las Vegas, Nevada June 2005.

[3] T. Joachims. Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *Proc. of ECML-98, 10th European Conference on Machine Learning*, pages 137-141, 1998.

[4] Internet States. World Internet Usage and Population statistics. http://www.internetworldstats.com/status.htm, December, 2006.

[5] Samuel Eyassu and Björn Gambäck. Classifying Amharic News Text Using Self-organizing Maps. In *Proc. of ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics*, Ann Arbor, Michigan, Jun. 2005.

[6] Atelach Alemu and Lars Asker. Dictionary – based Amharic – French Information Retrieval. Department of Computer and Systems Sciences, Stockholm University, 2005.

[7] Zelalem Sintayehu. Automatic Classification of Amharic News Items: The Case of Ethiopian News Agency. School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2001.

[8] Surafel Teklu. Automatic Categorization of Amharic News Text: A Machine Learning Approach. School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2003.

[9] Saba Amsalu. The application of Information Retrieval Techniques to Amharic Documents in the Web. School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2001.

[10] Bethlehem Mengistu. The Application of N-gram Indexing in Amharic Text Retrieval. School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2002.

[11] Theodros Gebre-Meskel. Automatic Text Retrieval: An Experimenting using Latent Semantic Indexing (LSI) with Singular Value Decomposition (SVD). School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2003.

[12] Abiyot Bayou. Design and Development of Word Parser for Amharic Language. School of Information Studies for Africa, Addis Ababa University, Addis Ababa, 2000.

[13] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. *Inductive Learing Algorithms and Representations for Text Categorization*, 1998.

[14] D.D. Lewis and M. Ringuette. Comparision of Two Learning Algorithms for Text Categorization. In *Proc. of the Third Annual Symposium on Document Analysis and Information Retrieval (SDAIR '94)*, 1994.

[15] I. Moulinier. *Is Learning Bias an Issue in Categorization Problem?* Technical Report. LAFORIA-LIP6, Universite Paris VI, 1997.

[16] E. Wiener, J.O. Pederson, and A.S. Weigend. A Neural Network Approach to Topic Spotting. In *Proc. of the Fourth Annual Symposium on Document Analysis and Information Retrieval (SDAIR '95)*, 1995.

[17] H.T. Ng, W.B. Goh, and K.L. Low. Feature Selection, Perception Learning and a Usability Case Study for Text Categorization. In: *20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '97), pages 6773*, 1997.

[18] J. Yan, N. Liu, B. Zhang, and S. Yan. *Optimal Orthogonal Centroid Feature Selection (OCFS) for Text Categorization*, 2005.

[19] Nega Alemayehu and Peter Willett. *Stemming of Amharic Words for Information Retrieval. Literary Linguistic Computing Vol. 17, No.1*, 2002.

[20] B. Yu. *Comparative Literary Style Mining between Native and Non-native English Writers.* Graduate School of Library and Information Science, University of Illinois, 2005.

[21] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* Massachusetts Institute of Technology, 2001.

[22] Getachew Haile. *The Problems of the Amharic Writing System.* Unpublished, 1966.

[23] ENA. *Problems of the Existing News Classification System and Suggested Solutions.* Unpublished, July 2006.

[24] D. Scuse and P. Reutemann. *Weka Experimenter Tutorial for Version 3-5-3.* The University of Waikato, 2006.

[25] R. Kirk and E. Frank. *Weka Explorer User Guide for Version 3-5-3.* The University of Waikato, 2006.

[26] Y. Yang. *An Evaluation of Statistical Approaches to Text Categorization.* The Netherlands: Kluwer Academic Publishers, 1999.

[27] R. Beza-Yates and R. Berthier. *Modern Information Retrieval.* Harlow, England: Addison Wesley, 1999.

[28] G. Salton. *Introduction to Modern Information Retrieval.* New York: McGraw Hill.

[29] F. Sebastiani. Machine Learning in Automated Text Classification. In *ACM. Computing Surveys. Vol.34 No. 1, pages 1-47,* 2002.

ANNEXES

Annex 1: The Amharic character set and numbers (Bender et al. as referenced in [8])

Order							Labialized				
1st	2nd	3rd	4th	5th	6th	7th					
ሀ	ሁ	ሂ	ሃ	ሄ	ህ	ሆ					
ለ	ሉ	ሊ	ላ	ሌ	ል	ሎ			ሏ		
ሐ	ሑ	ሒ	ሓ	ሔ	ሕ	ሖ					
መ	ሙ	ሚ	ማ	ሜ	ም	ሞ			ሟ		
ሠ	ሡ	ሢ	ሣ	ሤ	ሥ	ሦ					
ረ	ሩ	ሪ	ራ	ሬ	ር	ሮ			ሯ		
ሰ	ሱ	ሲ	ሳ	ሴ	ስ	ሶ			ሷ		
ሸ	ሹ	ሺ	ሻ	ሼ	ሽ	ሾ			ሿ		
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ	ቈ	ቍ	ቋ	ቌ	ቍ
በ	ቡ	ቢ	ባ	ቤ	ብ	ቦ			ቧ		
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ			ቷ		
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ			ቿ		
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ	ኈ	ኍ	ኋ	ኌ	ኍ
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ			ኗ		
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ			ኟ		
አ	ኡ	ኢ	ኣ	ኤ	እ	ኦ					
ወ	ዉ	ዊ	ዋ	ዌ	ው	ዎ					
ዐ	ዑ	ዒ	ዓ	ዔ	ዕ	ዖ					
ከ	ኩ	ኪ	ካ	ኬ	ክ	ኮ	ኰ	ኵ	ኳ	ኴ	ኵ
ኸ	ኹ	ኺ	ኻ	ኼ	ኽ	ኾ					
ዘ	ዙ	ዚ	ዛ	ዜ	ዝ	ዞ			ዟ		
ዠ	ዡ	ዢ	ዣ	ዤ	ዥ	ዦ					
የ	ዩ	ዪ	ያ	ዬ	ይ	ዮ					
ደ	ዱ	ዲ	ዳ	ዴ	ድ	ዶ			ዷ		
ጀ	ጁ	ጂ	ጃ	ጄ	ጅ	ጆ					
ገ	ጉ	ጊ	ጋ	ጌ	ግ	ጎ	ጐ	ጕ	ጓ	ጔ	ጕ
ጠ	ጡ	ጢ	ጣ	ጤ	ጥ	ጦ			ጧ		
ጨ	ጩ	ጪ	ጫ	ጬ	ጭ	ጮ			ጯ		
ጰ	ጱ	ጲ	ጳ	ጴ	ጵ	ጶ			ጷ		
ፀ	ፁ	ፂ	ፃ	ፄ	ፅ	ፆ					
ፈ	ፉ	ፊ	ፋ	ፌ	ፍ	ፎ			ፏ		
ፐ	ፑ	ፒ	ፓ	ፔ	ፕ	ፖ					

𖿡	𖿢	𖿣	𖿤	𖿥	𖿦	𖿧

1	2	3	4	5	6	7	8	9

10	20	30	40	50	60	70	80	90	100

Annex 2: List of symbols used in the Visual Geez-2 for the Amharic FIDEL [8]

ሀ	ሀ·	ሂ.	ሃ	ሄ	ህ	ሆ
h	h#	£	¦	ÿ	H	Ç
ለ	ለ·	ለ.	ላ	ሌ	ል	ሎ
l	l#	l!	§	l@	L	lÖ
ሐ	ሐ·	ሐ.	ሓ	ሔ	ሕ	ሖ
/	/#	/!	ˆ	/@	?	‡
መ	መ·	ሚ.	ማ	ሜ	ም	ሞ
m	Ñ	¸	¥	»	M	Ä
ሠ	ሠ·	ሢ.	ሣ	ሤ	ሥ	ሦ
\	\#	œ!	œ	œ@	|	ƒ
ረ	ሩ·	ሪ	ራ	ሬ	ር	ሮ
r	„	¶	‰	È	R	É
ሰ	ሰ·	ሰ.	ሳ	ሴ	ስ	ሶ
s	s#	s!	ú	s@	S	Î
ሸ	ሸ·	ሸ.	ሻ	ሼ	ሽ	ሾ
ˢ	ˢ#	ˢ!	š	ˢ@	>	Ë
ቀ	ቁ	ቂ	ቃ	ቄ	ቅ	ቆ
q	q$	qE	”	q&	Q	Ö
በ	በ·	በ.	ባ	ቤ	ብ	ቦ
b	b#	b!	Æ	b@	B	Ï
ተ	ቱ	ቲ	ታ	ቴ	ት	ቶ
t	t$	tE	¬	t&	T	È
ቸ	ቹ	ቺ	ቻ	ቼ	ች	ቾ
c	c$	cE	Ò	c&	C	Ó
ኀ	ኁ	ኂ	ኃ	ኄ	ኅ	ኆ
x	x#	x!	¨	x@	X	å
ነ	ኑ	ኒ	ና	ኔ	ን	ኖ
n	n#	n!	Â	n@	N	Ñ
ኘ	ኙ	ኚ	ኛ	ኜ	ኝ	ኞ
ß	ß #	ß!	¾	ß @	"	®
ከ	ኩ·	ኪ.	ካ	ኬ	ክ	ኮ
k	k#	k!	µ	k@	K	÷
ኸ	ኹ·	ኺ.	ኻ	ኼ	ኽ	ኾ
ˊ	ˊ#	ˊ!	á	ˊ@	<	Ó
ወ	ዉ·	ዊ.	ዋ	ዌ	ው·	ዎ
w	ý	êE	ê	è&	W	Ã
ዐ	ዑ·	ዒ.	ዓ	ዔ	ዕ	ዖ

;	;#	›!	›	›@	:	Â
ıı	ıı·	ıı.	ч	ıь	ıı	ıı
z	z#	z!	²	z@	Z	Ø
ıſ	ıſ·	ıſ.	ıſ	ıſ	ıſ	ıſ
¢	¢$	¢E	Ï	¢&	™	Î
ſ	ſ·	ſ.	ſ	ſь	ſ	ſ·
y	†	‘	Ã	ü	Y	×
ſ	ſ·	ſ.	ſ	ſь	ſ·	ſ
d	Ç	Ä!	Ä	Á	D	Ì
ſ	ſ·	ſ.	ſ	ſь	ſ	ſ
j	°	©!	©	Ë	J	Í
ı	ı·	ı.	ı	ıь	ı	ı
g	g#	g!	U	g@	G	¯
ı	ı·	ı.	ı	ıь	ı	ı
-	-#	-!	È	-@	_	Ŏ
ıı	ıı	ıı.	ıı	ıı	ıı	ıı
=	=#	À	Å	Œ	+	Ô
ſ	ſ·	ſ.	ſ	ſь	ſ	ſ
’	’#	’!	Ô	’@	ù	Õ
ſ	ſ·	ſ.	ſ	ſь	ſ	ſ
]	]#	É!	É	É@	}	Ò
ſ	ſ·	ſ.	ſ	ſь	ſ	ſ
[	[.	[!	Ú	[@	A	Û
ſ	ſ·	ſ	ſ	ſь	ſ	ſ
f	û	ð	Í	ø	F	Æ
ſ	ſ	ſ	ſ	ſ	ſ	ſ
p	p$	pE	·	p&	P	±
ı	ı·	ı.	ı	ıь	ı	ı
v	v#	v!	Š	v@	V	‹
ſ	ſ	ſ	ſ	ſ	ſ	ſ
%	Đ	–	—	,	“	a
ſ	ſ	ſ	ſ	ſ	ſ	
¡	•	…	Þ	à	O	
ıı	ſ		ſ	ſ	ſ	ı
þ	ª		Ì	Ù	“	Æ*

The diacritic markings

!	@	#	$	&	*	E~